DM811
Heuristics for Combinatorial Optimization

Examples & Exercises

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Course Overview

- ✔ Combinatorial Optimization, Methods and Models
- ✔ CH and LS: overview
- ✔ Working Environment and Solver Systems
- ✔ Methods for the Analysis of Experimental Results
- ✔ Construction Heuristics
- ✔ Local Search: Components, Basic Algorithms
- ✔ Local Search: Neighborhoods and Search Landscape
- ✔ Efficient Local Search: Incremental Updates and Neighborhood Pruning
- ✔ Stochastic Local Search & Metaheuristics
- ✖ Configuration Tools: F-race
- ✔ Very Large Scale Neighborhoods

Examples: GCP, CSP, TSP, SAT, MaxIndSet, SMTWP, Steiner Tree, Unrelated Parallel Machines, p-median, set covering, QAP, ...

Places in [B4] to read about metaheuristic approaches to the problems in some of the next slides:

SAT

GSAT + Tabu	pages 267-272
WalkSAT+Tabu	pages 274-276
Novelty	pages 276-284
Guided Local Search	pages 285-292

CSP

Tabu search	pag. 305

MAX-SAT

Guided local search	pages 324-329
Tabu search	pages 329-331
ILS	pages 331-335

TSP

ILS	pages 384-399
EA	pages 399-405

Linear Ordering Problem

ILS, EA	[A7]

QAP

Reactive TS	pages 482-484
Population based ILS	pages 484-486

Outline

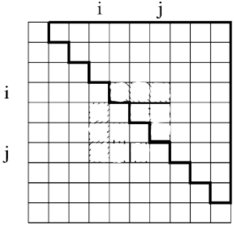
Linear Ordering Problem

Input: an $n \times n$ matrix C

Task: Find a permutation π of the column and row indices $\{1, \ldots, n\}$ such that the value

$$f(\pi) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{\pi_i \pi_j}$$

is maximized. In other terms, find a permutation of the columns and rows of C such that the sum of the elements in the upper triangle is maximized.



Consider as an example the (5,5)-matrix:

$$H = \begin{bmatrix} 0 & 16 & 11 & 15 & 7 \\ 21 & 0 & 14 & 15 & 9 \\ 26 & 23 & 0 & 26 & 12 \\ 22 & 22 & 11 & 0 & 13 \\ 30 & 28 & 25 & 24 & 0 \end{bmatrix}$$

$\pi = (1, 2, 3, 4, 5)$. The sum of its superdiagonal elements is 138.
$\pi = (5, 3, 4, 2, 1)$ i.e., H_{12} becomes $H_{\pi(1)\pi(2)} = H_{54}$ in the permuted matrix.
Thus the optimal triangulation of H is

$$H^* = \begin{bmatrix} 0 & 25 & 24 & 28 & 30 \\ 12 & 0 & 26 & 23 & 26 \\ 13 & 11 & 0 & 22 & 22 \\ 9 & 14 & 15 & 0 & 21 \\ 7 & 11 & 15 & 16 & 0 \end{bmatrix}$$

Now the sum of superdiagonal elements is 247.

LOP Applications: Graph Theory

Definition: A directed graph (or digraph) D consists of a non-empty finite set $V(D)$ of distinct vertices and a finite set A of ordered pairs of distinct vertices called arcs.

Feedback arc set problem (FASP)

Input: A directed graph $D = (V, A)$, where $V = \{1, 2, \dots, n\}$, and arc weights c_{ij} for all $[ij] \in A$

Task: Find a permutation $\pi_1, \pi_2, \dots \pi_n$ of V (that is, a linear ordering of V) such that the total costs of those arcs $[\pi_j \pi_i]$ where $j > i$ (that is, the arcs that point backwards in the ordering)

$$f(\pi) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{\pi_j \pi_i}$$

is minimized.

LOP Applications: Graph Theory (2)

Definition: A linear ordering of a finite set of vertices $V = \{1, 2, \ldots, n\}$ is a bijective mapping (permutation) $\pi : \{1, 2, \ldots, n\} \to V$. For $u, v \in V$, we say that u is "before" v if $pos_\pi(u) < pos_\pi(v)$

Definition: A digraph D is complete if, for every pair x, y of distinct vertices of D both xy and yx arcs are in D.

Definition: An oriented graph is a digraph with no cycle of length two. A tournament is an oriented graph where every pair of distinct vertices are adjacent.

Remark: Given a complete digraph $D = (V, A)$ and a linear ordering of the vertices V, the arc set $E = \{[uv] \mid pos_\pi(u) < pos_\pi(v)\}$ forms an acyclic tournament on V.
Similarly, an acyclic tournament $T = (V, E)$ induces a linear ordering of V.

Definition: The cost of a linear ordering is expressed by

$$\sum_{pos_\pi(u)<pos_\pi(v)} c_{uv}$$

where the costs c_{uv} are the costs associated to the arcs.

Linear Ordering Problem

Input: Given a complete digraph $D = (V, A)$ with arc weights c_{ij} for all $ij \in A$

Task: Find an acyclic tournament $T = (V, T)$ in D such that

$$f(T) = \sum_{ij \in T} c_{ij}$$

is maximized.

Aggregation of Individual Preferences

Kemeny's problem. Suppose that there are m persons and each person i, $i = 1, ..., m$, has ranked n objects by giving a linear ordering T_i of the objects. Which common linear ordering aggregates the individual orderings in the best possible way?

$\rightsquigarrow$ linear ordering problem by setting $c_{ij} =$ number of persons preferring object O_i to object O_j

Ranking in Sports Tournaments

$H_{ij} =$ number of goals which were scored by team i against team j.

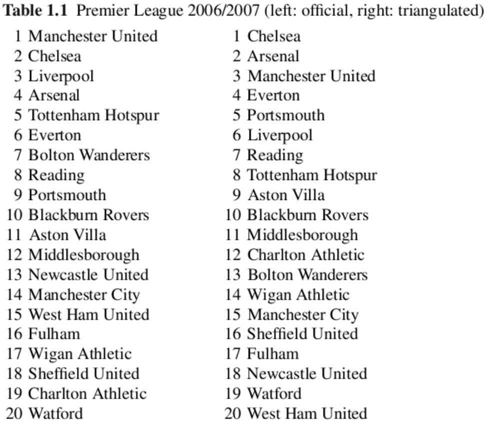


Table 1.1 Premier League 2006/2007 (left: official, right: triangulated)

1	Manchester United	1	Chelsea
2	Chelsea	2	Arsenal
3	Liverpool	3	Manchester United
4	Arsenal	4	Everton
5	Tottenham Hotspur	5	Portsmouth
6	Everton	6	Liverpool
7	Bolton Wanderers	7	Reading
8	Reading	8	Tottenham Hotspur
9	Portsmouth	9	Aston Villa
10	Blackburn Rovers	10	Blackburn Rovers
11	Aston Villa	11	Middlesborough
12	Middlesborough	12	Charlton Athletic
13	Newcastle United	13	Bolton Wanderers
14	Manchester City	14	Wigan Athletic
15	West Ham United	15	Manchester City
16	Fulham	16	Sheffield United
17	Wigan Athletic	17	Fulham
18	Sheffield United	18	Newcastle United
19	Charlton Athletic	19	Watford
20	Watford	20	West Ham United

[R. MartÃŋ and G. Reinelt. *The Linear Ordering Problem*, *Introduction*. Springer Berlin Heidelberg, 2011, 1-15]

LOP Applications: Economics

Input-output analysis (Leontief, Nobel prize)

The economy of a state is divided into n sectors, and an $n \times n$ input-output matrix C is constructed where the entry c_{ij} denotes the transactions from sector i to sector j in that year.

Triangulation (ie, solving associated LOP) allows identification of important inter-industry relations in an economy (from primary stage sectors via the manufacturing sectors to the sectors of final demand) and consequent comparisons between different countries.

Depicts dependencies between the different branches of an economy

Outline

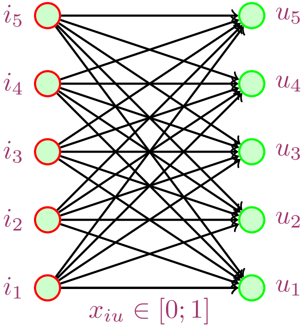
Quadratic Assignment Problem

- **Given:**
  n units with a matrix $F = [f_{ij}] \in \mathbf{R}^{n \times n}$ of flows between them and
  n locations with a matrix $D = [d_{uv}] \in \mathbf{R}^{n \times n}$ of distances
- **Task:** Find the assignment σ of units to locations that minimizes the sum of product between flows and distances, ie,

$$\min_{\sigma \in \Sigma} \sum_{i,j} f_{ij} d_{\sigma(i)\sigma(j)}$$

Applications: hospital layout; keyboard layout

Quadratic Programming Formulation



indices i, j for units and u, v for locations:

$$\min \sum_i \sum_u \sum_j \sum_v f_{ij} d_{uv} x_{iu} x_{jv} + \sum_i \sum_u c_{iu} x_{iu}$$

$$\text{s.t.} \sum_i x_{iu} = 1 \quad \forall u$$

$$\sum_u x_{iu} = 1 \quad \forall i$$

$$x \geq 0 \text{ and integer } \forall i, u$$

Largest instances solvable exactly $n = 30$

Example: QAP

$$D = \begin{pmatrix} 0 & 4 & 3 & 2 & 1 \\ 4 & 0 & 3 & 2 & 1 \\ 3 & 3 & 0 & 2 & 1 \\ 2 & 2 & 2 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} \qquad F = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 0 & 2 & 3 & 4 \\ 2 & 2 & 0 & 3 & 4 \\ 3 & 3 & 3 & 0 & 4 \\ 4 & 4 & 4 & 4 & 0 \end{pmatrix}$$

The optimal solution is $\sigma = (1, 2, 3, 4, 5)$, that is,
facility 1 is assigned to location 1,
facility 2 is assigned to location 2, etc.

The value of $f(\sigma)$ is 100.

Delta evaluation

Evaluation of 2-exchange $\{r, s\}$ can be done in $O(n)$

$$\begin{aligned}\Delta(\psi, r, s) = {} & b_{rr} \cdot (a_{\psi_s\psi_s} - a_{\psi_r\psi_r}) + b_{rs} \cdot (a_{\psi_s\psi_r} - a_{\psi_r\psi_s}) + \\ & b_{sr} \cdot (a_{\psi_r\psi_s} - a_{\psi_s\psi_r}) + b_{ss} \cdot (a_{\psi_r\psi_r} - a_{\psi_s\psi_s}) + \\ & \sum_{k=1, k\neq r,s}^{n} (b_{kr} \cdot (a_{\psi_k\psi_s} - a_{\psi_k\psi_r}) + b_{ks} \cdot (a_{\psi_k\psi_r} - a_{\psi_k\psi_s}) + \\ & b_{rk} \cdot (a_{\psi_s\psi_k} - a_{\psi_r\psi_k}) + b_{sk} \cdot (a_{\psi_r\psi_k} - a_{\psi_s\psi_k}))\end{aligned}$$

Example: Tabu Search for QAP

- **Solution representation**: permutation π
- **Initial Solution**: randomly generated
- **Neighborhood**: interchange
  $\Delta_I: \quad \delta(\pi) = \{\pi' | \pi'_k = \pi_k \text{ for all } k \neq \{i,j\} \text{ and } \pi'_i = \pi_j, \pi'_j = \pi_i\}$
- **Tabu status**: forbid δ that place back the items in the positions they have already occupied in the last `tt` iterations (short term memory)

- Implementation details:
  - compute $f(\pi') - f(\pi)$ in $O(n)$ or $O(1)$ by storing the values all possible previous moves.
  - maintain a matrix $[T_{ij}]$ of size $n \times n$ and write the last time item i was moved in location k plus tt
  - δ is tabu if it satisfies both:
    - $T_{i,\pi(j)} \geq$ current iteration
    - $T_{j,\pi(i)} \geq$ current iteration

Example: Robust Tabu Search for QAP

- **Aspiration criteria**:
  - allow forbidden δ if it improves the last π^*
  - select δ if never chosen in the last A iterations (long term memory)
- Parameters: $\texttt{tt} \in [\lfloor 0.9n \rfloor, \lceil 1.1n + 4 \rceil]$ and $A = 5n^2$

Example: Reactive Tabu Search for QAP

- **Aspiration criteria**:
  - allow forbidden δ if it improves the last π^*
- **Tabu Tenure**
  - maintain a hash table (or function) to record previously visited solutions
  - increase `tt` by a factor $\alpha_{inc}(= 1.1)$ if the current solution was previously visited
  - decrease `tt` by a factor $\alpha_{dec}(= 0.9)$ if `tt` not changed in the last $sttc$ iterations or all moves are tabu
- Trigger escape mechanism if a solution is visited more than $nr(= 3)$ times
- Escape mechanism $= 1 + (1 + r) \cdot ma/2$ random moves

Outline

1. Linear Ordering
2. Quadratic Assignment Problem
3. p-median Problem
4. Covering and Partitioning
5. Bin Packing
6. Course Timetabling
    - An Example

The p-median Problem

Given:
a set F of locations of m facilities
a set U of locations for n users
a distance matrix $D = [d_{ij}] \in \mathbf{R}^{n \times m}$
Task: Select p locations of F where to install facilities such that the sum of the distances of each user to its closest installed facility is minimized, *i.e.*,

$$\min \left\{ \sum_{i \in U} \min_{j \in J} d_{ij} \;\middle|\; J \subseteq F \text{ and } |J| = p \right\}$$

Outline

1. Linear Ordering
2. Quadratic Assignment Problem
3. p-median Problem
4. Covering and Partitioning
5. Bin Packing
6. Course Timetabling
    - An Example

Set Problems

Set Covering

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} c_j x_j \\ & \sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad \forall i \\ & x_j \in \{0,1\} \end{aligned}$$

Set Partitioning

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} c_j x_j \\ & \sum_{j=1}^{n} a_{ij} x_j = 1 \quad \forall i \\ & x_j \in \{0,1\} \end{aligned}$$

Set Packing

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j x_j \\ & \sum_{j=1}^{n} a_{ij} x_j \leq 1 \quad \forall i \\ & x_j \in \{0,1\} \end{aligned}$$

The independent set problem is equivalent to the set packing.
Vertex cover problem is a strict special case of set covering.

Outline

1. Linear Ordering
2. Quadratic Assignment Problem
3. p-median Problem
4. Covering and Partitioning
5. Bin Packing
6. Course Timetabling
    - An Example

Knapsack, Bin Packing, Cutting Stock

Knapsack

Given: a knapsack with maximum weight W and a set of n items $\{1, 2, \ldots, n\}$, with each item j associated to a profit p_j and to a weight w_j.

Task: Find the subset of items of maximal total profit and whose total weight is not greater than W.

One dimensional Bin Packing

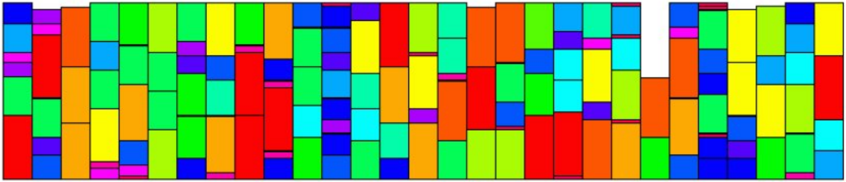
Given: A set $L = (a_1, a_2, \ldots, a_n)$ of *items*, each with a size $s(a_i) \in (0, 1]$ and an unlimited number of unit-capacity bins $B_1, B_2, \ldots, B_m$.

Task: Pack all the items into a minimum number of unit-capacity bins $B_1, B_2, \ldots, B_m$.

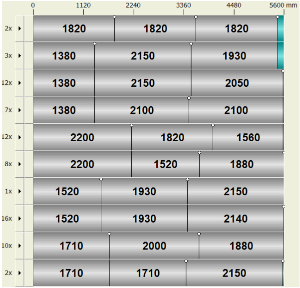
Cutting stock

Each item has a profit $p_j > 0$ and a number of times it must appear a_i. The task is to select a subset of items to be packed in a single finite bin that

Bin Packing



Cutting Stock



2x	1820	1820	1820
3x	1380	2150	1930
12x	1380	2150	2050
7x	1380	2100	2100
12x	2200	1820	1560
8x	2200	1520	1880
1x	1520	1930	2150
16x	1520	1930	2140
10x	1710	2000	1880
2x	1710	1710	2150

Two-Dimensional Packing Problems

Two dimensional bin packing

Given: A set $L = (a_1, a_2, \ldots, a_n)$ of n rectangular *items*, each with a width w_j and a height h_j and an unlimited number of identical rectangular bins of width W and height H.
Task: Allocate all the items into a minimum number of bins, such that the original orientation is respected (no rotation of the items is allowed).

Two dimensional strip packing

Given: A set $L = (a_1, a_2, \ldots, a_n)$ of n rectangular *items*, each with a width w_j and a height h_j and a bin of width W and infinite height (*a strip*).
Task: Allocate all the items into the strip by minimizing the used height and such that the original orientation is respected (no rotation of the items is allowed).

Two dimensional cutting stock

Each item has a profit $p_j > 0$ and the task is to select a subset of items to be packed in a single finite bin that maximizes the total selected profit.

Three dimensional

Given: A set $L = (a_1, a_2, \ldots, a_n)$ of rectangular *boxes*, each with a width w_j, height h_j and depth d_j and an unlimited number of three-dimensional bins $B_1, B_2, \ldots, B_m$ of width W, height H, and depth D.

Task: Pack all the boxes into a minimum number of bins, such that the original orientation is respected (no rotation of the boxes is allowed)

Outline

Course Timetabling

Input: a finite set of time periods and courses with assigned: a teacher, a set of attending students and a suitable room.

Task: Produce weekly timetable of courses, that is: assign a time period of the week (typically one hour) to every course such that courses are assigned to different time periods if:

- they are taught by the same teacher
- they use the same room
- they share students.

Course Timetabling

Definition

Feasible Course Timetabling

Input: A set of r students $S = \{1, 2, \ldots, r\}$, a set of n events $E = \{1, 2, \ldots, n\}$, a set of p periods $T = \{1, 2, \ldots, p\}$ and a set of m rooms $R = \{1, 2, \ldots, m\}$. For each student s a set of events to attend $N_s \subseteq E$, for each event i a set of unavailable periods $U_i \subseteq T$ and a set of suitable rooms $Z_i \subseteq R$. A precedence graph, i.e., a directed acyclic graph $D = (V, A)$ in which each vertex $i \in V$ represents an event $i \in E$ and each arc $(i, j) \in A$ a precedence constraint stating that the course i must precede course j.

Task: Find a timetable, i.e., a function $\phi : E \mapsto T \times R$, that satisfies all the following conditions:

All events assigned:	$\forall i \in E$	$\phi_T(i) \in T \wedge \phi_R(i) \in R$
Events:	$\forall i, j \in E$	$\phi_T(i) \neq \phi_T(j) \vee \phi_R(i) \neq \phi_R(j)$
Student conflicts:	$\forall i, j \in E : \exists s \in S : i, j \in N_s$	$\phi_T(i) \neq \phi_T(j)$
Room suitability:	$\forall i \in E$	$\phi_R(i) \in Z_i$
Availability:	$\forall i \in E$	$\phi_T(i) \notin U_i$
Precedences:	$\forall (i, j) \in A$	$\phi_T(i) < \phi_T(j)$

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

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)

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.

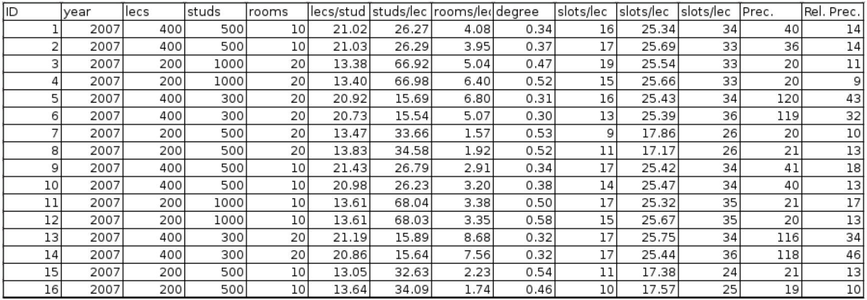
} Soft Constraints

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

Time-slots

Rooms

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

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	

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	

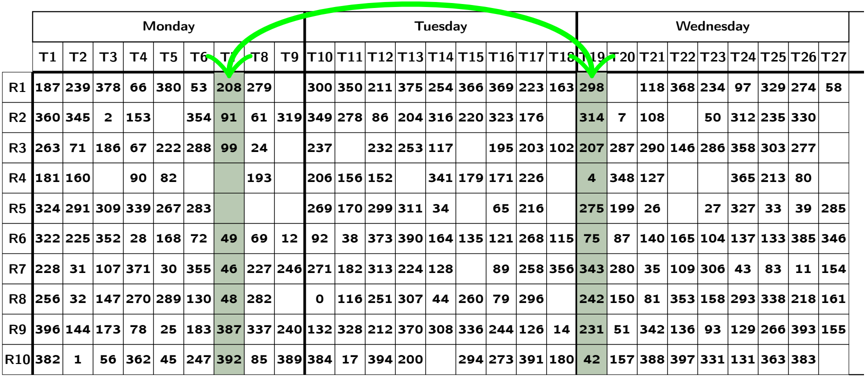
- N_1: One Exchange

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	372	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

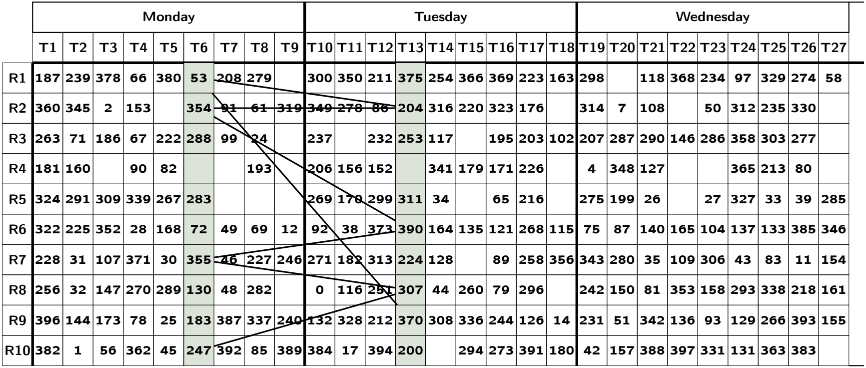
B. Room assignment included



	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_3: Period Swap

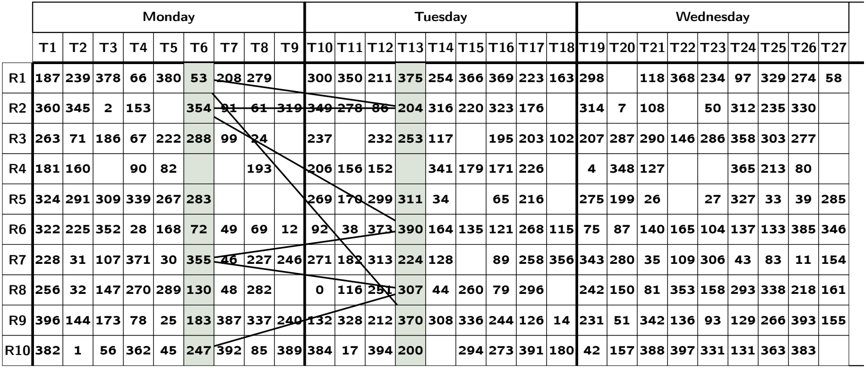
B. Room assignment included



	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_3: Period Swap
- N_4: Kempe Chain Interchange

B. Room assignment included



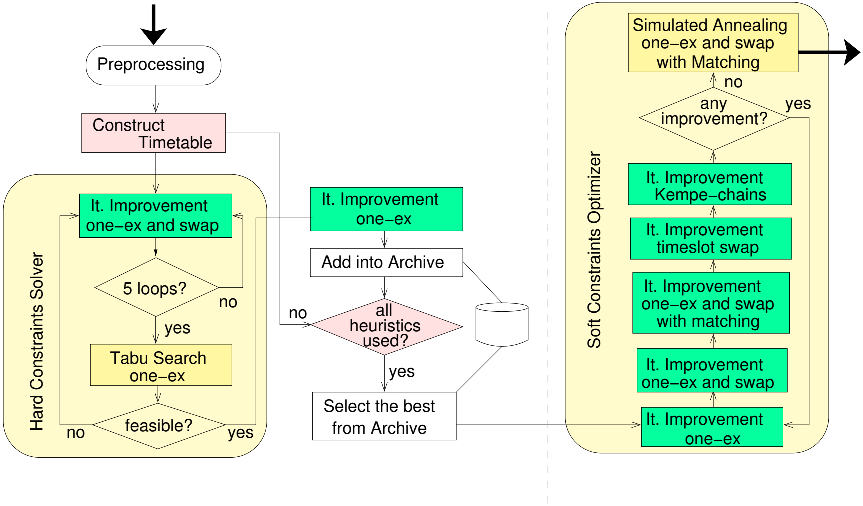
	Monday									Tuesday									Wednesday								
	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20	T21	T22	T23	T24	T25	T26	T27
R1	187	239	378	66	380	53	208	279		300	350	211	375	254	366	369	223	163	298		118	368	234	97	329	274	58
R2	360	345	2	153		354	91	61	319	349	278	86	204	316	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
```

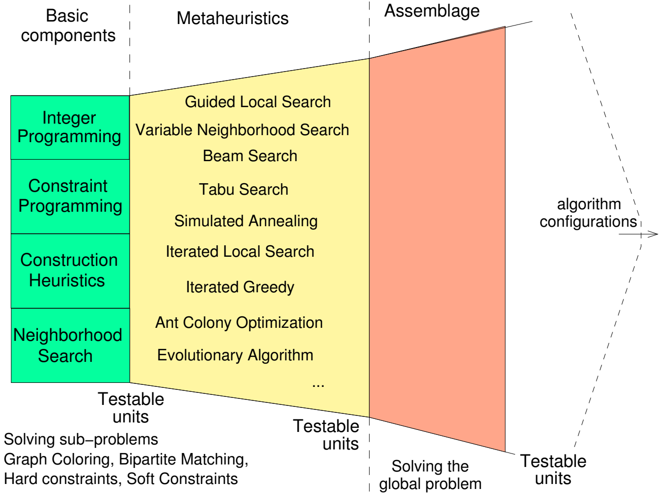
Algorithm Flowchart



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)

List of Problems

See `http://www.nada.kth.se/~viggo/problemlist/`