The Timetabling Activity

Assignment of events to a limited number of time periods and locations subject to constraints.

Two categories of constraints:

Hard constraints $H = \{H_1, \ldots, H_n\}$: must be strictly satisfied, no violation is allowed

Soft constraints $\Sigma = \{S_1, \ldots, S_m\}$: their violation should be minimized (determine quality)

Each institution may have some unique combination of hard constraints and take different views on what constitute the quality of a timetable.
Types of Timetabling

▶ Educational Timetabling
  ▶ Class timetabling
  ▶ Exam timetabling
  ▶ Course timetabling

▶ Employee Timetabling
  ▶ Crew scheduling
  ▶ Crew rostering

▶ Transport Timetabling,
▶ Sports Timetabling,
▶ Communication Timetabling

Educational timetabling process

<table>
<thead>
<tr>
<th>Phase:</th>
<th>Planning</th>
<th>Scheduling</th>
<th>Dispatching</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizon:</td>
<td>Long Term</td>
<td>Timetable Period</td>
<td>Day of Operation</td>
</tr>
<tr>
<td>Objective:</td>
<td>Service Level</td>
<td>Feasibility</td>
<td>Get it Done</td>
</tr>
<tr>
<td>Steps:</td>
<td>Curricula</td>
<td>Weekly Timetabling</td>
<td>Repair, find rooms</td>
</tr>
<tr>
<td></td>
<td>Manpower, Equipment</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We will concentrate on simple models that admit IP formulations or graph and network algorithms. These simple problems might:

▶ occur at various stages

▶ be instructive to derive heuristics for more complex cases

Outline

1. Introduction
2. Educational Timetabling
   School Timetabling
   Course Timetabling
3. A Solution Example
4. Timetabling in Practice
School Timetabling

[aka, teacher-class model]
The daily or weekly scheduling for all the classes of a high school, avoiding teachers meeting two classes in the same time, and vice versa.

**Input:**
- a set of classes $C = \{C_1, \ldots, C_m\}$
  - A class is a set of students who follow exactly the same program. Each class has a dedicated room.
- a set of teachers $P = \{P_1, \ldots, P_n\}$
- a requirement matrix $R_{m \times n}$ where $R_{ij}$ is the number of lectures given by teacher $P_j$ to class $C_i$.
- all lectures have the same duration (say one period)
- a set of time slots $T = \{T_1, \ldots, T_p\}$ (the available periods in a day).

**Output:** An assignment of lectures to time slots such that no teacher or class is involved in more than one lecture at a time.

**IP formulation:**
Binary variables: assignment of teacher $P_j$ to class $C_i$ in $T_k$

$x_{ijk} = \{0,1\} \quad \forall i = 1,\ldots, m; \quad j = 1,\ldots, n; \quad k = 1,\ldots, p$

**Constraints:**

$\sum_{k=1}^{p} x_{ijk} = R_{ij} \quad \forall i = 1,\ldots, m; \quad j = 1,\ldots, n$

$\sum_{j=1}^{n} x_{ijk} \leq 1 \quad \forall i = 1,\ldots, m; \quad k = 1,\ldots, p$

$\sum_{i=1}^{m} x_{ijk} \leq 1 \quad \forall j = 1,\ldots, n; \quad k = 1,\ldots, p$

**Graph model**
Bipartite multigraph $G = (C, T, R)$:
- nodes $C$ and $T$: classes and teachers
- $R_{ij}$ parallel edges
Time slots are colors \(\text{Graph-Edge Coloring problem}\)

**Theorem:** [König] There exists a solution to (1) iff:

$\sum_{i=1}^{m} R_{ij} \leq p \quad \forall j = 1,\ldots, n$

$\sum_{i=1}^{m} R_{ij} \leq p \quad \forall i = 1,\ldots, m$

**Extension**
From daily to weekly schedule (timeslots represent days)
- $a_i$ max number of lectures for a class in a day
- $b_j$ max number of lectures for a teacher in a day

**IP formulation:**
Variables: number of lectures to a class in a day

$\sum_{i=1}^{m} x_{ijk} \leq b_j \quad \forall j = 1,\ldots, n; \quad k = 1,\ldots, p$

$x_{ijk} \in \mathbb{N} \quad \forall i = 1,\ldots, m; \quad j = 1,\ldots, n; \quad k = 1,\ldots, p$

**Constraints:**

$\sum_{k=1}^{p} x_{ijk} = R_{ij} \quad \forall i = 1,\ldots, m; \quad j = 1,\ldots, n$

$\sum_{j=1}^{n} x_{ijk} \leq a_i \quad \forall i = 1,\ldots, m; \quad k = 1,\ldots, p$
Graph model

Edge coloring model still valid but with

- no more than $a_i$ edges adjacent to $C_i$ have same colors and
- and more than $b_j$ edges adjacent to $T_j$ have same colors

**Theorem: [König]** There exists a solution to (2) iff:

$$
\sum_{i=1}^{m} R_{ij} \leq b_j p \quad \forall j = 1, \ldots, n
$$

$$
\sum_{i=1}^{n} R_{ij} \leq a_i p \quad \forall i = 1, \ldots, m
$$

Further complications:

- Simultaneous lectures (e.g., gymnastic)
- Subject issues (more teachers for a subject and more subject for a teacher)
- Room issues (use of special rooms)

Further constraints that may arise:

- Preassignments
- Unavailabilities
  (can be expressed as preassignments with dummy class or teachers)

They make the problem **NP-complete**.

A recurrent sub-problem in Timetabling is Matching

**Input:** A (weighted) bipartite graph $G = (V, E)$ with bipartition $\{A, B\}$.

**Task:** Find the largest size set of edges $M \in E$ such that each vertex in $V$ is incident to at most one edge of $M$.

Efficient algorithms for constructing matchings are based on augmenting paths in graphs. An implementation is available at: http://www.cs.sunysb.edu/~algorith/implement/bipm/implement.shtml

**Theorem [Hall, 1935]:** $G$ contains a matching of $A$ if and only if $|N(U)| \geq |U|$ for all $U \subseteq A$. 

Bipartite matchings can still help in developing heuristics, for example, for solving $x_{ijk}$ keeping any index fixed.
So far feasibility problem.

Preferences (soft constraints) may be introduced
▶ Desirability of assignment $p_j$ to class $c_i$ in $t_k$

$$
\min_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{p} d_{ijk} x_{ijk}
$$

▶ Organizational costs: having a teacher available for possible temporary teaching posts

▶ Specific day off for a teacher

Introducing soft constraints the problem becomes a multiobjective problem.

Possible ways of dealing with multiple objectives:
▶ weighted sum
▶ lexicographic order
▶ minimize maximal cost
▶ distance from optimal or nadir point
▶ Pareto-frontier
▶ ...

Heuristic Methods

Construction heuristic
Based on principles:
▶ most-constrained lecture on first (earliest) feasible timeslot
▶ most-constrained lecture on least constraining timeslot

Enhancements:
▶ limited backtracking
▶ local search optimization step after each assignment

More later

Local Search Methods and Metaheuristics

High level strategy:
▶ Single stage (hard and soft constraints minimized simultaneously)
▶ Two stages (feasibility first and quality second)

Dealing with feasibility issue:
▶ partial assignment: do not permit violations of $H$ but allow some lectures to remain unscheduled
▶ complete assignment: schedule all the lectures and seek to minimize $H$ violations

More later
University Course Timetabling

The weekly scheduling of the lectures 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 $C = \{L_1, \ldots, L_l\}$.
- 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.

IP formulation

$m_t$ rooms $\Rightarrow$ maximum number of lectures in time slot $t$

Variables

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

Number of lectures per course

$\sum_{i=1}^{p} x_{it} = l_i \quad \forall i = 1, \ldots, n$

Number of lectures per time slot

$\sum_{i=1}^{n} x_{it} \leq m_t \quad \forall t = 1, \ldots, p$

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?

Number of lectures per time slot (students’ perspective)

$\sum_{C_i \in S_j} x_{it} \leq 1 \quad \forall i = 1, \ldots, n; \quad t = 1, \ldots, p$

If some preferences are added:

$max \quad \sum_{i=1}^{p} \sum_{i=1}^{n} d_{it} x_{it}$

Corresponds to a bounded coloring.
It can be solved up for 70 lectures, 25 courses and 40 curricula. [de Werra, 1985]
Further complications:
- Teachers that teach more than one course (treated similarly to students’ enrollment)
- A set of rooms $\mathcal{R} = \{R_1, \ldots, R_n\}$ with suitability and availability constraints (this can be modeled as Hypergraph Coloring!)

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

IP formulation to include room eligibility [Lach and Lübbecke, 2008]

Decomposition of the problem in two stages:
1. assign courses to timeslots
2. match courses with rooms within each timeslot

In stage 1
Let $R(C_i) \subseteq \mathcal{R}$ be the rooms eligible for course $C_i$
Let $G_{conf} = (V_{conf}, E_{conf})$ be the conflict graph (vertices are pairs $(C_i, T_t)$)

Variables: course $C_i$ assigned to time slot $T_t$

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

So far feasibility.
Preferences (soft constraints) may be introduced
- Compactness or distribution
- Minimum working days
- Room stability
- Student min max load per day
- Travel distance
- Room suitability
- Double lectures
- Professors’ preferences for time slots

For most of these different way to model them exist.
Exam 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>

Solution 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 are specific of the institution.
Appropriate techniques exist to aid in the experimental assessment of algorithms.
1. Introduction

2. Educational Timetabling
   - School Timetabling
   - Course Timetabling

3. A Solution Example

4. Timetabling in Practice

A Solution Example on Course Timetabling

Course Timetabling Problem

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;

Hard Constraints

- 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.

Soft Constraints

A look at the instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>events</th>
<th>students</th>
<th>rooms</th>
<th>events/students</th>
<th>students/event</th>
<th>rooms/事件</th>
<th>degree</th>
<th>av. slot_event</th>
</tr>
</thead>
<tbody>
<tr>
<td>comp-2007-2-10.tnm</td>
<td>400</td>
<td>500</td>
<td>10</td>
<td>20.984</td>
<td>26.23</td>
<td>1.2025</td>
<td>0.3843</td>
<td>25.4675</td>
</tr>
<tr>
<td>comp-2007-2-3.tnm</td>
<td>400</td>
<td>500</td>
<td>10</td>
<td>21.07</td>
<td>26.27</td>
<td>1.6387</td>
<td>0.3749</td>
<td>25.66</td>
</tr>
<tr>
<td>comp-2007-2-1.tnm</td>
<td>400</td>
<td>500</td>
<td>10</td>
<td>21.02</td>
<td>25.75</td>
<td>4.6775</td>
<td>0.9418</td>
<td>25.3425</td>
</tr>
<tr>
<td>comp-2007-2-14.tnm</td>
<td>400</td>
<td>300</td>
<td>20</td>
<td>20.8667</td>
<td>15.6432</td>
<td>7.5625</td>
<td>0.2025</td>
<td>25.44</td>
</tr>
<tr>
<td>comp-2007-2-5.tnm</td>
<td>400</td>
<td>300</td>
<td>20</td>
<td>20.9167</td>
<td>15.6872</td>
<td>6.805</td>
<td>0.3081</td>
<td>25.4725</td>
</tr>
<tr>
<td>comp-2007-2-6.tnm</td>
<td>400</td>
<td>300</td>
<td>23</td>
<td>20.7067</td>
<td>15.5432</td>
<td>5.653</td>
<td>0.3020</td>
<td>25.3925</td>
</tr>
<tr>
<td>comp-2007-2-12.tnm</td>
<td>200</td>
<td>1000</td>
<td>10</td>
<td>13.6097</td>
<td>60.035</td>
<td>3.955</td>
<td>0.5842</td>
<td>25.67</td>
</tr>
<tr>
<td>comp-2007-2-15.tnm</td>
<td>200</td>
<td>500</td>
<td>10</td>
<td>13.054</td>
<td>32.632</td>
<td>2.23</td>
<td>0.5365</td>
<td>17.275</td>
</tr>
<tr>
<td>comp-2007-2-7.tnm</td>
<td>200</td>
<td>500</td>
<td>21</td>
<td>13.46</td>
<td>32.665</td>
<td>1.375</td>
<td>0.5335</td>
<td>17.86</td>
</tr>
<tr>
<td>comp-2007-2-8.tnm</td>
<td>200</td>
<td>1000</td>
<td>10</td>
<td>13.9947</td>
<td>60.946</td>
<td>6.4</td>
<td>0.1569</td>
<td>25.665</td>
</tr>
<tr>
<td>comp-2007-2-11.tnm</td>
<td>200</td>
<td>500</td>
<td>10</td>
<td>13.972</td>
<td>34.54</td>
<td>1.915</td>
<td>0.5175</td>
<td>17.17</td>
</tr>
<tr>
<td>comp-2007-2-1.tnm</td>
<td>200</td>
<td>1000</td>
<td>10</td>
<td>13.908</td>
<td>68.04</td>
<td>3.375</td>
<td>0.5056</td>
<td>25.32</td>
</tr>
<tr>
<td>comp-2007-2-2.tnm</td>
<td>200</td>
<td>1000</td>
<td>10</td>
<td>13.982</td>
<td>68.012</td>
<td>5.045</td>
<td>0.4740</td>
<td>25.56</td>
</tr>
<tr>
<td>comp-2007-2-16.tnm</td>
<td>200</td>
<td>500</td>
<td>10</td>
<td>13.538</td>
<td>34.095</td>
<td>1.74</td>
<td>0.4550</td>
<td>17.57</td>
</tr>
</tbody>
</table>

These are large scale instances.

A look at the evaluation of a timetable can help in understanding the solution strategy

High level solution strategy:

- Single phase strategy (not well suited here due to soft constraints)
- Two phase strategy: Feasibility first, quality second

Searching a feasible solution:

- Suitability of rooms complicate the use of IP and CP.
- Heuristics:
  1. Complete assignment of lectures
  2. Partial assignment of lectures

- Room assignment:
  A. Left to matching algorithm
  B. Carried out heuristically
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

| Rooms | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 | T11 | T12 | T13 | T14 | T15 | T16 | T17 | T18 | T19 | T20 | T21 | T22 | T23 | T24 | T25 | T26 | T27 |
|-------|----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| R1    | −1 | L4 | ⋯  | L10 | ⋯  | L14 | ⋯  | −1 |
| R2    | L1 | L5 | ⋯  | L11 | ⋯  | L15 | ⋯  | −1 |
| R3    | L2 | L6 | ⋯  | L12 | ⋯  | −1 | ⋯  | −1 |
| ⋮     | ⋮  | ⋮  | ⋮  | ⋮  | ⋮  | ⋮  | ⋮  | ⋮  |
| Rr    | L3 | L7 | ⋯  | L13 | ⋯  | L16 | ⋯  | −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 $X$ be the set of all positions for $L_i$ in the assignment matrix with minimal violations of the hard constraints $H$.
Step 4. Let $X \subseteq \hat{X}$ be the subset of positions of $X$ with minimal violations of the soft constraints $\Sigma$.
Step 5. Choose an assignment for $L_i$ in $X$ 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:
  1. complete assignment representation: TabuCol with one exchange
  2. partial assignment representation: PartialCol with i-swaps

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

B. Room assignment included

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>T2</td>
<td>T3</td>
</tr>
<tr>
<td>R1</td>
<td>187</td>
<td>176</td>
</tr>
<tr>
<td>R2</td>
<td>360</td>
<td>306</td>
</tr>
<tr>
<td>R3</td>
<td>263</td>
<td>71</td>
</tr>
<tr>
<td>R4</td>
<td>181</td>
<td>150</td>
</tr>
<tr>
<td>R5</td>
<td>324</td>
<td>291</td>
</tr>
<tr>
<td>R6</td>
<td>322</td>
<td>229</td>
</tr>
<tr>
<td>R7</td>
<td>238</td>
<td>110</td>
</tr>
<tr>
<td>R8</td>
<td>288</td>
<td>32</td>
</tr>
<tr>
<td>R9</td>
<td>396</td>
<td>144</td>
</tr>
<tr>
<td>R10</td>
<td>382</td>
<td>1</td>
</tr>
</tbody>
</table>

➤ N1: One Exchange
➤ N2: Swap
➤ N3: Period Swap
➤ N4: Kempe Chain Interchange
Example of stochastic local search for case 1. with 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
                            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

Algorithm Flowchart

In Practice

A timetabling system consists of:

- Information Management
- 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.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.

Current Research Directions

1. Attempt to formulate standard timetabling problems with super sets of constraints where portable programs can be developed and compared

2. Development of general frameworks that leave the user the final instantiation of the program

3. Methodology for choosing automatically and intelligently the appropriate algorithm for the problem at hand (hyper-heuristics case-based reasoning systems and racing for algorithm configuration).

4. Robust timetabling

For latest developments see results of International Timetabling Competition 2007: [http://www.cs.qub.ac.uk/itc2007/]