DM87
SCHEDULING,
TIMETABLING AND ROUTING

Lecture 14

Educational Timetabling

Marco Chiarandini

Outline

Outline

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

Phase:	Planning	Scheduling	Dispatching
Horizon:	Long Term	Timetable Period	Day of Operation
Objective:	Service Level	Feasibility	Get it Done
Steps:	Curricula Manpower, Equipment	Weekly Timetabling	Repair, find rooms

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

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

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

Constraints:

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

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

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

Graph model

Bipartite multigraph $G = (\mathcal{C}, \mathcal{T}, \mathcal{R})$:

- nodes $\mathcal{C}$ and $\mathcal{T}$: classes and teachers
- R_{ij} parallel edges

Time slots are colors ➜ Graph-Edge Coloring problem

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

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

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

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

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

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

Constraints:

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

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

Graph model

Edge coloring model still valid but with

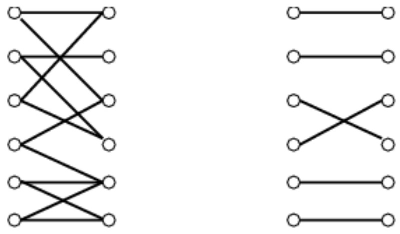
- no more than a_i edges adjacent to C_i have same colors and
- and more than b_j edges adjacent to T_j have same colors

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

$$\sum_{i=1}^{m} R_{ij} \le b_j p \quad \forall j = 1, \dots, n$$
$$\sum_{i=1}^{n} R_{ij} \le a_i p \quad \forall i = 1, \dots, m$$

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 $|N(U)| \ge |U|$ for all $U \subseteq A$.

- The edge coloring problem in the multigraph is solvable in polynomial time by solving a sequence of network flows problems p.
  Possible approach: solve the weekly timetable first and then the daily timetable

Further constraints that may arise:

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

They make the problem NP-complete.

- Bipartite matchings can still help in developing heuristics, for example, for solving x_{ijk} keeping any index fixed.

Further complications:

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

So far feasibility problem.

Preferences (soft constraints) may be introduced

- Desirability of assignment p_j to class c_i in t_k

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

- 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 $\mathcal{C} = \{C_1, \dots, C_n\}$ each consisting of a set of lectures $\mathcal{C}_i = \{L_{i1}, \dots, L_{il_i}\}$. Alternatively, A set of lectures $\mathcal{L} = \{L_1, \dots, L_l\}$).
- A set of curricula $\mathcal{S} = \{S_1, \dots, S_r\}$ that are groups of courses with common students (curriculum based model). Alternatively, A set of enrollments $\mathcal{S} = \{S_1, \dots, S_s\}$ that are groups of courses that a student wants to attend (Post enrollment model).
- a set of time slots $\mathcal{T} = \{T_1, \dots, T_p\}$ (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

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

Number of lectures per course

$$\sum_{t=1}^{p} x_{it} = l_i \qquad \forall i = 1, \dots, n$$

Number of lectures per time slot

$$\sum_{i=1}^{n} x_{it} \le m_t \qquad \forall t = 1, \dots, p$$

Number of lectures per time slot (students' perspective)

$$\sum_{C_i \in S_j}^{n} x_{it} \le 1 \qquad \forall i = 1, \dots, n;\ t = 1, \dots, p$$

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]

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 ➜ Graph-Vertex Coloring problem ➜ NP-complete (exact solvers max 100 vertices)

Typical further constraints:

- Unavailabilities
- Preassignments

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

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

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

Edge constraints
(forbids that C_{i_1} is assigned to T_{t_1} and C_{i_2} to T_{t_2} simultaneously)

$$x_{i_1,t_1} + x_{j_2,t_2} \leq 1 \qquad \forall\, ((i_1, t_1), (i_2, t_2)) \in E_{conf}$$

Hall's constraints
(guarantee that in stage 1 we find only solutions that are feasibile for stage 2)

$$\sum_{C_i \in U}^{n} x_{it} \leq |N(U)| \qquad \forall\, U \in, t \in \mathcal{T}$$

If some preferences are added:

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

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

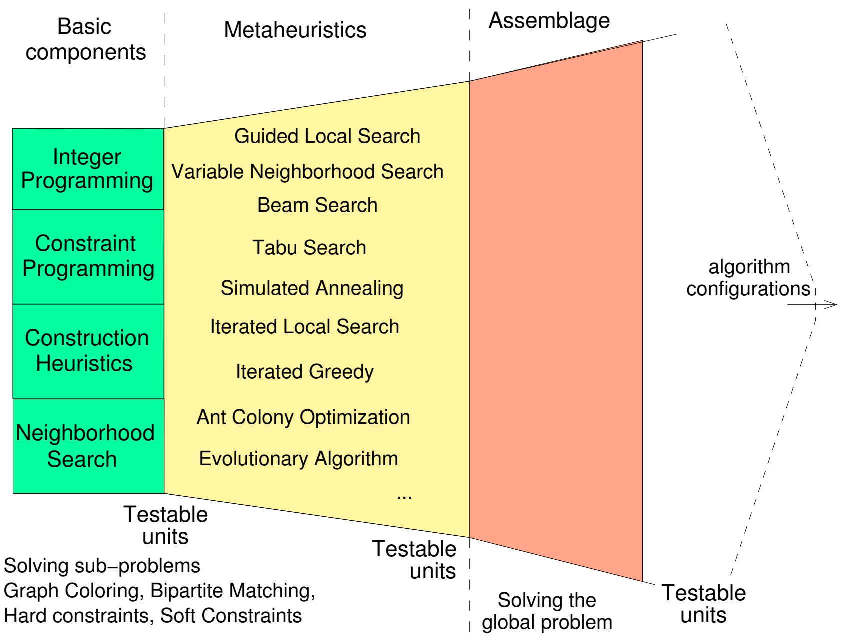
By substituting lecture with exam we have the same problem!
However:

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

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.

Outline

A Solution Example on Course Timetabling

Course Timetabling Problem

Find an assignment of lectures to time slots and rooms which is

Feasible

$\left.\begin{array}{l}\text{rooms are only used by one lecture at a time,}\\ \text{each lecture is assigned to a suitable room,}\\ \text{no student has to attend more than one lecture at once,}\\ \text{lectures are assigned only time slots where they are available;}\end{array}\right\}$ Hard Constraints

and Good

$\left.\begin{array}{l}\text{no more than two lectures in a row for a student,}\\ \text{unpopular time slots avoided (last in a day),}\\ \text{students do not have one single lecture in a day.}\end{array}\right\}$ Soft Constraints

A look at the instances

instance	events	students	rooms	events/students	students/event	rooms/event	degree	av_slot_event
comp-2007-2-10.tim	400	500	10	20.984	26.23	3.2025	0.3843	25.4675
comp-2007-2-2.tim	400	500	10	21.03	26.2875	3.9475	0.3744	25.69
comp-2007-2-1.tim	400	500	10	21.02	26.275	4.0775	0.3418	25.3425
comp-2007-2-9.tim	400	500	10	21.428	26.785	2.9125	0.3409	25.4175
comp-2007-2-13.tim	400	300	20	21.1933	15.895	8.6825	0.3236	25.7525
comp-2007-2-14.tim	400	300	20	20.8567	15.6425	7.5625	0.3209	25.44
comp-2007-2-5.tim	400	300	20	20.9167	15.6875	6.805	0.3081	25.425
comp-2007-2-6.tim	400	300	20	20.7267	15.545	5.065	0.3029	25.3925
comp-2007-2-12.tim	200	1000	10	13.607	68.035	3.355	0.5842	25.67
comp-2007-2-15.tim	200	500	10	13.054	32.635	2.23	0.5365	17.375
comp-2007-2-7.tim	200	500	20	13.466	33.665	1.575	0.5336	17.86
comp-2007-2-4.tim	200	1000	20	13.396	66.98	6.4	0.5199	25.665
comp-2007-2-8.tim	200	500	20	13.832	34.58	1.915	0.5175	17.17
comp-2007-2-11.tim	200	1000	10	13.608	68.04	3.375	0.5006	25.32
comp-2007-2-3.tim	200	1000	20	13.383	66.915	5.045	0.4748	25.535
comp-2007-2-16.tim	200	500	10	13.638	34.095	1.74	0.4558	17.57

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

Time-slots

$$
\text{Rooms}\quad
\begin{array}{c|ccccccccc}
 & T_1 & T_2 & & T_i & & T_j & & T_{45} \\
\hline
R_1 & -1 & L_4 & \cdots & \mathbf{L_{10}} & \cdots & L_{14} & \cdots & -1 \\
R_2 & L_1 & L_5 & \cdots & L_{11} & \cdots & \mathbf{L_{15}} & \cdots & -1 \\
R_3 & L_2 & L_6 & \cdots & \mathbf{L_{12}} & \cdots & -1 & \cdots & -1 \\
\vdots & \vdots & & \vdots & & \vdots & & \vdots & \\
R_r & L_3 & L_7 & \cdots & L_{13} & & L_{16} & \cdots & -1
\end{array}
$$

Construction Heuristic

most-constrained lecture on least constraining time slot

Step 1. Initialize the set $\widehat{L}$ of all unscheduled lectures with $\widehat{L} = L$.
Step 2. Choose a lecture $L_i \in \widehat{L}$ according to a *heuristic rule*.
Step 3. Let $\widehat{X}$ be the set of all positions for L_i in the assignment matrix with minimal violations of the hard constraints H.
Step 4. Let $\bar{X} \subseteq \widehat{X}$ be the subset of positions of $\widehat{X}$ with minimal violations of the soft constraints Σ.
Step 5. Choose an assignment for L_i in $\bar{X}$ according to a *heuristic rule*. Update information.
Step 6. Remove L_i from $\widehat{L}$, and go to step 2 until $\widehat{L}$ 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

	Monday								
	T1	T2	T3	T4	T5	T6	T7	T8	T9
R1	187	239	378	66	380	53	208	279	
R2	360	345	2	153		354	91	61	319
R3	263	71	186	67	222	288	99	24	
R4	181	160		90	82			193	
R5	324	291	309	339	267	283			
R6	322	225	352	28	168	72	49	69	12
R7	228	31	107	371	30	355	46	227	246
R8	256	32	147	270	289	130	48	282	
R9	396	144	173	78	25	183	387	337	240
R10	382	1	56	362	45	247	392	85	389

	Tuesday								
	T10	T11	T12	T13	T14	T15	T16	T17	T18
R1	300	350	211	375	254	366	369	223	163
R2	349	278	86	204	316	220	323	176	
R3	237		232	253	117		195	203	102
R4	206	156	152		341	179	171	226	
R5	269	170	299	311	34		65	216	
R6	92	38	373	390	164	135	121	268	115
R7	271	182	313	224	128		89	258	356
R8	0	116	251	307	44	260	79	296	
R9	132	328	212	370	308	336	244	126	14
R10	384	17	394	200		294	273	391	180

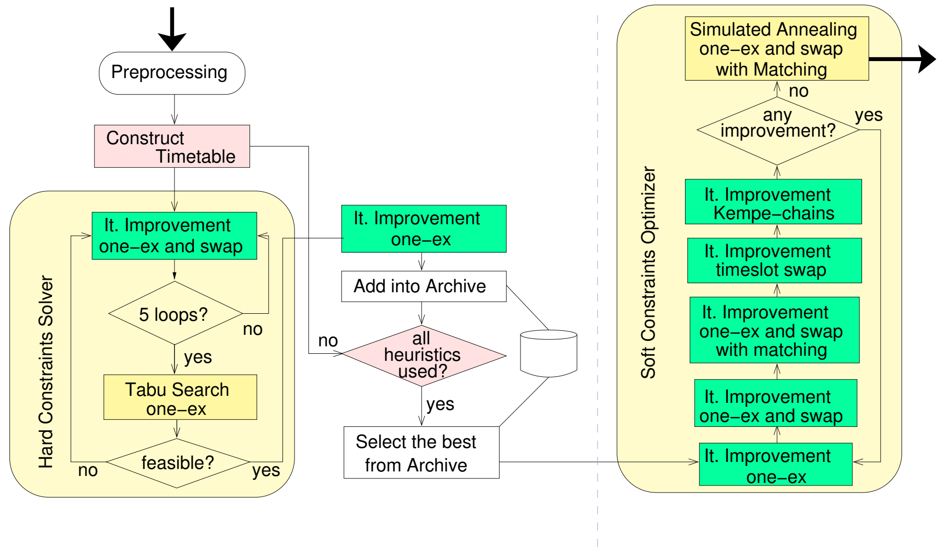
	Wednesday								
	T19	T20	T21	T22	T23	T24	T25	T26	T27
R1	298		118	368	234	97	329	274	58
R2	314	7	108		50	312	235	330	
R3	207	287	290	146	286	358	303	277	
R4	4	348	127			365	213	80	
R5	275	199	26		27	327	33	39	285
R6	75	87	140	165	104	137	133	385	346
R7	343	280	35	109	306	43	83	11	154
R8	242	150	81	353	158	293	338	218	161
R9	231	51	342	136	93	129	266	393	155
R10	42	157	388	397	331	131	363	383	

- N_1: One Exchange
- N_2: Swap
- N_3: Period Swap
- N_4: 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
               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
```

Algorithm Flowchart



Outline

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/`