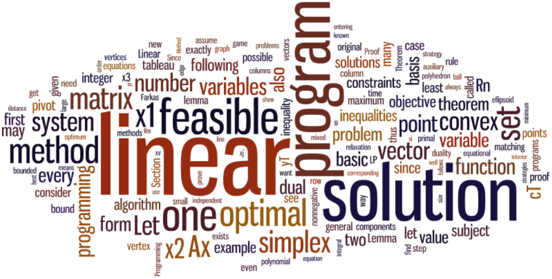
Integer Linear Programs: A ‘Real World’ Primer

Matthias Rost
Technische Universität Berlin



FG INET Retreat 2016

Overview

What you should expect

- What are Integer Linear Programs (ILPs); what can they be used for?
- Glimpse at tools: Gurobi[a] and GMPL[b]
- Glimpse at modeling 'real world' examples
- Example files can be downloaded at: https://matthias-rost.de/doc/rost-ilp-primer.tar.gz

[a] https://www.gurobi.com/documentation/6.5/quickstart_linux/index.html
[b] ftp://ftp.gnu.org/gnu/glpk/
https://en.wikibooks.org/wiki/GLPK

What you *should not* expect

- Fancy algorithmic techniques to solve Integer Linear Programs (ILPs)
- Mathematical background of Integer Linear Programming
- Real-world research examples

Motivation

Integer Linear Programming ...

- ... is one of the cornerstones of optimization.
- ... is a matured technology to solve really big problems.
- ... is quite easy to understand and apply.
- ... can be used to solve '80%' of the NP-hard optimization problems.

General Applications...

- Route planning / timetables / crew scheduling (BVG, DB, ...)
- Frequency assignments for cellular networks (Nokia, Telekom, ...)
- Verification / validation of chip designs (Siemens, ...)
- Scheduling the production process in chemical plants (BASF, ...)

Introduction to Linear Programs

$$
\begin{array}{ll}
\text{Maximize the value} & x_1 + x_2 \\
\text{among all vectors } (x_1, x_2) \in \mathbb{R}^2 & \\
\text{satisfying the constraints} & x_1 \geq 0 \\
 & x_2 \geq 0 \\
 & x_2 - x_1 \leq 1 \\
 & x_1 + 6x_2 \leq 15 \\
 & 4x_1 - x_2 \leq 10.
\end{array}
$$

Figure : Example taken from the great book by Matousek and Gärtner [2007].

Introduction to Linear Programs

$$\text{Maximize } x_1 + x_2$$

$$x_1 \geq 0$$
$$x_2 \geq 0$$
$$x_2 - x_1 \leq 1$$
$$x_1 + 6x_2 \leq 15$$
$$4x_1 - x_2 \leq 10.$$

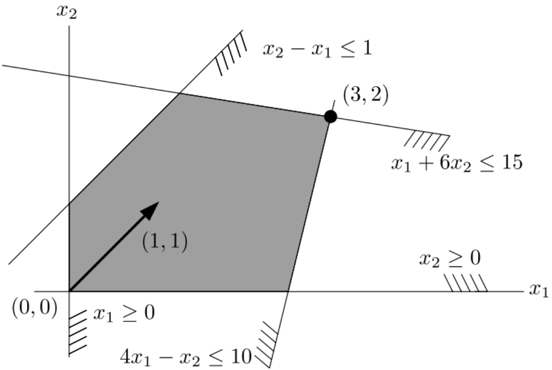


Figure : Example taken from the great book by Matousek and Gärtner [2007].

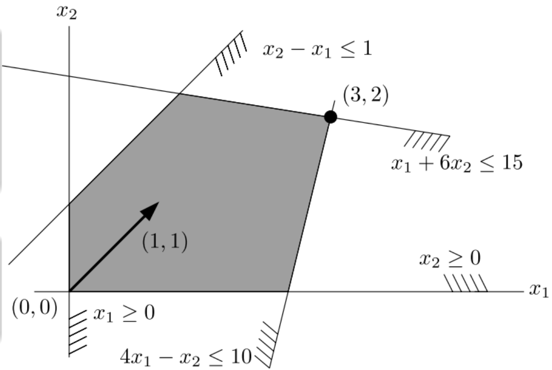
Introduction to Linear Programs

~~Integer~~ Linear Programs

- continuous variables
- *linear* constraints
- *linear* objective function

Solving ~~Integer~~ Linear Programs...

...is in P.



Using Gurobi[1]: Python File 101.py [?]

```
from gurobipy import *                              #import gurobi library

m = Model("101")                                    #create model

x_1 = m.addVar(name="x_1",lb=0,ub=GRB.INFINITY) #add variable
x_2 = m.addVar(name="x_2",lb=0,ub=GRB.INFINITY) #add variable
m.update()                                          #render variables accessible

m.addConstr(x_2 - x_1 <= 1)                         #add constraint
m.addConstr(x_1 + 6*x_2 <= 15)                      #add constraint
m.addConstr(4*x_1-x_2 <= 10)                        #add constraint

m.setObjective(x_1 + x_2, sense=GRB.MAXIMIZE)       #set objective function

m.update()                                          #realize model changes
m.optimize()                                        #optimize it!

print "\nGurobi finished computing!\n"

if m.getAttr("Status") == 2:                        #check if (optimal) solution was found
    print "Optimal solution of value {} was computed: x_1 = {} and x_2 = {}".\
        format(m.getAttr("ObjVal"), x_1.X, x_2.X)
else:
    print "No optimal solution was found!"
```

[1] https://www.gurobi.com/documentation/6.5/quickstart_linux/index.html

Using Gurobi[2]: Output of 101.py[3]

```
Optimize a model with 3 rows, 2 columns and 6 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 6e+00]
  Objective range [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 2e+01]
Presolve time: 0.00s
Presolved: 3 rows, 2 columns, 6 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.0000000e+30   1.625000e+30   2.000000e+00      0s
       2    5.0000000e+00   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.00 seconds
Optimal objective  5.000000000e+00

Gurobi finished computing!

Optimal solution of value 5.0 was computed: x_1 = 3.0 and x_2 = 2.0
```

[2] https://www.gurobi.com/documentation/6.5/quickstart_linux/index.html

[3] https://matthias-rost.de/doc/rost-ilp-primer.tar.gz

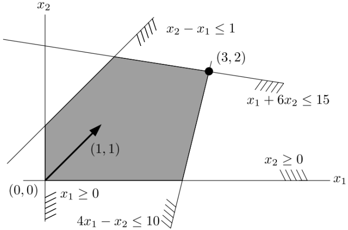
Bottomline

'Programming' Integer Linear Programs is simple

1. → *Come up with the right mathematical model.* ←
2. Choose any of the commercial (Gurobi, CPLEX, XPress, ...) or any of the open-source solvers (SCIP, GLPSOL, ...).
3. Input the model to the respective solver via APIs, files, ...
4. Let the *solver* do the job and obtain an optimal / near-optimal solution.

$$\text{Maximize } x_1 + x_2$$

$$\begin{aligned} x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_2 - x_1 &\leq 1 \\ x_1 + 6x_2 &\leq 15 \\ 4x_1 - x_2 &\leq 10. \end{aligned}$$



Modeling: Assigning Students to Tutorials

MOSES Problem 2014

MOSES assigned a lot of students to tutorials at time slots at which mandatory courses took place. Students were assigned arbitrarily to tutorials with free seats.
Hence, many students wanted to change their assigned tutorial.

Input from Florian's OSIRIS System

- Set of students S (who want to change their assigned tutorials)
- Current tutorial $current\_tutorial_s \in T$ for each student $s \in S$
- Set $Wanted\_Tutorials_s \subseteq T$ of wanted tutorials for student $s \in S$
- Number of attendees $current\_load_t \in \mathbb{N}$ for each tutorial $t \in T$
- Capacity $capacity_t \in \mathbb{N}$ for each tutorial $t \in T$

Modeling: Assigning Students to Tutorials

Input from Florian's OSIRIS System

- Set of students S (who want to change their assigned tutorials)
- Current tutorial $current\_tutorial_s \in T$ for each student $s \in S$
- Set $Wanted\_Tutorials_s \subseteq T$ of wanted tutorials for student $s \in S$
- Number of attendees $current\_load_t \in \mathbb{N}$ for each tutorial $t \in T$
- Capacity $capacity_t \in \mathbb{N}$ for each tutorial $t \in T$

Example

- $S = \{alice, bob\}$ and $T = \{mon, tue\}$
- $current\_tutorial_{alice} = mon$, $current\_tutorial_{bob} = tue$
- $Wanted\_Tutorials_{alice} = \{tue\}$, $Wanted\_Tutorials_{bob} = \{mon\}$
- $current\_load_{mon} = current\_load_{tue} = 1$
- $capacity_{mon} = capacity_{tue} = 1$

Modeling: Assigning Students to Tutorials

Example

- $S = \{alice, bob\}$ and $T = \{mon, tue\}$
- $current\_tutorial_{alice} = mon$, $current\_tutorial_{bob} = tue$
- $Wanted\_Tutorials_{alice} = \{tue\}$, $Wanted\_Tutorials_{bob} = \{mon\}$
- $current\_load_{mon} = current\_load_{tue} = 1$
- $capacity_{mon} = capacity_{tue} = 1$

Task

Find maximal reassignment of students to wanted tutorials such that:

1. Students can only be assigned to the current or a wanted tutorials
2. The capacity of tutorials may not be exceeded

Modeling: Assigning Students to Tutorials

Task

Find maximal reassignment of students to wanted tutorials such that:

1. Students can only be assigned to the current or a wanted tutorials
2. The capacity of tutorials may not be exceeded

Solution for the example: 2 happy students

1. *alice* switches to the tutorial *tue* (wanted) and *bob* to *mon* (wanted)
2.
   - Attendees *mon*: $\underbrace{1}_{\text{alice}} - \underbrace{1}_{\text{alice leaves}} + \underbrace{1}_{\text{bob comes}} \leq capacity_{mon} = 1$ ✓
   - Attendees *tue*: $\underbrace{1}_{\text{bob}} - \underbrace{1}_{\text{bob leaves}} + \underbrace{1}_{\text{alice comes}} \leq capacity_{tue} = 1$ ✓

It's a very small problem!?

Input from OSIRIS...

- 48 students and 22 tutorials
- each student indicated 1 to 7 wanted tutorials
- together with the pre-assigned tutorial there are 2 to 8 potential assignments to tutorials for each student

How many potential assignments of students to tutorials exist?

It's a very small problem!?

Input from OSIRIS...

- 48 students and 22 tutorials
- each student indicated 1 to 7 wanted tutorials
- together with the pre-assigned tutorial there are 2 to 8 potential assignments to tutorials for each student

How many potential assignments of students to tutorials exist?

$$\approx 1.74 \times 10^{21}$$

It's a very small problem!?

Input from OSIRIS...

- 48 students and 22 tutorials
- each student indicated 1 to 7 wanted tutorials
- together with the pre-assigned tutorial there are 2 to 8 potential assignments to tutorials for each student

How many potential assignments of students to tutorials exist?

$$\approx 1.74 \times 10^{21}$$

How would / should you solve this problem?

It's a very small problem!?

Input from OSIRIS...

- 48 students and 22 tutorials
- each student indicated 1 to 7 wanted tutorials
- together with the pre-assigned tutorial there are 2 to 8 potential assignments to tutorials for each student

How many potential assignments of students to tutorials exist?

$$\approx 1.74 \times 10^{21}$$

How would / should you solve this problem?

Integer Linear Programming!

Modeling: Assigning Students to Tutorials

Formulation as an Integer Linear Program

- Utilize variables $assignment_{s,t} \in \{0,1\}$ to indicate the assignment of students $s \in S$ to tutorials $t \in T$
  - $assignment_{s,t} = 0$ should be read as *student s is NOT assigned to tutorial t*
  - $assignment_{s,t} = 1$ should be read as *student s is assigned to tutorial t*

Integer Linear Programs

- continuous & integral variables
- *linear* constraints
- *linear* objective function

Solving Integer Linear Programs...

...is generally NP-complete.

Modeling: Assigning Students to Tutorials

Formulation as an Integer Linear Program

- Utilize variables $assignment_{s,t} \in \{0, 1\}$ to indicate the assignment of students $s \in S$ to tutorials $t \in T$
- Assignment of student $s \in S$ is valid:

assignment to current or wanted tutorial: $$\sum_{t \in (Wanted\_Tutorials_{\boldsymbol{s}} \cup \{current\_tutorial_{\boldsymbol{s}}\})} assignment_{s,t} = 1$$

forbidding assignments to other tutorials: $$\sum_{t \in T \setminus (Wanted\_Tutorials_{\boldsymbol{s}} \cup \{current\_tutorial_{\boldsymbol{s}}\})} assignment_{s,t} = 0$$

Modeling: Assigning Students to Tutorials

Formulation as an Integer Linear Program

- Utilize variables $assignment_{s,t} \in \{0,1\}$ to indicate the assignment of students $s \in S$ to tutorials $t \in T$
- Assignment of student $s \in S$ is valid:

assignment to current or wanted tutorial: $$\sum_{t \in (Wanted\_Tutorials_{\boldsymbol{s}} \cup \{current\_tutorial_{\boldsymbol{s}}\})} assignment_{s,t} = 1$$

forbidding assignments to other tutorials: $$\sum_{t \in T \setminus (Wanted\_Tutorials_{\boldsymbol{s}} \cup \{current\_tutorial_{\boldsymbol{s}}\})} assignment_{s,t} = 0$$

- For all tutorials $t \in T$ the capacity is not exceeded:

$$current\_load_t - \sum_{\substack{s \in S: \\ t = current\_tutorial_{\boldsymbol{s}} \\ t' \in T \setminus \{t\}}} assignment_{s,t'} + \sum_{\substack{s \in S: \\ t \neq current\_tutorial}} assignment_{s,t} \leq capacity_t$$

Modeling: Assigning Students to Tutorials

Formulation as an Integer Linear Program

- Assignment of student $s \in S$ is valid:

assignment to current or wanted tutorial: $$\sum_{t \in (Wanted\_Tutorials_{\mathbf{s}} \cup \{current\_tutorial_{\mathbf{s}}\})} assignment_{s,t} = 1$$

forbidding assignments to other tutorials: $$\sum_{t \in T \setminus (Wanted\_Tutorials_{\mathbf{s}} \cup \{current\_tutorial_{\mathbf{s}}\})} assignment_{s,t} = 0$$

- For all tutorials $t \in T$ the capacity is not exceeded:

$$current\_load_t - \underbrace{\sum_{\substack{s \in S: \\ t = current\_tutorial_{\mathbf{s}} \\ t' \in T \setminus \{t\}}} assignment_{s,t'}}_{\text{leaving tutorial} t} + \underbrace{\sum_{\substack{s \in S: \\ t \neq current\_tutorial}} assignment_{s,t}}_{\text{entering tutorial} t} \leq capacity_t$$

Modeling: Assigning Students to Tutorials

Formulation as an Integer Linear Program

- Assignment of student $s \in S$ is valid:

assignment to current or wanted tutorial: $$\sum_{t \in (Wanted\_Tutorials_s \cup \{current\_tutorial_s\})} assignment_{s,t} = 1$$

forbidding assignments to other tutorials: $$\sum_{t \in T \setminus (Wanted\_Tutorials_s \cup \{current\_tutorial_s\})} assignment_{s,t} = 0$$

- For all tutorials $t \in T$ the capacity is not exceeded:

$$current\_load_t - \sum_{\substack{s \in S: \\ t = current\_tutorial_s \\ t' \in T \setminus \{t\}}} assignment_{s,t'} + \sum_{\substack{s \in S: \\ t \neq current\_tutorial}} assignment_{s,t} \leq capacity_t$$

- Objective: $\max \sum_{s \in S, t \in Wanted\_Tutorials_s} assignment_{s,t}$

Modeling using GNU Math Programming Language[4]

Input from Florian's OSIRIS System

- Set of students S (who want to change their assigned tutorials)
- Current tutorial $current\_tutorial_s \in T$ for each student $s \in S$
- Set $Wanted\_Tutorials_s \subseteq T$ of wanted tutorials for student $s \in S$
- Number of attendees $current\_load_t \in \mathbb{N}$ for each tutorial $t \in T$
- Capacity $capacity_t \in \mathbb{N}$ for each tutorial $t \in T$

GMPL Code: tutorial-optimization.mod

```
set Students;
set Tutorials;
param current_tutorial{Students} symbolic in Tutorials;
set Wanted_Tutorials{Students} in Tutorials;
param current_load{Tutorials} > 0;
param capacity{Tutorials} > 0;
```

[4] ftp://ftp.gnu.org/gnu/glpk/, https://en.wikibooks.org/wiki/GLPK

Input Data

GMPL Data: tutorial-optimization.dat

```
set Students := studi_a, studi_b, studi_c, studi_d, studi_e, studi_f, studi_g, studi_h, st

set Tutorials := Monday_at_16_h_in_FH_303__26_30_, Tuesday_at_10_h_in_MAR_2.068__30_30_, M

param current_tutorial[*] := studi_a Monday_at_16_h_in_FH_316__23_30_
 studi_b Monday_at_12_h_in_MAR_0.007__29_30_
 studi_c Tuesday_at_8_h_in_MAR_0.010__29_30_
...

param current_load[*] := Monday_at_16_h_in_FH_303__26_30_ 26
 Tuesday_at_10_h_in_MAR_2.068__30_30_ 30
 Monday_at_12_h_in_MAR_0.007__29_30_ 29
 Tuesday_at_12_h_in_FH_314__30_30_ 30
...

param capacity[*] := Monday_at_16_h_in_FH_303__26_30_ 30
 Tuesday_at_10_h_in_MAR_2.068__30_30_ 30
 Monday_at_12_h_in_MAR_0.007__29_30_ 30
 Tuesday_at_12_h_in_FH_314__30_30_ 30
...

set Wanted_Tutorials[studi_a] := Monday_at_14_h_in_FH_316__29_30_ Monday_at_8_h_in_MAR_0.0
```

Modeling using GNU Math Programming Language

Variables

- Utilize variables $assignment_{s,t} \in \{0, 1\}$ to indicate the assignment of students $s \in S$ to tutorials $t \in T$
  - $assignment_{s,t} = 0$ should be read as *student s is NOT assigned to tutorial t*
  - $assignment_{s,t} = 1$ should be read as *student s is assigned to tutorial t*

GMPL Code: tutorial-optimization.mod

```
var assignment{Students, Tutorials} binary;
```

Modeling using GNU Math Programming Language

1st Constraint

$$\text{for all students } s \in S: \sum_{t \in (Wanted\_Tutorials_s \cup \{current\_tutorial_s\})} assignment_{s,t} = 1$$

GMPL Code: tutorial-optimization.mod

```
subject to only_assign_to_wanted_tutorials
{studi in Students}:
    sum {tut in Wanted_Tutorials[studi] union {current_tutorial[studi]}}
        assignment[studi, tut]
    == 1;
```

Modeling using GNU Math Programming Language

2nd Constraint

$$\text{for all students } s \in S: \sum_{t \in T \setminus (Wanted\_Tutorials_s \cup \{current\_tutorial_s\})} assignment_{s,t} = 0$$

GMPL Code: tutorial-optimization.mod

```
subject to forbid_assignment_to_unwanted_tutorials
{studi in Students}:
    sum{tut in Tutorials diff (Wanted_Tutorials[studi] union{current_tutorial[studi]})}
        assignment[studi, tut]
    == 0;
```

Modeling using GNU Math Programming Language

3rd Constraint: for all tutorials $t \in T$

$$current\_load_t - \sum_{\substack{s \in S: \\ t = current\_tutorial_s \\ t' \in T \setminus \{t\}}} assignment_{s,t'} + \sum_{\substack{s \in S: \\ t \neq current\_tutorial}} assignment_{s,t} \leq capacity_t$$

GMPL Code: tutorial-optimization.mod

```
subject to capacity_may_not_be_exceeded
{tut in Tutorials}:
    current_load[tut] +
    sum{studi in Students} (
        if current_tutorial[studi] == tut then (
            sum{tut2 in Wanted_Tutorials[studi] diff {tut}} (
              - assignment[studi, tut2]
            )
        )
        else (
            assignment[studi, tut]
        )
    )
    <= capacity[tut];
```

Modeling using GNU Math Programming Language

Objective

$$\max \sum_{s \in S, t \in Wanted\_Tutorials_s} assignment_{s,t}$$

GMPL Code: tutorial-optimization.mod

```
maximize number_of_studis_with_wanted_tutorial:
    sum{studi in Students, tut in Wanted_Tutorials[studi]}  assignment[studi, tut];



solve;

#iterate over all students and all tutorial which are not the current tutorial of
#the respective student
for {studi in Students, tut in Tutorials:  tut != current_tutorial[studi] }
{   #only print assignments which are set to true, i.e. novel tutorial assignments
    for {foo in {current_tutorial[studi]}: assignment[studi,tut] > 0}
printf "%s %s\n", studi, tut;
}

end;
```

Modeling using GNU Math Programming Language

Computing the optimal tutorial assignment via glpsol

```
glpsol --math --model tutorial-optimization.mod --data tutorial-optimization.dat
```

Example Execution[5]...

[5]Please find the files tutorial-optimization.mod and tutorial-optimization.dat at `https://matthias-rost.de/doc/rost-ilp-primer.tar.gz`

Real World Optimization: C-Course Planning

Problem

During the C-Course we have to schedule $\approx$ 25 tutors over ...

- 9 days and 8-10 time slots per day
- to 3 different kind of jobs (tutorials, pc-pools, grading) and
- assign the tutors to $\approx$ 30 different rooms.

Real World Optimization: C-Course Planning

Problem

During the C-Course we have to schedule $\approx$ 25 tutors over ...

- 9 days and 8-10 time slots per day
- to 3 different kind of jobs (tutorials, pc-pools, grading) and
- assign the tutors to $\approx$ 30 different rooms.

Constraints

- Tutor / room availability
- Work hours / happiness of tutors
- Breaks for tutors
- ‘Even’ distribution of tutorials / pc-pools over the day ...

Real World Optimization: C-Course Planning

Problem

During the C-Course we have to schedule $\approx$ 25 tutors over ...

- 9 days and 8-10 time slots per day
- to 3 different kind of jobs (tutorials, pc-pools, grading) and
- assign the tutors to $\approx$ 30 different rooms.

Constraints

- Tutor / room availability
- Work hours / happiness of tutors
- Breaks for tutors
- 'Even' distribution of tutorials / pc-pools over the day ...

Objectives

1. min. deviation from our plan
2. min. tutor hopping between buildings
3. min. external room usage (overhead)
4. max. 'tutor-room stability'

C-Course Planning: 1st phase ILP

First, there was a plan. . .

```
day = Converter.convertDateStringIntoDay("14.10.")
start = 12
result[Tasks.TUTORIUM][day] =   self.toDictionary([12,8,5,3,2,0], start, 1)
result[Tasks.UEBUNG_TEL][day] = self.toDictionary([2,4,6,6,4,4], start, 1)
result[Tasks.UEBUNG_MAR][day] = self.toDictionary([0,2,2,2,2,2], start, 1)
result[Tasks.KONTROLLE][day] =  self.toDictionary([0] * 6, start, 1)


day = Converter.convertDateStringIntoDay("15.10.")
start = 12
result[Tasks.TUTORIUM][day] =   self.toDictionary([12,8,5,3,2,0], start, 1)
result[Tasks.UEBUNG_TEL][day] = self.toDictionary([2,4,6,6,4,4], start, 1)
result[Tasks.UEBUNG_MAR][day] = self.toDictionary([0,2,2,2,2,2], start, 1)
result[Tasks.KONTROLLE][day] =  self.toDictionary([0] * 6, start, 1)
```

C-Course Planning: 1st phase ILP

First, there was a plan...

```
day = Converter.convertDateStringIntoDay("14.10.")
start = 12
result[Tasks.TUTORIUM][day] =   self.toDictionary([12,8,5,3,2,0], start, 1)
result[Tasks.UEBUNG_TEL][day] = self.toDictionary([2,4,6,6,4,4], start, 1)
result[Tasks.UEBUNG_MAR][day] = self.toDictionary([0,2,2,2,2,2], start, 1)
result[Tasks.KONTROLLE][day] =  self.toDictionary([0] * 6, start, 1)


day = Converter.convertDateStringIntoDay("15.10.")
start = 12
result[Tasks.TUTORIUM][day] =   self.toDictionary([12,8,5,3,2,0], start, 1)
result[Tasks.UEBUNG_TEL][day] = self.toDictionary([2,4,6,6,4,4], start, 1)
result[Tasks.UEBUNG_MAR][day] = self.toDictionary([0,2,2,2,2,2], start, 1)
result[Tasks.KONTROLLE][day] =  self.toDictionary([0] * 6, start, 1)
```

ILP Formulation: variables (1st phase)

We introduce a variable $assignment_{tutor,day,hour,task} \in \{0, 1\}$ to assign a specific tutor to a specific task (tutorial, pc-pool-MAR, pc-pool-TEL,grading) at a specific day and a specific hour.

C-Course Planning: 1st phase ILP

ILP Formulation: variables (1st phase)

We introduce a variable $assignment_{tutor,day,hour,task} \in \{0, 1\}$ to assign a specific tutor to a specific task (tutorial, pc-pool-MAR, pc-pool-TEL,grading) at a specific day and a specific hour.

Strict constraints

- Tutors don't work too much and not too little...

$$\forall tutor : \sum_{day,hour,task} assignment_{tutor,day,hour,task} \leq 1.9 \cdot weekly\_hours_{tutor}$$

$$\forall tutor : \sum_{day,hour,task} assignment_{tutor,day,hour,task} \geq 1.2 \cdot weekly\_hours_{tutor}$$

C-Course Planning: 1st phase ILP

ILP Formulation: variables (1st phase)

We introduce a variable $assignment_{tutor,day,hour,task} \in \{0, 1\}$ to assign a specific tutor to a specific task (tutorial, pc-pool-MAR, pc-pool-TEL,grading) at a specific day and a specific hour.

Strict constraints

- Tutors don't work too much and not too little. . .
- Tutors are assigned at most to a single task . . .

$$\forall tutor, day, hour : \quad \sum_{task} assignment_{tutor,day,hour,task} \leq 1$$

C-Course Planning: 1st phase ILP

ILP Formulation: variables (1st phase)

We introduce a variable $assignment_{tutor,day,hour,task} \in \{0,1\}$ to assign a specific tutor to a specific task (tutorial, pc-pool-MAR, pc-pool-TEL,grading) at a specific day and a specific hour.

Strict constraints

- Tutors don't work too much and not too little...
- Tutors are assigned at most to a single task ...
- Tutors have time ...

$$\forall tutor, day, hour, task : assignment_{tutor,day,hour,task} \leq availability_{tutor,day,hour}$$

C-Course Planning: 1st phase ILP

ILP Formulation: variables (1st phase)

We introduce a variable $assignment_{tutor,day,hour,task} \in \{0,1\}$ to assign a specific tutor to a specific task (tutorial, pc-pool-MAR, pc-pool-TEL,grading) at a specific day and a specific hour.

Strict constraints

- Tutors don't work too much and not too little...
- Tutors are assigned at most to a single task ...
- Tutors have time ...
- Tutors have at least one hour break in 4 hours...

$$\forall tutor, day, hour : \sum_{i=0}^{3} \sum_{task} assignment_{tutor,day,hour+i,task} \leq 3$$

C-Course Planning: 1st phase ILP

ILP Formulation: variables (1st phase)

We introduce a variable $assignment_{tutor,day,hour,task} \in \{0,1\}$ to assign a specific tutor to a specific task (tutorial, pc-pool-MAR, pc-pool-TEL,grading) at a specific day and a specific hour.

Strict constraints

- Tutors don't work too much and not too little...
- Tutors are assigned at most to a single task ...
- Tutors have time ...
- Tutors have at least one hour break in 4 hours...
- Tasks are upper bounded by plan...

$$\forall day, hour, task : \sum_{tutor} assignment_{tutor,day,hour,task} \leq plan_{day,hour,task}$$

C-Course Planning: 1st phase ILP

ILP Formulation: variables (1st phase)

We introduce a variable $assignment_{tutor,day,hour,task} \in \{0,1\}$ to assign a specific tutor to a specific task (tutorial, pc-pool-MAR, pc-pool-TEL,grading) at a specific day and a specific hour.

Objective of first phase: minimize deviation from plan

- Introduce variable $max\_deviation \geq 0$
- Introduce constraints ... $\forall day, hour, task$:

$$plan_{day,hour,task} - \sum_{tutor} assignment_{tutor,day,hour,task} \leq max\_deviation$$

- Consider the objective $\min max\_deviation$.

C-Course Planning: 1st phase ILP

Objective of first phase: minimize deviation from plan

- Introduce variable $max\_deviation \geq 0$
- Introduce constraints . . .
- Consider the objective min $max\_deviation$.

Having obtained an optimal solution to this problem . . .

- Try to understand where the plan is violated most (manually); try to adapt the plan
- Having obtained an assignment not violating the plan. . .
  1. minimize MAR-TEL hoppings of tutors while *not deviating from the plan*
  2. minimize external room usage, while *not deviating from the plan* and *with the TEL-MAR hoppings being bounded*
  3. consider tutor-room assignments in the last step: approx. 10^{80} potential tutor-room mappings

C-Course Planning: 1st phase ILP

Implementation is simple

```
def createConstraint_UniqueTaskAtAGivenTime(self):
  print "  ..constructing UniqueTaskAtAGivenTime"
  for tutor in self.data.tutoren.keys():
    for day in Days:
      for hour in Hours(day):
        expr = LinExpr([(1.0, self.scheduleEntry[tutor][day][hour][task]) for task in Tasks])
        constrName = "UniqueTaskAtAGivenTime" + "_".join([tutor, str(day), str(hour)])
        self.model.addConstr(expr, GRB.LESS_EQUAL, 1.0, name = constrName)
```

Applying ILPs in Research

My Applications...

Wide-area analytics	Computing near-optimal aggregation / multicast trees.	Rost and Schmid [2013]
Temporal scheduling of VNets	Improving the performance of naive approaches by orders of magnitude.	Rost et al. [2014]
Pathlet-stitching at IXPs	Baseline for evaluating the quality of our online heuristics.	Kotronis et al. [2016]
Network update problem	Finding update schedules quickly or proving that no update schedule exists.	Ludwig et al. [2016]
VC embedding in DCs	Evaluating resource savings by using direct interconnections instead of centralized logical switches in the virtual cluster abstraction.	Rost et al. [2015]
Middlebox deployment	Understanding the performance of greedy algorithms.	Lukovszki et al. [2016]
Service chain embeddings	Finding provably good solutions in polynomial time.	Even et al. [2016] Rost and Schmid [2016]

A complex Linear Program: > 1 months of work

Linear Program 2: Novel Decomposable Formulation for VNEP

$$\max \sum_{r \in \mathcal{R}} b_r \cdot x_r \tag{7}$$

$$\sum_{u \in V_S^{r, r_r}} f^+_{r, r_r, u} = x_r \qquad \forall r \in \mathcal{R} \tag{8}$$

$$\sum_{w \in V_{S,t}^{C_k}} f_{r,\ (u^+_{r,i}, u^{i,j}_{r,w})} = f^+_{r,i,u} \qquad \forall r \in \mathcal{R}, C_k \in \mathcal{C}_r,\ (i,j) \in E_r^{C_k, B_1}, i = s_r^{C_k}, u \in V_S^{r,i} \tag{9}$$

$$f_{r,\ (u^+_{r,i}, u^{i,j}_{r,w})} - f_{r,\ (u^+_{r,i}, u^{i,j'}_{r,w})} = 0 \qquad \forall r \in \mathcal{R}, C_k \in \mathcal{C}_r,\ (i,j) \in E_r^{C_k, B_1},\ (i,j') \in E_r^{C_k, B_2}, i = s_r^{C_k}, w \in V_{S,t}^{C_k} \tag{10}$$

$$f_{r,\ (u^+_{r,i}, u^{i,j}_{r})} = f^+_{r,i,u} \qquad \forall r \in \mathcal{R}, P_k \in \mathcal{P}_r,\ (i,j) \in E_r^{P_k}, i = s_r^{P_k}, u \in V_S^{r,i} \tag{11}$$

$$\sum_{e \in \delta^+(u)} f_{r,e} - \sum_{e \in \delta^-(u)} f_{r,e} = 0 \qquad \forall r \in \mathcal{R}, u \in V_{r,\text{flow}}^{\text{ext,SCG}} \tag{12}$$

$$f_{r,\ (w^{i,j}_{r,w}, w^-_{r,j})} = f^+_{r,j,w} \qquad \forall r \in \mathcal{R}, C_k \in \mathcal{C}_r,\ (i,j) \in E_r^{C_k, B_1}, j = t_r^{C_k}, w \in V_S^{r,j} \tag{13}$$

$$f_{r,\ (u^{i,j}_{r}, u^-_{r,j})} = f^+_{r,j,u} \qquad \forall r \in \mathcal{R}, P_k \in \mathcal{P}_r,\ (i,j) \in E_r^{P_k}, j = t_r^{P_k}, u \in V_S^{r,j} \tag{14}$$

$$\sum_{w \in V_{S,t}^{C_k}} f_{r,\ (u^{i,j}_{r,w}, u^{j,l}_{r,w})} = f^+_{r,j,u} \qquad \forall r \in \mathcal{R}, C_k \in \mathcal{C}_r, j \in \mathcal{B}_r^{C_k}, (i,j), (j,l) \in E_r^{C_k}, u \in V_S^{r,j} \tag{15}$$

$$\sum_{(e,i) \in E_{r,\tau,u}^{\text{ext,SCG}}} d_r(i) \cdot f_{r,e} + \sum_{\substack{i \in V_r^{\pm} \setminus \mathcal{B}_r^{C_k} \\ \tau_r(i) = \tau}} d_r(i) \cdot f^+_{r,i,u} = l_{r,\tau,u} \qquad \forall r \in \mathcal{R}, (\tau, u) \in R_S^V \tag{16}$$

$$\sum_{(e,i,j) \in E_{r,u,v}^{\text{ext,SCG}}} d_r(i,j) \cdot f_{r,e} = l_{r,u,v} \qquad \forall r \in \mathcal{R}, (u,v) \in E_S \tag{17}$$

$$\sum_{r \in \mathcal{R}} l_{r,x,y} \leqslant d_S(x,y) \qquad \forall (x,y) \in R_S \tag{18}$$

$$x_r \in [0,1] \qquad \forall r \in \mathcal{R} \tag{19}$$

$$f^+_{r,i,u} \in [0,1] \qquad \forall i \in V_r^{\pm} \tag{20}$$

$$f_{r,e} \in [0,1] \qquad \forall r \in \mathcal{R}, e \in E_{r,\text{flow}}^{\text{ext,SCG}} \tag{21}$$

$$l_{r,x,y} \geqslant 0 \qquad \forall r \in \mathcal{R}, (x,y) \in R_S \tag{22}$$

Summary

Bottomline

- The language of Integer Linear Programming is math, but …
  …it is simple math: "$+ \ - \ x \ \leq = \geq$"
- Modeling the problem – and defining the right notation – is hard; implementing it is generally not…
  - Solvers: Gurobi, CPLEX, XPRESS, SCIP, …
  - Mathematical modeling tools: AMPL, GMPL, ZMPL, Julia, …
- Different use cases in research:
  - baselines for heuristic algorithms
  - solving large planning problems in reasonable time (offline)
  - deriving approximation algorithms from ILPs

References I

Guy Even, Matthias Rost, and Stefan Schmid. An Approximation Algorithm for Path Computation and Function Placement in SDNs. In *23rd International Colloquium on Structural Information and Communication Complexity (SIROCCO)*. Springer, 2016.

Vasileios Kotronis, Rowan Klöti, Matthias Rost, Panagiotis Georgopoulos, Bernhard Ager, Stefan Schmid, and Xenofontas Dimitropoulos. Stitching inter-domain paths over ixps. In *ACM SIGCOMM Symposium on Software Defined Networking Research*, March 2016.

Arne Ludwig, Szymon Dudyzc, Matthias Rost, and Stefan Schmid. Transiently secure network updates. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, SIGMETRICS '16, New York, NY, USA, 2016. ACM. doi: 10.1145/2896377.2901476.

References II

Tamás Lukovszki, Matthias Rost, and Stefan Schmid. It's a match!: Near-optimal and incremental middlebox deployment. *SIGCOMM Comput. Commun. Rev.*, 46(1):30–36, January 2016. doi: 10.1145/2875951.2875956.

Jiri Matousek and Bernd Gärtner. *Understanding and using linear programming*. Springer Science & Business Media, 2007.

Matthias Rost and Stefan Schmid. Virtucast: Multicast and aggregation with in-network processing. In Roberto Baldoni, Nicolas Nisse, and Maarten Steen, editors, *Principles of Distributed Systems*, volume 8304 of *Lecture Notes in Computer Science*, pages 221–235. Springer International Publishing, 2013. doi: 10.1007/978-3-319-03850-6_16.

Matthias Rost and Stefan Schmid. Service chain and virtual network embeddings: Approximations using randomized rounding. *CoRR*, abs/1604.02180, 2016.

References III

Matthias Rost, Stefan Schmid, and Anja Feldmann. It's about time: On optimal virtual network embeddings under temporal flexibilities. In *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pages 17–26. IEEE, May 2014. doi: 10.1109/IPDPS.2014.14.

Matthias Rost, Carlo Fuerst, and Stefan Schmid. Beyond the stars: Revisiting virtual cluster embeddings. *SIGCOMM Comput. Commun. Rev.*, 45(3):12–18, July 2015. doi: 10.1145/2805789.2805792.