Course Timetabling

The weekly scheduling of the lectures/events/courses of courses avoiding students, teachers and room conflicts.

Input:
- A set of courses $C = \{C_1, \ldots, C_n\}$ each consisting of a set of lectures $C_i = \{L_{i1}, \ldots, L_{il_i}\}$. Alternatively, a set of lectures $L = \{L_1, \ldots, L_l\}$.
- A set of curricula $S = \{S_1, \ldots, S_r\}$ that are groups of courses with common students (curriculum based model). Alternatively, A set of enrollments $S = \{S_1, \ldots, S_s\}$ that are groups of courses that a student wants to attend (Post enrollment model).
- A set of time slots $T = \{T_1, \ldots, T_p\}$ (the available periods in the scheduling horizon, one week).
- All lectures have the same duration (say one period)

Output:
An assignment of each lecture $L_i$ to some period in such a way that no student is required to take more than one lecture at a time.
Graph model

Graph \( G = (V, E) \):
- \( V \) correspond to lectures \( L_i \)
- \( E \) correspond to conflicts between lectures due to curricula or enrollments

Time slots are colors \( \Rightarrow \) Graph-Vertex Coloring problem \( \Rightarrow \) NP-complete (exact solvers max 100 vertices)

Typical further constraints:
- Unavailabilities
- Preassignments

The overall problem can still be modeled as Graph-Vertex Coloring. How?

IP model

Including the assignment of indistinguishable rooms
\( m_t \) rooms \( \Rightarrow \) maximum number of lectures in time slot \( t \)

Variables
\[ x_{it} \in \{0,1\} \quad i = 1, \ldots, n; \ t = 1, \ldots, p \]

Number of lectures per course
\[ \sum_{t=1}^{p} x_{it} = l_i \quad \forall i = 1, \ldots, n \]

Number of lectures per time slot
\[ \sum_{i=1}^{n} x_{it} \leq m_t \quad \forall t = 1, \ldots, p \]

Further complications:
- Teachers that teach more than one course
  (not really a complication: treated similarly to students’ enrollment)
- A set of rooms \( R = \{R_1, \ldots, R_n\} \)
  with eligibility constraints
  (this can be modeled as Hypergraph Coloring [de Werra, 1985]:
    - introduce an (hyper)edge for events that can be scheduled in the same room
    - the edge cannot have more colors than the rooms available of that type)

Moreover,
- Students’ fairness
- Logistic constraints: not two adjacent lectures if at different campus
- Max number of lectures in a single day and changes of campuses.
- Precedence constraints
- Periods of variable length

Number of lectures per time slot (students’ perspective)
\[ \sum_{C_i \in S_j} x_{it} \leq 1 \quad \forall i = 1, \ldots, n; \ t = 1, \ldots, p \]

If some preferences are added:
\[ \max \sum_{i=1}^{p} \sum_{t=1}^{n} d_{it} x_{it} \]

Corresponds to a bounded coloring. [de Werra, 1985]
IP approach

3D IP model including room eligibility [Lach and Lübbecke, 2008]

\[ R(c) \subseteq \mathbb{R} : \text{rooms eligible for course } c \]

\[ G_{\text{conf}} = (V_{\text{conf}}, E_{\text{conf}}) : \text{conflict graph (vertices are pairs } (c, t)) \]

\[
\min \sum_{c \in C} d(c, t)x_{ctr} \quad \forall c \in C
\]

\[
\sum_{t \in T} x_{ctr} = l(c) \quad \forall c \in C
\]

\[
\sum_{c \in R^{-1}(r)} x_{ctr} \leq 1 \quad \forall t \in T, r \in R
\]

\[
\sum_{r \in R(c_1)} x_{c_1,t_1 r} + \sum_{r \in R(c_2)} x_{c_2,t_2 r} \leq 1 \quad \forall (c_1, t_1), (c_2, t_2)) \in E_{\text{conf}}
\]

\[ x_{ctr} \in \{0, 1\} \quad \forall (c, t) \in V_{\text{conf}}, r \in R \]

This 3D model is too large in size and computationally hard to solve.

Hall’s constraints

(guarantee that in stage 1 we find only solutions that are feasible for stage 2)

\[ G_t = (C_t \cup R_t, E_t) \text{ bipartite graph for each } t \]

\[ G = \cup_t G_t \]

\[
\sum_{c \in U} x_{ct} \leq |N(U)| \quad \forall U \in C, t \in T
\]

If some preferences are added:

\[
\max \sum_{i=1}^p \sum_{t=1}^n d_{it} x_{it}
\]

2D IP model including room eligibility [Lach and Lübbecke, 2008]

Decomposition of the problem in two stages:

Stage 1 assign courses to timeslots

Stage 2 match courses with rooms within each timeslot solved by bipartite matching

Model in stage 1

Variables: course \( c \) assigned to time slot \( t \)

\[ x_{ct} \in \{0, 1\} \quad c \in C, t \in T \]

Edge constraints
(forbids that \( c_1 \) is assigned to \( t_1 \) and \( c_2 \) to \( t_2 \) simultaneously)

\[ x_{c_1,t_1} + x_{c_2,t_2} \leq 1 \quad \forall ((c_1, t_1), (c_2, t_2)) \in E_{\text{conf}} \]

Hall’s constraints are exponentially many

[Lach and Lübbecke] study the polytope of the bipartite matching and find strengthening conditions

(polytope: convex hull of all incidence vectors defining subsets of \( C \) perfectly matched)

Algorithm for generating all facets not given but claimed efficient

Could solve the overall problem by branch and cut (separation problem is easy).

However the the number of facet inducing Hall inequalities is in practice rather small hence they can be generated all at once.
So far feasibility.

Preferences (soft constraints) may be introduced [Lach and Lübbecke, 2008b]

- Compactness or distribution
- Minimum working days
- Room stability
- Student min max load per day
- Travel distance
- Room eligibility
- Double lectures
- Professors’ preferences for time slots

Different ways to model them exist. Often the auxiliary variables have to be introduced

---

**Examination Timetabling**

By substituting lecture with exam we have the same problem! However:

<table>
<thead>
<tr>
<th>Course Timetabling</th>
<th>Exam Timetabling</th>
</tr>
</thead>
<tbody>
<tr>
<td>limited number of time slots</td>
<td>unlimited number of time slots, seek to minimize</td>
</tr>
<tr>
<td>conflicts in single slots, seek to compact</td>
<td>conflicts may involve entire days and consecutive days, seek to spread</td>
</tr>
<tr>
<td>one single course per room</td>
<td>possibility to set more than one exam in a room with capacity constraints</td>
</tr>
<tr>
<td>lectures have fixed duration</td>
<td>exams have different duration</td>
</tr>
</tbody>
</table>

---

**2007 Competition**

- Constraint Programming is shown by [Cambazard et al. (PATAT 2008)] to be not yet competitive
- Integer programming is promising [Lach and Lübbecke] and under active development (see J.Marecek [http://www.cs.nott.ac.uk/~jxm/timetabling/]) however it was not possible to submit solvers that make use of IP commercial programs
- Two teams submitted to all three tracks:
  - [Ibaraki, 2008] models everything in terms of CSP in its optimization counterpart. The CSP solver is relatively simple, binary variables + tabu search
  - [Tomas Mueller, 2008] developed an open source Constraint Solver Library based on local search to tackle University course timetabling problems [http://www.unitime.org]
  - All methods ranked in the first positions are heuristic methods based on local search

---

**Finalist Ordering**

The following information about the finals for each track is to be ordered. The list has been ordered by time slot, by name, course and school.

**Examination Track**

Examinations may not be allowed free. In the single time slot, the slot station entries in the exam.

1. Place: Thomas Riler (UK)
2. Place: Olivier (France)
3. Place: Alexandru Costin (Romania)
4. Place: Juriy Kuznetsov (Kazakhstan)
5. Place: Valeriya Volynets (Ukraine)
6. Place: Hector Gaviño (Spain)
7. Place: Tomas Riler (UK)
8. Place: Alexandru Costin (Romania)
9. Place: Valeriya Volynets (Ukraine)
10. Place: Juriy Kuznetsov (Kazakhstan)
Heuristic Methods

Hybrid Heuristic Methods
- Some metaheuristic solve the general problem while others or exact algorithms solve the special problem.
- Replace a component of a metaheuristic with one of another or of an exact method (ILS+ SA, VLSN).
- Treat algorithmic procedures (heuristics and exact) as black boxes and serialize.
- Let metaheuristics cooperate (evolutionary + tabu search).
- Use different metaheuristics to solve the same solution space or a partitioned solution space.

Configuration Problem
Algorithms must be configured and tuned and the best selected.
This has to be done anew every time because constraints and their density (problem instance) are specific of the institution.

Post Enrollment Timetabling

Definition
Find an assignment of lectures to time slots and rooms which is Feasible
rooms are only used by one lecture at a time, each lecture is assigned to a suitable room, no student has to attend more than one lecture at once, lectures are assigned only time slots where they are available; precedences are satisfied; and Good
no more than two lectures in a row for a student, unpopular time slots avoided (last in a day), students do not have one single lecture in a day.
Graph models

We define:

- **precedence digraph** \( D = (V, A) \): directed graph having a vertex for each lecture in the vertex set \( V \) and an arc from \( u \) to \( v \), \( u, v \in V \), if the corresponding lecture \( u \) must be scheduled before \( v \).

- **Transitive closure of** \( D \): \( D' = (V, A') \)

- **conflict graph** \( G = (V, E) \): edges connecting pairs of lectures if:
  - the two lectures share students;
  - the two lectures can only be scheduled in a room that is the same for both;
  - there is an arc between the lectures in the digraph \( D' \).

A look at the instances

These are large scale instances.

Solution Representation

A. Room assignment left to matching algorithm:

Array of Lectures and Time-slots and/or Collection of sets Lectures, one for each Time-slot

B. Room assignment included

Assignment Matrix

<table>
<thead>
<tr>
<th>Rooms</th>
<th>Time-slots</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>( L_4 ) ( \ldots ) ( L_{10} ) ( \ldots ) ( L_{14} ) ( \ldots )</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>( L_1 ) ( \ldots ) ( L_{11} ) ( \ldots )</td>
</tr>
<tr>
<td>( R_3 )</td>
<td>( L_2 ) ( \ldots ) ( L_{12} ) ( \ldots ) ( -1 ) ( \ldots )</td>
</tr>
</tbody>
</table>
| \( \ldots \) | \( \ldots \) \( \ldots \) | | | |}
| \( R_r \) | \( L_3 \) \( \ldots \) \( L_{13} \) | \( L_{16} \) | \( -1 \) | | |
Construction Heuristic

most-constrained lecture on least constraining time slot

Step 1. Initialize the set \( \hat{L} \) of all unscheduled lectures with \( \hat{L} = L \).
Step 2. Choose a lecture \( L_i \in \hat{L} \) according to a heuristic rule.
Step 3. Let \( \tilde{X} \) be the set of all positions for \( L_i \) in the assignment matrix with minimal violations of the hard constraints \( H \).
Step 4. Let \( \hat{X} \subseteq \tilde{X} \) be the subset of positions of \( \tilde{X} \) with minimal violations of the soft constraints \( \Sigma \).
Step 5. Choose an assignment for \( L_i \) in \( \hat{X} \) according to a heuristic rule. Update information.
Step 6. Remove \( L_i \) from \( \hat{L} \), and go to step 2 until \( \hat{L} \) is not empty.

Local Search Algorithms

Neighborhood Operators:

A. Room assignment left to matching algorithm

The problem becomes a bounded graph coloring

- Apply well known algorithms for GCP with few adaptations

Ex:

- complete assignment representation: TabuCol with one exchange
- partial assignment representation: PartialCol with i-swaps

See [Blöchliger and N. Zufferey, 2008] for a description

Example of stochastic local search for Hard Constraints, representation A.

initialize data (fast updates, dont look bit, etc.)
while (hcv!=0 && stillTime && idle iterations < PARAMETER)
shuffle the time slots for each lecture \( L \) causing a conflict
for each time slot \( T \)
if not dont look bit
if lecture is available in \( T \)
if lectures in \( T \) < number of rooms
try to insert \( L \) in \( T \)
compute delta
if \( \delta < 0 \) || with a PARAMETER probability if \( \delta = 0 \)
if there exists a feasible matching room-lectures
implement change
update data
if \( \delta = 0 \) idle iterations++ else idle iterations=0;
break
for all lectures in time slot
try to swap time slots
compute delta
if \( \delta < 0 \) || with a PARAMETER probability if \( \delta = 0 \)
implement change
update data
if \( \delta = 0 \) idle iterations++ else idle iterations=0;
break

- \( N_1 \): One Exchange
- \( N_2 \): Swap
- \( N_3 \): Insert + Rematch
- \( N_4 \): Kempe Chain Interchange
- \( N_5 \): Swap + Rematch
### In Practice

A timetabling system consists of:

- Information management (database maintenance)
- Solver (written in a fast language, i.e., C, C++)
- Input and Output management (various interfaces to handle input and output)
- Interactivity: Declaration of constraints (professors’ preferences may be inserted directly through a web interface and stored in the information system of the University)

See examples [http://www.easystaff.it](http://www.easystaff.it)  
[http://www.eventmap-uk.com](http://www.eventmap-uk.com)

---

The timetabling process

1. Collect data from the information system

2. Execute a few runs of the Solver starting from different solutions selecting the timetable of minimal cost. The whole computation time should not be longer than say one night. This becomes a “draft” timetable.

3. The draft is shown to the professors who can require adjustments. The adjustments are obtained by defining new constraints to pass to the Solver.

4. Post-optimization of the “draft” timetable using the new constraints

5. The timetable can be further modified manually by using the Solver to validate the new timetables.