The University Timetabling Problem – Complexity and an Integer Linear Programming Formulation: a Case Study of UP FAMNIT

Nevena Mitrović
Assoc. Prof Martin Milanič, Assist. Prof. Jernej Vičič

University of Primorska

January 15, 2018

Decision problem is a problem containing arbitrary question that separates input data on two sets, one containing data with “yes” answer, and another with “no” answer.

Example: given two numbers x and y, check if they are coprime or not.

Decision problem is a problem containing arbitrary question that separates input data on two sets, one containing data with “yes” answer, and another with “no” answer.

Example: given two numbers x and y, check if they are coprime or not.

Optimization problem is problem of finding the best solution from the set of all feasible solutions. If set of feasible solutions is discrete, then we deal with combinatorial optimization problem.

Example: given two numbers x and y, determine the greatest divisor of x and y.

Decision problem is a problem containing arbitrary question that separates input data on two sets, one containing data with "yes" answer, and another with "no" answer.

Example: given two numbers x and y, check if they are coprime or not.

Optimization problem is problem of finding the best solution from the set of all feasible solutions. If set of feasible solutions is discrete, then we deal with combinatorial optimization problem.

Example: given two numbers x and y, determine the greatest divisor of x and y.

If problem can be solved in time which is polynomial function of the size of the input data, then we say that problem is solvable in polynomial time.

There are three complexity classes of decision problems:

- P consists of decision problems which are solvable in polynomial time.
- NP consists of decision problems for which positive answer of instance I can be verified in polynomial time.
- $co - NP$ consists of decision problems for which negative answer of instance I can be verified in polynomial time.

There are three complexity classes of decision problems:

- P consists of decision problems which are solvable in polynomial time.
- NP consists of decision problems for which positive answer of instance I can be verified in polynomial time.
- $co - NP$ consists of decision problems for which negative answer of instance I can be verified in polynomial time.

NP-hard problems: more difficult than all problems in NP

Definition

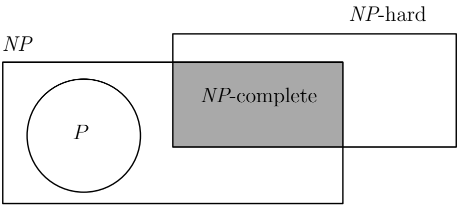
Problem Π is NP-hard if existence of polynomial algorithm for Π implies existence of polynomial algorithm for any problem in NP.

Definition

Problem Π is *NP*-complete, if it is *NP*-hard and if it is in *NP*.

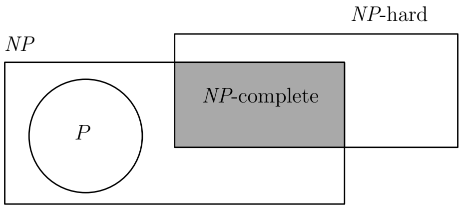
Definition

Problem Π is *NP*-complete, if it is *NP*-hard and if it is in *NP*.



Definition

Problem Π is *NP*-complete, if it is *NP*-hard and if it is in *NP*.



Problem Π_1 can be polynomially reduced to problem Π_2 if for arbitrary instance I_1 of problem Π_1 we can construct instance $I_2 = \mathcal{T}(I_1)$ of problem Π_2 in polynomial time, so that answer to I_1 in Π_1 is same as answer to I_2 in Π_2.

Theorem

Problem Π is NP-complete if it is in NP, and if there exists NP-hard problem Π' which can be polynomially reduced to Π.

Theorem

Problem Π is NP-complete if it is in NP, and if there exists NP-hard problem Π' which can be polynomially reduced to Π.

- the hardest problems in *NP*
- if some of *NP*-complete problems would be solvable in polynomial time, then so would be any *NP*-complete problem
- some *NP*-complete problems: satisfiability, maximum independent set, chromatic number,

Theorem

Problem Π is NP-complete if it is in NP, and if there exists NP-hard problem Π' which can be polynomially reduced to Π.

- the hardest problems in *NP*
- if some of *NP*-complete problems would be solvable in polynomial time, then so would be any *NP*-complete problem
- some *NP*-complete problems: satisfiability, maximum independent set, chromatic number, timetabling, ...

Timetabling problem

Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space and time, in such a way as to satisfy as nearly as possible a set of desirable constraints.

Timetabling problem

Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space and time, in such a way as to satisfy as nearly as possible a set of desirable constraints.

Timetabling problems arise in many areas of human activity, such as:

- transport companies
- production and manufacturing
- sport competitions
- **educational institutions**
- etc.

Timetabling problem

Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space and time, in such a way as to satisfy as nearly as possible a set of desirable constraints.

Timetabling problems arise in many areas of human activity, such as:

- transport companies
- production and manufacturing
- sport competitions
- **educational institutions**
- etc.

Usually it takes a lot of time to prepare the corresponding timetable by hand.

Automation of the whole timetabling process?

Example

Suppose there are 3 men, 7 tasks and 2 days. Each man can complete some of the tasks, but not all. Solving of each task lasts one day. Can we schedule this problem so that all tasks are done?

Example

Suppose there are 3 men, 7 tasks and 2 days. Each man can complete some of the tasks, but not all. Solving of each task lasts one day. Can we schedule this problem so that all tasks are done?

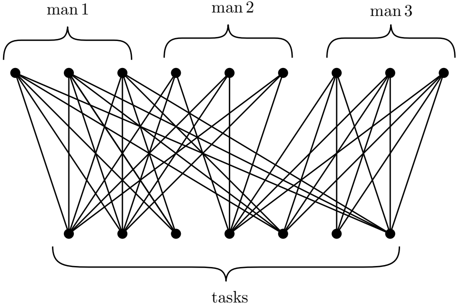
Answer: NO! A set of feasible solutions is empty.

Example

Suppose there are 3 men, 7 tasks and 2 days. Each man can complete some of the tasks, but not all. Solving of each task lasts one day. Can we schedule this problem so that all tasks are done?

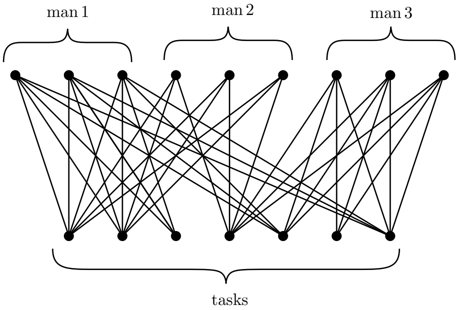
Answer: NO! A set of feasible solutions is empty.

What about the same problem with minor change: 3 men, 7 tasks and 3 days?

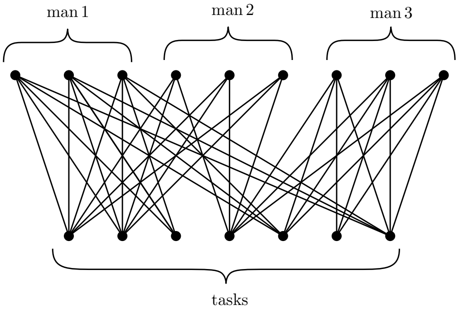


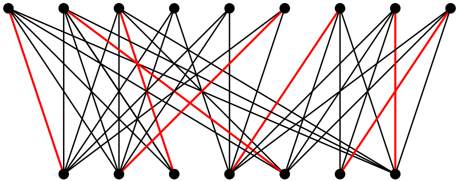
Matching M of a graph $G = (V, E)$: set of edges from E, such that no vertex from V is the endpoint of more than one edge in M.

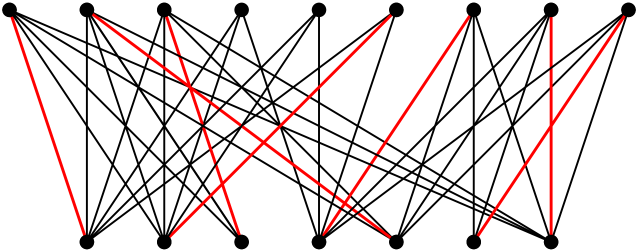
Matching M of a graph $G = (V, E)$: set of edges from E, such that no vertex from V is the endpoint of more than one edge in M.



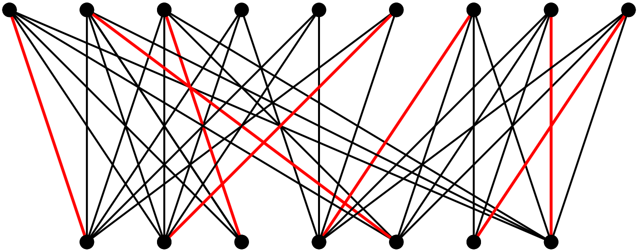
Matching M of a graph $G = (V, E)$: set of edges from E, such that no vertex from V is the endpoint of more than one edge in M.







Bipartite graphs: polynomially solvable problem!
Reason: no time restrictions for solving the tasks.



Bipartite graphs: polynomially solvable problem!

Reason: no time restrictions for solving the tasks.

If we have additional requirements, the problem becomes difficult to solve.

Timetable design problem

Instance:

- set H of "work periods",
- set C of "craftsmen",
- set T of "tasks",
- a subset $A(c) \subseteq H$ of "available hours" $\forall c \in C$,
- a subset $A(t) \subseteq H$ of "available hours" $\forall t \in T$,
- a number $R(c, t) \in Z_0^+$ of "required work periods".

Timetable design problem

Instance:

- set H of "work periods",
- set C of "craftsmen",
- set T of "tasks",
- a subset $A(c) \subseteq H$ of "available hours" $\forall c \in C$,
- a subset $A(t) \subseteq H$ of "available hours" $\forall t \in T$,
- a number $R(c, t) \in Z_0^+$ of "required work periods".

Question:

Is there a function $f : C \times T \times H \to \{0, 1\}$, so that

1. $f(c, t, h) = 1$ only if $h \in A(c) \cap A(t)$,
2. $\forall h \in H$, $\forall c \in C$ there is at most one $t \in T$ for which $f(c, t, h) = 1$,
3. $\forall h \in H$, $\forall t \in T$ there is at most one $c \in C$ for which $f(c, t, h) = 1$,
4. $\forall (c, t) \in C \times T$ there are exactly $R(c, t)$ values of h for which $f(c, t, h) = 1$.

Complexity

TD is *NP*-complete, even if $R(c, t) \in \{0, 1\}$ for all $c \in C, t \in T$.

Complexity

TD is *NP*-complete, even if $R(c, t) \in \{0, 1\}$ for all $c \in C, t \in T$.

University course timetabling problem: the process of assigning university courses to specific time periods throughout the 5 working days of the week and to specified classrooms suitable for the needs of each course.

Requirements depend on the teaching process of the institution:
Constraints are divided in two sets:

- HARD CONSTRAINTS: must be satisfied.
  Teacher can not teach two different courses at the same time.

Complexity

TD is *NP*-complete, even if $R(c, t) \in \{0, 1\}$ for all $c \in C, t \in T$.

University course timetabling problem: the process of assigning university courses to specific time periods throughout the 5 working days of the week and to specified classrooms suitable for the needs of each course.

Requirements depend on the teaching process of the institution:
Constraints are divided in two sets:

- HARD CONSTRAINTS: must be satisfied.
  Teacher can not teach two different courses at the same time.
- SOFT CONSTRAINTS: can be violated, but for each violation we determine penalties.
  Students would like to have as compact timetable as possible.

Constraints relevant for UP FAMNIT

Hard constraints: constraints that must be satisfied.

A) Every meeting has to be assigned to available resources.
B) Overlapping is not permitted.
C) The timetable has to be complete.
D) Pre-scheduled meetings.
E) Upper bounds on the number of hours per lecturer per day.
F) Students' restrictions.

Constraints relevant for UP FAMNIT

Soft constraints: constraints that are desired to be satisfied, but violation of some of them has no influence on the feasibility of the timetable.

S_1) Minimize use of payable classrooms.

S_2) Compact timetable from the lecturers' point of view.

S_3) Requirements related to students: afternoon meetings for some students groups, upper bound on number of hours per day, minimized number of hours at Friday in the afternoon.

S_4) Requirements related to lecturers: measure of lecturers' preferences to some timeslots.

Famnit Timetabling Design – FTD

FTD is a natural generalization of the UP FAMNIT timetabling problem.

Famnit Timetabling Design – FTD

FTD is a natural generalization of the UP FAMNIT timetabling problem.

Basic structural elements:

- a set D of days,
- a set T of timeslots,
- a set C of courses,
- a set S of student groups,
- a set L of lecturers,
- a set M of meetings,

FTD is a natural generalization of the UP FAMNIT timetabling problem.

Basic structural elements:

- a set D of days,
- a set T of timeslots,
- a set C of courses,
- a set S of student groups,
- a set L of lecturers,
- a set M of meetings,

Meeting m – ordered pair with first coordinate being course and second a groups of students.

Example: $m = (c, \{s_1, s_2\})$.

For each $m \in M$ a set of corresponding lecturers and student groups, as well as a division into blocks and type are known.

Famnit Timetable Design – FTD

FTD is a natural generalization of the UP FAMNIT timetabling problem.

Basic structural elements:

- a set D of days,
- a set T of timeslots,
- a set C of courses,
- a set S of student groups,
- a set L of lecturers,
- a set M of meetings,

Famnit Timetable Design – FTD

FTD is a natural generalization of the UP FAMNIT timetabling problem.

Basic structural elements:

- a set D of days,
- a set T of timeslots,
- a set C of courses,
- a set S of student groups,
- a set L of lecturers,
- a set M of meetings,
- a set K of locations,
- a multi-set $P(m) \subseteq \mathbb{N}$ consisting of blocks of meeting m,
- subsets $T(\ell), T(r) \subseteq T$ of available timeslots for lecturer and room,

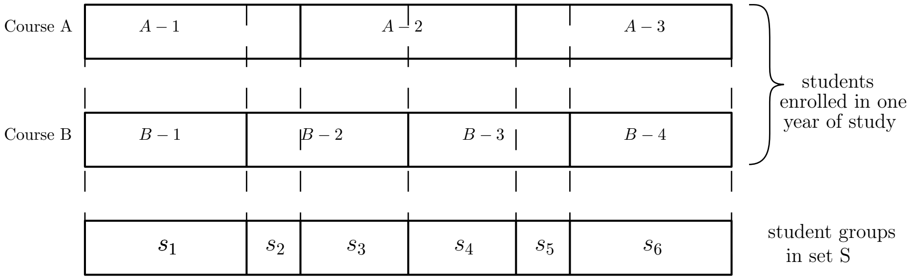
Famnit Timetable Design – FTD

FTD is a natural generalization of the UP FAMNIT timetabling problem.

Basic structural elements:

- a set D of days,
- a set T of timeslots,
- a set C of courses,
- a set S of student groups,
- a set L of lecturers,
- a set M of meetings,
- a set K of locations,
- a multi-set $P(m) \subseteq \mathbb{N}$ consisting of blocks of meeting m,
- subsets $T(\ell), T(r) \subseteq T$ of available timeslots for lecturer and room,
- subsets $M(s), M(\ell) \subseteq M$ of meetings incident with student group and lecturer,
- a subset $R(m) \subseteq R$ of acceptable rooms for meeting m,
- a subset $M(k) \subseteq M$ of meetings taking place at location k,
- a set of pre-scheduled meetings,
- a number $\rho(\ell) \in \mathbb{N}$ representing the maximum number of teaching hours for lecturer per day.

Students' sectioning



Famnit Timetable Design

Instance: sets T, D, M, R, L, S, K, subsets $T(m), T(r), M(s), M(\ell), R(m), M(k)$, multi-set $P(m)$, number $\rho(\ell)$.

Instance: sets T, D, M, R, L, S, K, subsets $T(m), T(r), M(s), M(\ell), R(m), M(k)$, multi-set $P(m)$, number $\rho(\ell)$.

Question: Is there a timetable that schedules all meetings,

Instance: sets T, D, M, R, L, S, K, subsets $T(m), T(r), M(s), M(\ell), R(m), M(k)$, multi-set $P(m)$, number $\rho(\ell)$.

Question: Is there a timetable that schedules all meetings, that is, a function $f : M \times T \times R \to \{0, 1\}$ (where $f(m, t, r) = 1$ means that meeting m is assigned to timeslot t and room r) that schedules the desired number of hours of all meetings and satisfies all hard constraints?

Instance: sets T, D, M, R, L, S, K, subsets $T(m), T(r), M(s), M(\ell), R(m), M(k)$, multi-set $P(m)$, number $\rho(\ell)$.

Question: Is there a timetable that schedules all meetings, that is, a function $f : M \times T \times R \to \{0, 1\}$ (where $f(m, t, r) = 1$ means that meeting m is assigned to timeslot t and room r) that schedules the desired number of hours of all meetings and satisfies all hard constraints?

Soft constraints are modeled with the objective function:
we introduce a penalty term for each violated soft constraint, and we would like to **minimize** the sum of all penalties.

FTD is *NP*-complete

Recall:

A problem Π is *NP-complete* if it is in *NP* and there exists some known *NP*-complete problem that polynomially reduces to Π.

FTD is *NP*-complete

Recall:

A problem Π is *NP-complete* if it is in *NP* and there exists some known *NP*-complete problem that polynomially reduces to Π.

We proved that TIMETABLE DESIGN polynomially reduces to FAMNIT TIMETABLE DESIGN.

FTD is *NP*-complete

Recall:

A problem Π is *NP-complete* if it is in *NP* and there exists some known *NP*-complete problem that polynomially reduces to Π.

We proved that TIMETABLE DESIGN polynomially reduces to FAMNIT TIMETABLE DESIGN.

Theorem

FAMNIT TIMETABLE DESIGN is *NP*-complete!

Literature overview

Approaches that have been proposed in the literature:

- graph coloring,
- metaheuristics,
- neural networks,
- constraint logic programming,
- integer linear programming.

Graph coloring

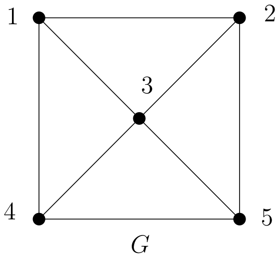
Proper k-coloring of graph G is coloring of vertices of G with at most k colors, so that adjacent vertices have distinct colors.

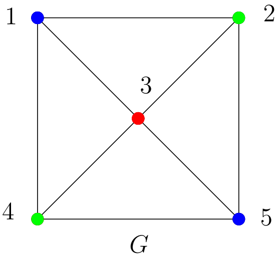
Given a graph $G = (V, E)$, and integer k, does there exist a proper k-coloring of G?

Graph coloring

Proper k-coloring of graph G is coloring of vertices of G with at most k colors, so that adjacent vertices have distinct colors.

Given a graph $G = (V, E)$, and integer k, does there exist a proper k-coloring of G?





Graph coloring for timetabling problem

Given:

- set C of courses $c_1, c_2, \ldots c_{|C|}$,
- set R of classrooms $r_1, r_2, \ldots r_{|R|}$,
- set $W \subseteq C \times R$ of pairs (c_i, r_j) for which holds that room r_j is acceptable for course c_i (it can be given with matrix of size $|C||R|$)

Graph coloring for timetabling problem

Given:

- set C of courses $c_1, c_2, \dots c_{|C|}$,
- set R of classrooms $r_1, r_2, \dots r_{|R|}$,
- set $W \subseteq C \times R$ of pairs (c_i, r_j) for which holds that room r_j is acceptable for course c_i (it can be given with matrix of size $|C||R|$)

Construct a graph $G = (V, E)$ so that

Graph coloring for timetabling problem

Given:

- set C of courses c_1, c_2, $\dots c_{|C|}$,
- set R of classrooms r_1, r_2, $\dots r_{|R|}$,
- set $W \subseteq C \times R$ of pairs (c_i, r_j) for which holds that room r_j is acceptable for course c_i (it can be given with matrix of size $|C||R|$)

Construct a graph $G = (V, E)$ so that

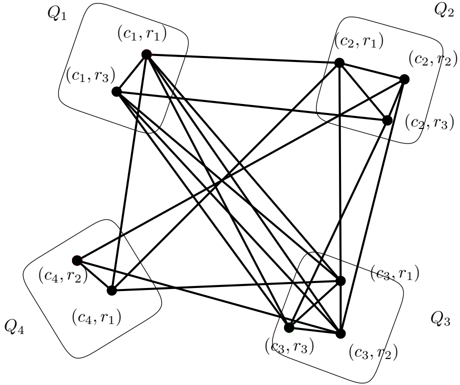
- V consists of $|C|$ cliques, $Q_1, Q_2, \dots, Q_{|C|}$, where clique i contains vertex for each classroom which is acceptable for course c_i; such vertices are labelled with pairs (c_i, r_j)
- vertices which agree at first coordinate are in the same clique
- add edges between vertices of same second coordinate
- if courses c_i and c_j are in conflict (overlapping is not allowed), then add all edges between cliques Q_i and Q_j

Example

Let $C = \{c_1, c_2, c_3, c_4\}$, $R = \{r_1, r_2, r_3\}$, and $W' = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$, and let courses c_1 and c_3 be in conflict.

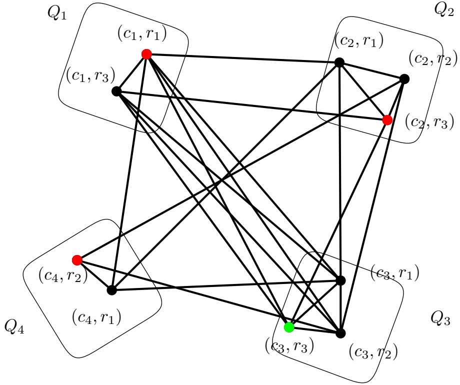
Example

Let $C = \{c_1, c_2, c_3, c_4\}$, $R = \{r_1, r_2, r_3\}$, and $W' = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$, and let courses c_1 and c_3 be in conflict.

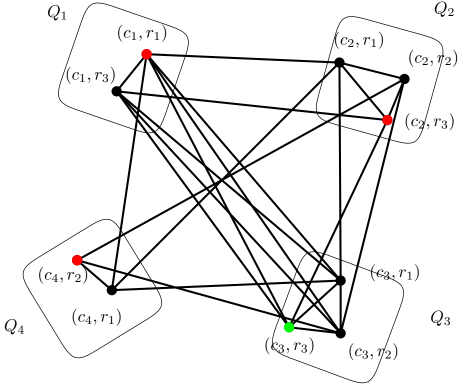


We would like to color graph G so that exactly one vertex is colored in each clique Q_i, and two adjacent vertices can not have the same color.

We would like to color graph G so that exactly one vertex is colored in each clique Q_i, and two adjacent vertices can not have the same color.



We would like to color graph G so that exactly one vertex is colored in each clique Q_i, and two adjacent vertices can not have the same color.



Not enough known about complexity of such algorithm. Almost impossible to model certain constraints.

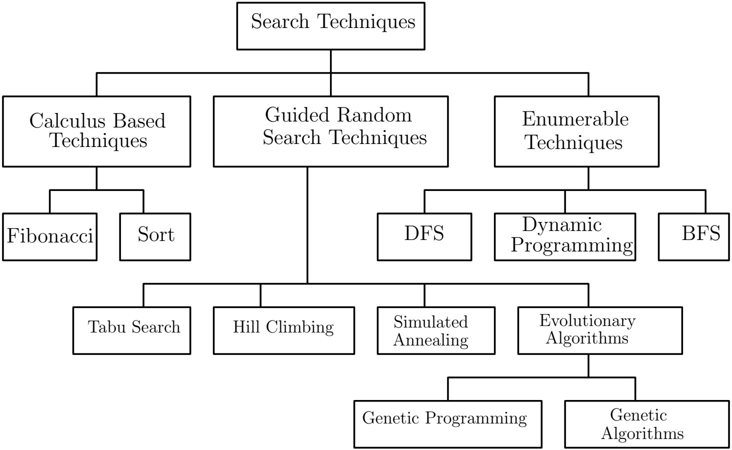
Search techniques

A search algorithm is an algorithm that retrieves information stored within some data structure.

Sometimes very good solution can be found in few steps, but sometimes it is not found at all.

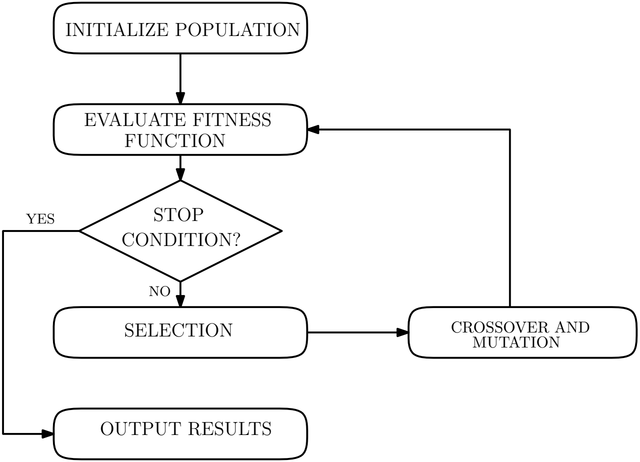
Search techniques

A search algorithm is an algorithm that retrieves information stored within some data structure.
Sometimes very good solution can be found in few steps, but sometimes it is not found at all.



Genetic algorithms

Population of potential solutions is evaluated similarly as in biology.



A linear program in standard form is defined as

$$\begin{array}{lrcl} \text{minimize} & \mathbf{c}^{\mathbf{T}}\mathbf{x} & & \\ \text{subject to} & \mathbf{Ax} & = & \mathbf{b}\ , \\ & \mathbf{x} & \geq & \mathbf{0}, \end{array}$$

where $\mathbf{c} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^m, \mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{x} \in \mathbb{R}^n$ is the vector of variables.

A linear program in standard form is defined as

$$\begin{array}{lrcl} \text{minimize} & \mathbf{c}^{\mathbf{T}}\mathbf{x} & & \\ \text{subject to} & \mathbf{A}\mathbf{x} & = & \mathbf{b}\ , \\ & \mathbf{x} & \geq & \mathbf{0}, \end{array}$$

where $\mathbf{c} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^m, \mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{x} \in \mathbb{R}^n$ is the vector of variables.

If we add the requirement that $\mathbf{x}$ must be integer-valued, we get the *integer linear program*.

A linear program in standard form is defined as

$$
\begin{array}{lrcl}
\text{minimize} & \mathbf{c}^{\mathbf{T}}\mathbf{x} & & \\
\text{subject to} & \mathbf{A}\mathbf{x} & = & \mathbf{b}\ , \\
 & \mathbf{x} & \geq & \mathbf{0},
\end{array}
$$

where $\mathbf{c} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^m, \mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{x} \in \mathbb{R}^n$ is the vector of variables.

If we add the requirement that $\mathbf{x}$ must be integer-valued, we get the *integer linear program*.

LP = optimization of linear function with respect to linear conditions
ILP = LP with variables restricted to be integer (it is not possible to have 2.5 cars)

When we deal with events, variables are usually restricted to be binary: event happens or not.

Example of ILP

Consider a problem from the beginning:

- a set of men: $M = \{1, 2, 3\}$
- a set of tasks: $T = \{1, 2, 3, 4, 5, 6, 7\}$
- a set of days: $D = \{1, 2, 3\}$
- a number of required hours $R(m, t)$.

Example of ILP

Consider a problem from the beginning:

- a set of men: $M = \{1, 2, 3\}$
- a set of tasks: $T = \{1, 2, 3, 4, 5, 6, 7\}$
- a set of days: $D = \{1, 2, 3\}$
- a number of required hours $R(m, t)$.

We define

$$x_{m,t,d} = \begin{cases} 1, & \text{if task } t \text{ is taken at day } d, \text{ by man } m \\ 0, & \text{otherwise} \end{cases}$$

So we have $x_{1,3,2} = 1$ if man 1 works on task 3 during the day 2.

Example of ILP

Consider a problem from the beginning:

- a set of men: $M = \{1, 2, 3\}$
- a set of tasks: $T = \{1, 2, 3, 4, 5, 6, 7\}$
- a set of days: $D = \{1, 2, 3\}$
- a number of required hours $R(m, t)$.

We define

$$x_{m,t,d} = \begin{cases} 1, & \text{if task } t \text{ is taken at day } d, \text{ by man } m \\ 0, & \text{otherwise} \end{cases}$$

So we have $x_{1,3,2} = 1$ if man 1 works on task 3 during the day 2.

One constraint: task 1 can be done just once and by one man:

$$x_{1,1,1} + x_{2,1,1} + x_{3,1,1} + x_{1,1,2} + x_{2,1,2} + x_{3,1,2} = 1$$

ILP model for FTD – variables

We have variables of three types.

ILP model for FTD – variables

We have variables of three types.

(i) For every triple of a meeting $m \in M$, a timeslot $t \in T$, and a room $r \in R_m$ that is acceptable for that meeting, there is one corresponding variable $x_{m,t,r}$.

$$x_{m,t,r} = \begin{cases} 1, & \text{if meeting } m \text{ is scheduled at timeslot } t \text{ in classroom } r, \\ 0, & \text{otherwise.} \end{cases}$$

ILP model for FTD – variables

We have variables of three types.

(i) For every triple of a meeting $m \in M$, a timeslot $t \in T$, and a room $r \in R_m$ that is acceptable for that meeting, there is one corresponding variable $x_{m,t,r}$.

$$x_{m,t,r} = \begin{cases} 1, & \text{if meeting } m \text{ is scheduled at timeslot } t \text{ in classroom } r, \\ 0, & \text{otherwise.} \end{cases}$$

(ii) For every triple of a meeting $m \in M$, a timeslot $t \in T$ and a predefined length $i \in H_m$ of individual blocks of meeting m we define a variable $y_{m,t,i}$.

$$y_{m,t,i} = \begin{cases} 1, & \text{if timeslot } t \text{ is first appearance of } i \text{ consecutive} \\ & \quad \text{hours of meeting } m \\ 0, & \text{otherwise.} \end{cases}$$

(iii) Auxiliary z-variables used for soft constraints.

Constraints

$$\sum_{m \in M_\ell} \sum_{t \in T \setminus T_\ell} \sum_{r \in R_m} x_{m,t,r} = 0 \quad \forall \ell \in L.$$

$$\sum_{(m,r) \in M^R} \sum_{t \in T \setminus T_r} x_{m,t,r} = 0, \quad \forall r \in R.$$

$$\sum_{m \in M_s} \sum_{r \in R_m} x_{m,t,r} \leq 1 \quad \forall s \in S, \forall t \in T.$$

$$\sum_{t \in T} \sum_{r \in R_m} x_{m,t,r} = a_m \quad \forall m \in M.$$

$$\sum_{i \in H_m} \sum_{t \in T_d} y_{m,t,i} \leq 1, \quad \forall m \in M, \forall d \in D.$$

$$i \cdot \sum_{t \in T_d} y_{m,t,i} \leq \sum_{r \in R_m} \sum_{t \in T_d} x_{m,t,r}, \quad \forall m \in M, \forall d \in D, \forall i \in H_m.$$

$$y_{m,t,i} = 0 \quad \forall m \in M, \forall i \in H_m, \forall t = (d,h) \in T \text{ s.t. } h > \tau - i + 1.$$

$$\vdots$$

Objective function

Every variable z (or x, y) that should be respected in objective function is desired to have value 0. If variable z has value 1, we increase the value of objective function using the corresponding weight w_z.

Objective function

Every variable z (or x, y) that should be respected in objective function is desired to have value 0. If variable z has value 1, we increase the value of objective function using the corresponding weight w_z.

$$\sum_{t \in T_r} \sum_{m \in M} \sum_{r \in R_m} w_{S_1,r,t} \cdot x_{m,t,r} + \sum_{\ell \in L_+} \sum_{d \in D} w_{S_2,\ell,d} \cdot z_{S_2,\ell,d} +$$

$$\sum_{m \in M_{PM}} \sum_{t \in T \setminus T_{PM}} \sum_{i \in H_m} w_{S_3,m} \cdot y_{m,t,i} + \sum_{m \in M} \sum_{t \in T_5 \cap T_{PM}} \sum_{r \in R_m} w_{S_3,m,t} \cdot x_{m,t,r} +$$

$$\sum_{d \in D} \sum_{s \in S} w_{S_3,s,d} \cdot z_{S_3,s,d} + \sum_{\ell \in L} \sum_{m \in M(\ell)} \sum_{t \in T} \sum_{r \in R_m} w_{S_4,\ell,t} \cdot x_{m,t,r}.$$

Size of ILP

For input data for Spring semester of academic year 2016/17, we have

- 185 meetings,
- 26 classrooms,
- 65 timeslots,
- 48 student groups,
- 118 lecturers

In total:

- $171,455$ variables,
- $2,752,376$ constraints,
- $7,780,635$ entries of constraint matrix are nonzero.

Results

A model is implemented using the programming language Zimpl and evaluated using Gurobi software.

Results

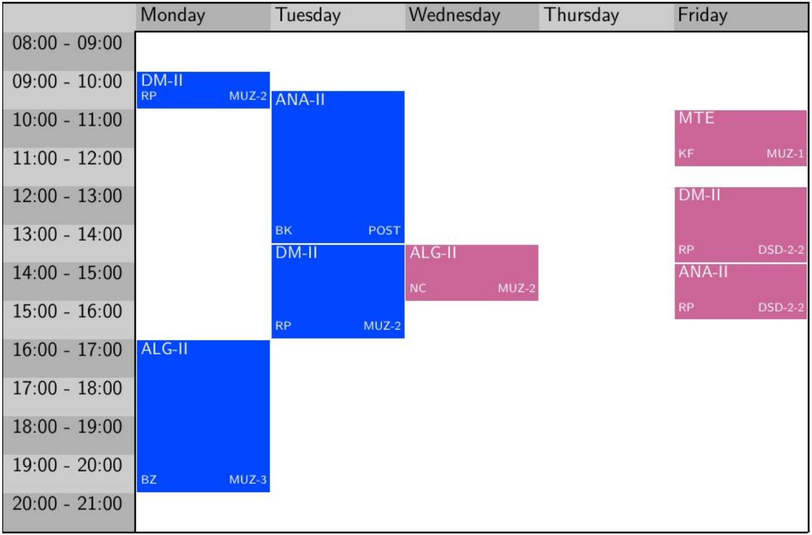
A model is implemented using the programming language Zimpl and evaluated using Gurobi software.

- For the whole model in 48 hours we did not get the feasible solution.
- For the model with objective function concerning just the necessity of afternoon meetings for corresponding courses of Master's programmes, we got solution in $\approx$ 20000s, that is in $\approx$ 6 hours. Obtained solution was the first feasible solution, and optimal at the same time.

Results: timetable for the student group MA1

	Monday	Tuesday	Wednesday	Thursday	Friday
08:00 - 09:00	MTE KF F-VP				
09:00 - 10:00					
10:00 - 11:00	DM-II RP DSD-3-1				
11:00 - 12:00					ANA-II BK DSD-2-2
12:00 - 13:00					
13:00 - 14:00					
14:00 - 15:00					
15:00 - 16:00				DM-II RP MUZ-2	
16:00 - 17:00	ALG-II NC F-MP1				
17:00 - 18:00		ALG-II BZ F-VP			
18:00 - 19:00					
19:00 - 20:00				ANA-II RP F-RU2	
20:00 - 21:00					

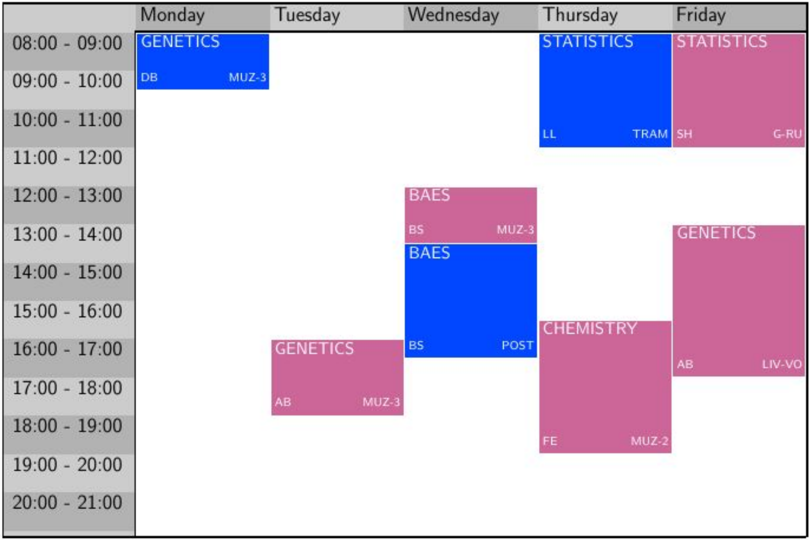
Real timetable for the student group MA1



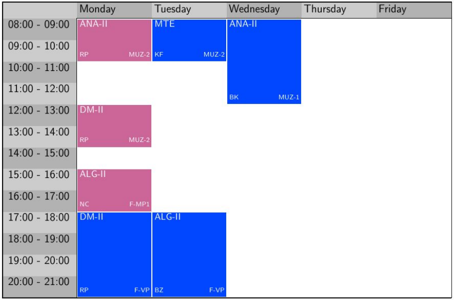
Results: timetable for the student group BI2

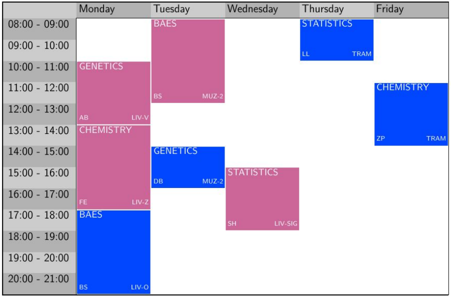
<table>
<tr><th></th><th>Monday</th><th>Tuesday</th><th>Wednesday</th><th>Thursday</th><th>Friday</th></tr>
<tr><td>08:00 - 09:00</td><td></td><td></td><td rowspan="4">CHEMISTRY
FE LIV-ZR</td><td rowspan="4">BAES
BS LIV-O</td><td></td></tr>
<tr><td>09:00 - 10:00</td><td></td><td rowspan="2">GENETICS
DB LIV-S</td><td rowspan="3">CHEMISTRY
ZP LIV-S</td></tr>
<tr><td>10:00 - 11:00</td><td></td></tr>
<tr><td>11:00 - 12:00</td><td></td><td rowspan="3">STATISTICS
SH LIV-S</td></tr>
<tr><td>12:00 - 13:00</td><td></td><td></td><td></td><td></td></tr>
<tr><td>13:00 - 14:00</td><td></td><td></td><td></td><td></td></tr>
<tr><td>14:00 - 15:00</td><td></td><td rowspan="2">STATISTICS
LL LIV-S</td><td></td><td></td><td></td></tr>
<tr><td>15:00 - 16:00</td><td></td><td></td><td></td><td></td></tr>
<tr><td>16:00 - 17:00</td><td></td><td></td><td rowspan="3">GENETICS
AB LIV-V</td><td></td><td></td></tr>
<tr><td>17:00 - 18:00</td><td></td><td></td><td></td><td></td></tr>
<tr><td>18:00 - 19:00</td><td></td><td></td><td></td><td></td></tr>
<tr><td>19:00 - 20:00</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>20:00 - 21:00</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

Real timetable for the student group BI2



Last results





Last results

	Monday	Tuesday	Wednesday	Thursday	Friday
08:00 - 09:00			ANA-II		
09:00 - 10:00					
10:00 - 11:00					
11:00 - 12:00					
12:00 - 13:00					
13:00 - 14:00	ANA-IV				
14:00 - 15:00					
15:00 - 16:00					
16:00 - 17:00					
17:00 - 18:00					
18:00 - 19:00					
19:00 - 20:00					
20:00 - 21:00					

	Monday	Tuesday	Wednesday	Thursday	Friday
08:00 - 09:00		TEO.GRAF.			
09:00 - 10:00					
10:00 - 11:00					
11:00 - 12:00					
12:00 - 13:00		DM-II	BIONF		
13:00 - 14:00					
14:00 - 15:00					
15:00 - 16:00					
16:00 - 17:00			DM-II		
17:00 - 18:00					
18:00 - 19:00			ALG.NA GR.		
19:00 - 20:00					
20:00 - 21:00					

	Monday	Tuesday	Wednesday	Thursday	Friday
08:00 - 09:00				RAZVOJ IGER	
09:00 - 10:00					
10:00 - 11:00					
11:00 - 12:00		SIST.PROG.			
12:00 - 13:00					
13:00 - 14:00				JEZ.TEHN.	
14:00 - 15:00					
15:00 - 16:00					
16:00 - 17:00					
17:00 - 18:00					
18:00 - 19:00					
19:00 - 20:00					
20:00 - 21:00					

Summary

Advantages:

- it is not so difficult to interpret the solution obtained by ILP,
- we modeled almost all conditions for FTD,
- conditions that are relevant just for some special courses can be easily represented by ILP.

Disadvantages:

- number of variables rapidly grows,
- difficult to reoptimize,
- impossible to solve on the week level.

Future work

We constructed a feasible timetable.

Next step: construction of the timetable with respect to the whole objective function.

Future work

We constructed a feasible timetable.

Next step: construction of the timetable with respect to the whole objective function.

Goals for the future:

- Reduce the amount of time required to find a solution.
- Automate all steps of the process.

Future work

We constructed a feasible timetable.

Next step: construction of the timetable with respect to the whole objective function.

Goals for the future:

- Reduce the amount of time required to find a solution.
- Automate all steps of the process.
- **Use the automated timetable in the practice.**

THANK YOU FOR ATTENTION!