DM204 – Autumn 2013

Scheduling, Timetabling and Routing

Lecture 8
Course Timetabling

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Outline

1. Introduction

2. School Timetabling

3. Course Timetabling
   - Formalization and Modelling
   - An Example
   - Timetabling in Practice

Educational timetabling process

Phase:	Planning	Scheduling	Dispatching
Horizon:	Long Term	Timetable Period	Day of Operation
Objective:	Service Level	Feasibility	Get it Done
Steps:	Manpower, Curriculum, Equipment	Quarterly Timetabling, Project assignment, student sectioning	Repair

Timetabling

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.

Outline

1. Introduction

2. School Timetabling

3. Course Timetabling
   - Formalization and Modelling
   - An Example
   - 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.

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 P_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:

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

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

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

Graph model

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

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

Time slots are colors ➜ Graph-Edge Coloring problem

Theorem: [König] There exists a solution that uses p colors iff:

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

Deciding R_{ij}

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

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

Constraints:

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

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

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

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 that uses p slots iff:

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

Hence, we can find the minimum number of periods needed
or, if p is given, find a formulation of the problem that admits a solution balancing the work load

$\rightsquigarrow$ The edge coloring problem in the multigraph is solvable in polynomial time by solving a sequence of p network flows problems. [De Werra, 1985]

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 if any teacher is unavailable during more than 2 periods.
(Reduction from 3-SAT, [Even, Itai, Shamir, 1975])

- 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 subjects for a teacher)
- Room issues (use of special rooms)

So far feasibility problem.

Preferences (soft constraints) may be introduced

- Desirability of assigning teacher 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
- Social welfare approaches

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

Outline

Course Timetabling

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

Input:

- A set of courses $\mathcal{C} = \{C_1, \ldots, C_n\}$ each consisting of a set of lectures $\mathcal{C}_i = \{L_{i1}, \ldots, L_{il_i}\}$. Alternatively,
  A set of lectures $\mathcal{L} = \{L_1, \ldots, L_l\}$.
- A set of curricula $\mathcal{S} = \{S_1, \ldots, S_r\}$ that are groups of courses with common students (curriculum based model). Alternatively,
  A set of enrollments $\mathcal{S} = \{S_1, \ldots, S_s\}$ that are groups of courses that a student wants to attend (Post enrollment model).
- a set of time slots $\mathcal{T} = \{T_1, \ldots, T_p\}$ (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 ➜ Graph-Vertex Coloring problem ➜ NP-complete (exact solvers max 100 vertices)

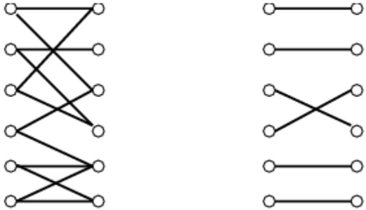
Typical further constraints:

- Unavailabilities
- Preassignments

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

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

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

IP model

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

Variables

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

Number of lectures per course

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

Number of lectures per time slot

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

Number of lectures per time slot (students' perspective)

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

If some preferences are added:

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

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

Further complications:

- Teachers that teach more than one course
  (not really a complication: treated similarly to students' enrollment)

- A set of rooms $\mathcal{R} = \{R_1, \ldots, R_n\}$
  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: no 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

IP approach

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

$R(c) \subseteq \mathcal{R}$: rooms eligible for course c
$G_{conf} = (V_{conf}, E_{conf})$: conflict graph (vertices are pairs (c, t))

$$\min \sum_{ctr} d(c,t) x_{ctr} \qquad \forall c \in \mathcal{C}$$

$$\sum_{\substack{t \in T \\ r \in R(c)}} x_{ctr} = l(c) \qquad \forall c \in \mathcal{C}$$

$$\sum_{c \in R^{-1}(r)} x_{ctr} \leq 1 \qquad \forall t \in T, r \in \mathcal{R}$$

$$\sum_{r \in R(c_1)} x_{c_1 t_1 r} + \sum_{r \in R(c_2)} x_{c_2 t_2 r} \leq 1 \qquad \forall ((c_1, t_1)(c_2, t_2)) \in E_{conf}$$

$$x_{ctr} \in \{1, 0\} \qquad \forall (c,t) \in V_{conf}, r \in \mathcal{R}$$

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

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

$$x_{ct} \in \{0,1\} \qquad c \in \mathcal{C}, t \in \mathcal{T}$$

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

$$x_{c_1,t_1} + x_{c_2,t_2} \leq 1 \qquad \forall\,((c_1,t_1),(c_2,t_2)) \in E_{conf}$$

Hall's constraints
(guarantee that in stage 1 we find only solutions that are feasible for stage 2)
$G_t = (\mathcal{C}_t \cup \mathcal{R}_t, E_t)$ bipartite graph for each t
$G = \cup_t G_t$

$$\sum_{c \in U}^{n} x_{ct} \leq |N(U)| \qquad \forall\, U \subseteq \mathcal{C}, t \in \mathcal{T}$$

If some preferences are added:

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

- Hall's constraints are exponentially many
- [Lach and Lübbecke, 2008] study the polytope of the bipartite matching and find strengthening conditions

  (polytope: convex hull of all incidence vectors defining subsets of $\mathcal{C}$ perfectly matched)
- Algorithm for generating all facets is polynomial if the number of defining $\mathcal{C}$-sets is polynomially bounded.
- Could solve the overall problem by branch and cut (separation problem is easy).
  However 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

international TIMETABLING COMPETITION

sponsored by PATAT

- Home
- Overview
- Introducing the Team
- Competition Tracks
- The Rules
- Benchmarking
- Winner
- Finalist Ordering
- Solutions
- Discussion Forum
- Download Datasets
- Papers

REGISTER NOW
Already registered? Click here to login.

Finalist Ordering

The following information details the finalists for each track in place order.

Please note that a report detailing the background to the competition can be found here. This has been submitted for consideration to INFORMS Journal on Computing.

Examination Track

Best recorded scores may be viewed here. By clicking on individual names more details relating to scores are available.

1st Place: Tomas Müller (USA)

2nd Place: Christos Gogos (Greece)

3rd Place:Mitsunori Atsuta, Koji Nonobe, and Toshihide Ibaraki (Japan)

4th Place: Geoffrey De Smet (Belgium)

5th Place: Nelishia Pillay (South Africa)

Post Enrolment based Course Timetabling

An excell spreadsheet containing all the scores can be downloaded here. This information is also available as .csv or.xml format.

1st Place: Hadrien Cambazard, Emmanuel Hebrard, Barry O'Sullivan, Alexandre Papadopoulos (Ireland) (pdf description)

2nd Place: Mitsunori Atsuta, Koji Nonobe, and Toshihide Ibaraki (Japan) (pdf description)

3rd Place: Marco Chiarandini, Chris Fawcett, Holger H Hoos (Denmark) (pdf description)

4th Place: Clemens Nothegger, Alfred Mayer, Andreas Chwatal, Gunther Raidl (Austria) (pdf description)

5th Place: Tomas Müller (USA) (pdf description)

Curriculum based Course Timetabling

An excell document containing all the scores can be found here. This information is also available as .csv or.xml format.

1st Place: Tomas Müller (USA)

2nd Place: Zhipeng Lu and Jin-Kao Hao (France)

3rd Place: Mitsunori Atsuta, Koji Nonobe, and Toshihide Ibaraki (Japan)

4th Place: Martin Josef Geiger (Germany)

5th Place: Michael Clark, Martin Henz, and Bruce Love (Singapore)

Contact Details

Dr Barry McCollum
SARC Building
School of Electronics, Electrical Engineering & Computer Science
Queen's University Belfast

Phone: +44 (0) 2890974622
Fax: +44 (0) 2890975666
Email: b.mccollum@qub.ac.uk

Course/Exam Timetabling

By substituting events with lecture or exam we have the course or exam timetabling, respectively

Differences

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

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 very 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

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;

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

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

ID	year	lecs	studs	rooms	lecs/stud	studs/lec	rooms/le(	degree	slots/lec	slots/lec	slots/lec	Prec.	Rel. Prec.
1	2007	400	500	10	21.02	26.27	4.08	0.34	16	25.34	34	40	14
2	2007	400	500	10	21.03	26.29	3.95	0.37	17	25.69	33	36	14
3	2007	200	1000	20	13.38	66.92	5.04	0.47	19	25.54	33	20	11
4	2007	200	1000	20	13.40	66.98	6.40	0.52	15	25.66	33	20	9
5	2007	400	300	20	20.92	15.69	6.80	0.31	16	25.43	34	120	43
6	2007	400	300	20	20.73	15.54	5.07	0.30	13	25.39	36	119	32
7	2007	200	500	20	13.47	33.66	1.57	0.53	9	17.86	26	20	10
8	2007	200	500	20	13.83	34.58	1.92	0.52	11	17.17	26	21	13
9	2007	400	500	10	21.43	26.79	2.91	0.34	17	25.42	34	41	18
10	2007	400	500	10	20.98	26.23	3.20	0.38	14	25.47	34	40	13
11	2007	200	1000	10	13.61	68.04	3.38	0.50	17	25.32	35	21	17
12	2007	200	1000	10	13.61	68.03	3.35	0.58	15	25.67	35	20	13
13	2007	400	300	20	21.19	15.89	8.68	0.32	17	25.75	34	116	34
14	2007	400	300	20	20.86	15.64	7.56	0.32	17	25.44	36	118	46
15	2007	200	500	10	13.05	32.63	2.23	0.54	11	17.38	24	21	13
16	2007	200	500	10	13.64	34.09	1.74	0.46	10	17.57	25	19	10

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:

  - Room eligibility complicate the use of IP and CP.
  - Solution Representation:
    Approach:
    1. Complete (infeasible) assignment of lectures
    2. Partial (feasible) assignment of lectures

    Room assignment:

    A. Left to matching algorithm
    B. Carried out heuristically (matrix representation of solutions)

Solution Representation

A. Room assignment left to matching algorithm:

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

B. Room assignment included

   Assignment Matrix

$$
\text{Rooms}\quad
\begin{array}{c|cccccccc}
\multicolumn{9}{c}{\text{Time-slots}} \\
 & T_1 & T_2 & & T_i & & T_j & & T_{45} \\
\hline
R_1 & -1 & L_4 & \cdots & \mathbf{L_{10}} & \cdots & L_{14} & \cdots & -1 \\
R_2 & L_1 & L_5 & \cdots & L_{11} & \cdots & \mathbf{L_{15}} & \cdots & -1 \\
R_3 & L_2 & L_6 & \cdots & \mathbf{L_{12}} & \cdots & -1 & \cdots & -1 \\
\vdots & \vdots & & \vdots & & \vdots & & \vdots & \\
R_r & L_3 & L_7 & \cdots & L_{13} & & L_{16} & \cdots & -1
\end{array}
$$

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

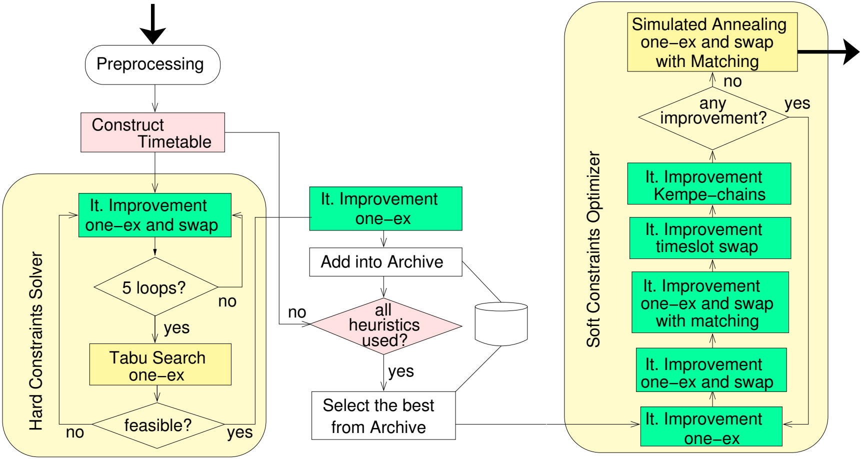
	Monday									Tuesday									Wednesday								
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24	T25	T26	T27
R1	187	239	378	66	380	53	208	279		300	350	211	375	254	366	369	223	163	298		118	368	234	97	329	274	58
R2	360	345	2	153		354	91	61	319	349	278	86	204	316	220	323	176		314	7	108		50	312	235	330	
R3	263	71	186	67	222	288	99	24		237		232	253	117		195	203	102	207	287	290	146	286	358	303	277	
R4	181	160		90	82			193		206	156	152		341	179	171	226		4	348	127			365	213	80	
R5	324	291	309	339	267	283				269	170	299	311	34		65	216		275	199	26		27	327	33	39	285
R6	322	225	352	28	168	72	49	69	12	92	38	373	390	164	135	121	268	115	75	87	140	165	104	137	133	385	346
R7	228	31	107	371	30	355	46	227	246	271	182	313	224	128		89	258	356	343	280	35	109	306	43	83	11	154
R8	256	32	147	270	289	130	48	282		0	116	251	307	44	260	79	296		242	150	81	353	158	293	338	218	161
R9	396	144	173	78	25	183	387	337	240	132	328	212	370	308	336	244	126	14	231	51	342	136	93	129	266	393	155
R10	382	1	56	362	45	247	392	85	389	384	17	394	200		294	273	391	180	42	157	388	397	331	131	363	383	

- N_1: One Exchange
- N_2: Swap
- N_5: Insert + Rematch
- N_3: Period Swap
- N_4: Kempe Chain Interchange
- N_6: Swap + Rematch

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
```

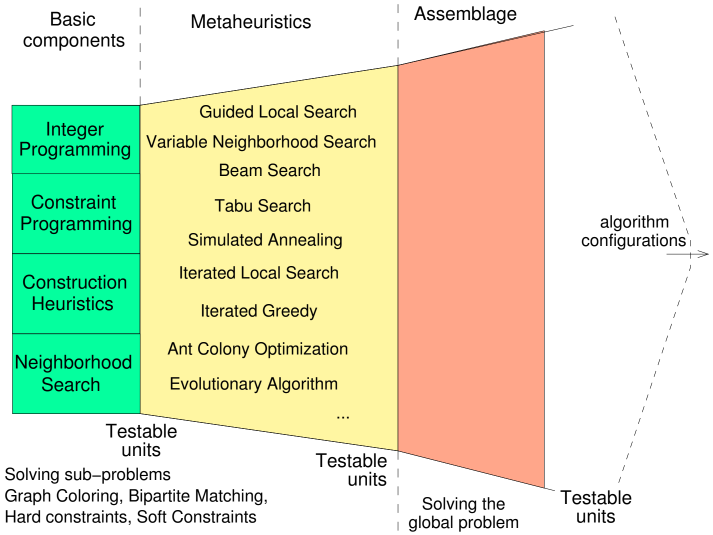
Algorithm Flowchart



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 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


Basic components
Metaheuristics
Assemblage
Integer Programming
Constraint Programming
Construction Heuristics
Neighborhood Search
Guided Local Search
Variable Neighborhood Search
Beam Search
Tabu Search
Simulated Annealing
Iterated Local Search
Iterated Greedy
Ant Colony Optimization
Evolutionary Algorithm
...
algorithm configurations
Testable units
Testable units
Testable units
Solving sub−problems
Graph Coloring, Bipartite Matching,
Hard constraints, Soft Constraints
Solving the global problem

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.

Appropriate techniques exist to aid in the experimental assessment of algorithms. Example: F-race [Birattari et al. 2002]
(see: `http://www.imada.sdu.dk/~marco/exp/` for a full list of references)

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.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.