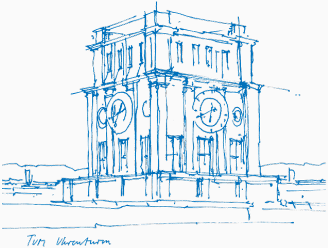
Computer Course Linear Programming

Introduction to Gurobipy



Stefan Kober

28-29 October 2020

Technical University of Munich

Organizational Things

What to expect

What this course offers:

- praxis-oriented introduction to python and gurobipy
- lots of examples
- preparation for further lectures, case studies and theses

What this course does not offer:

- detailed installation instructions
- the time needed to become an expert in python and gurobipy

Schedule

- Wednesday:
  - Introduction to Python
  - Introduction to Gurobi
- Thursday:
  - Features Python (advanced input and output methods)
  - Features Gurobi (advanced variable types and output interpretation)

Schedule

10:15	first slot
11:45	lunch break
13:15	second slot
14:45	coffee break
15:15	third slot

Work in teams!

Outlook

Structure of Gurobi

Basics

Linear Programming

Modelling

Output Interpretation

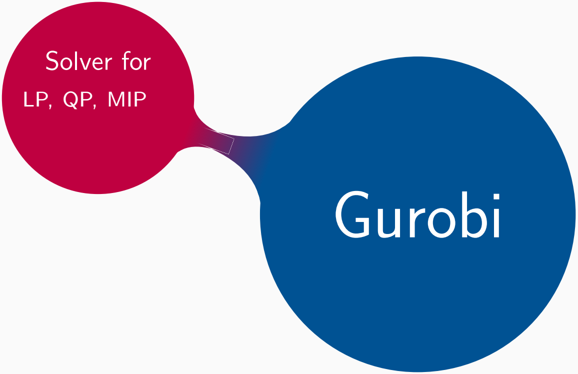
Advanced Input Methods

Advanced Gurobi Datatypes

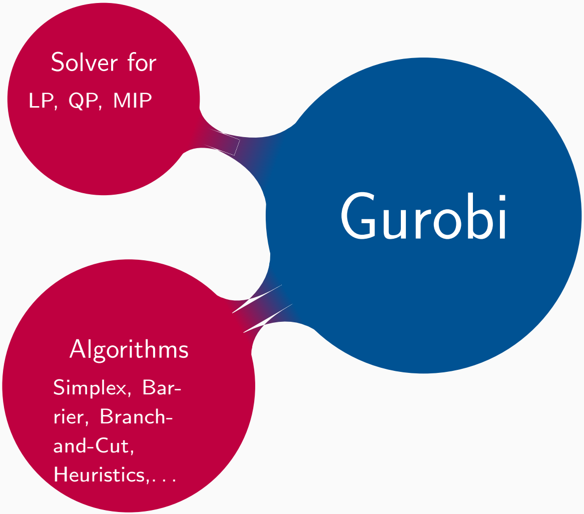
Visualization

Structure of Gurobi

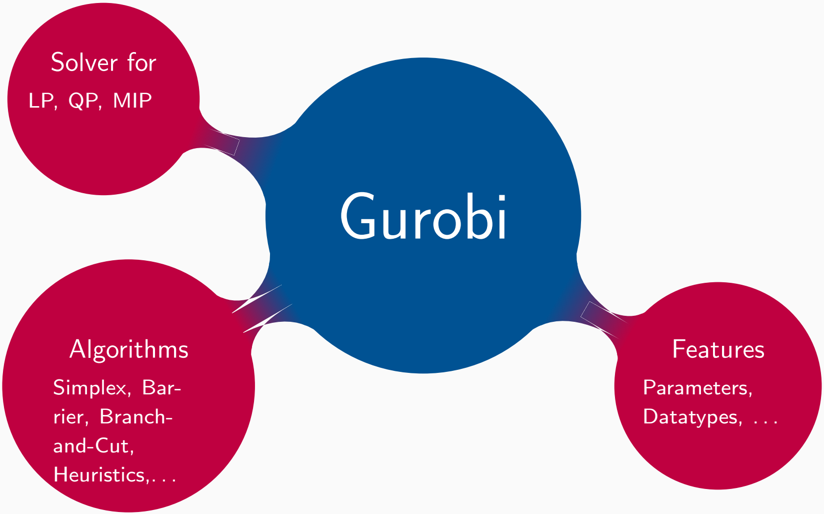
What is Gurobi?



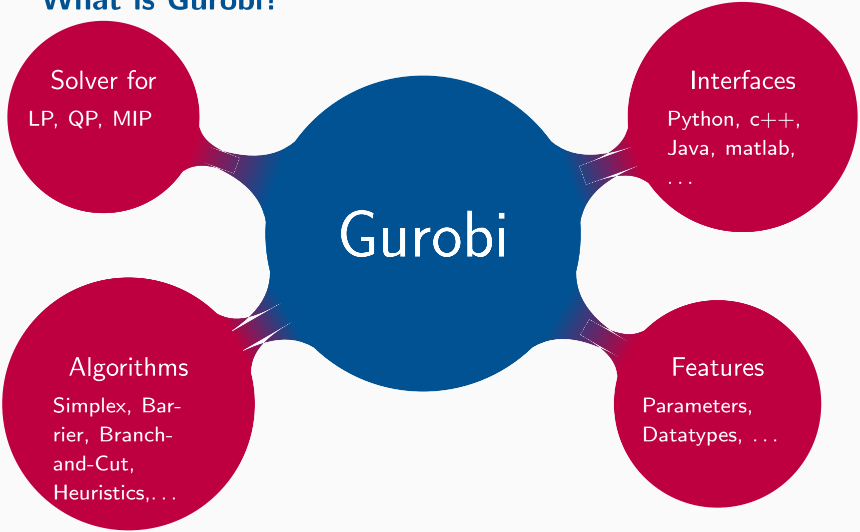
What is Gurobi?



What is Gurobi?



What is Gurobi?

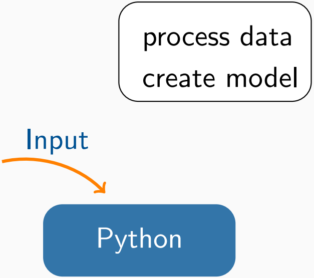


How can we use Gurobi?

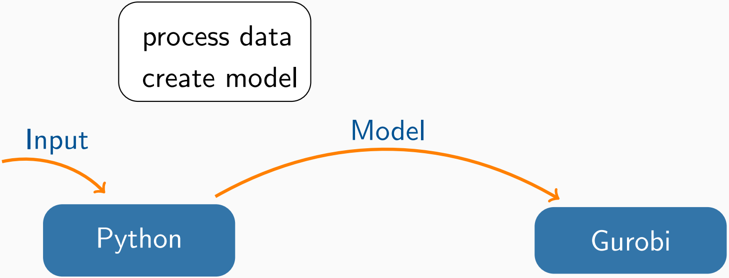
Input

Python

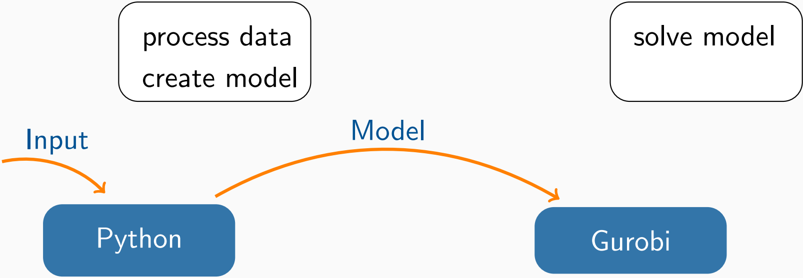
How can we use Gurobi?



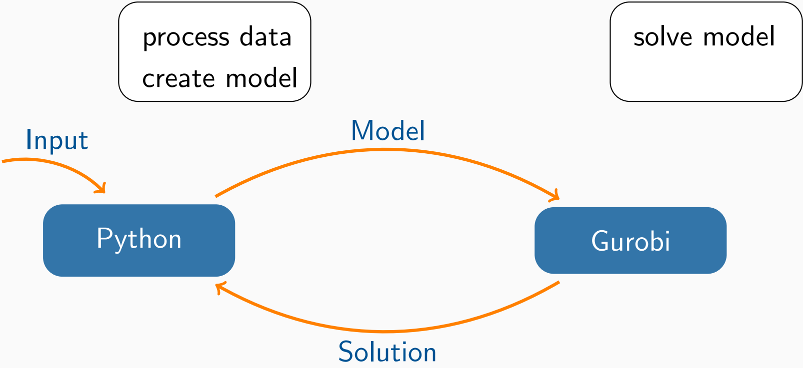
How can we use Gurobi?



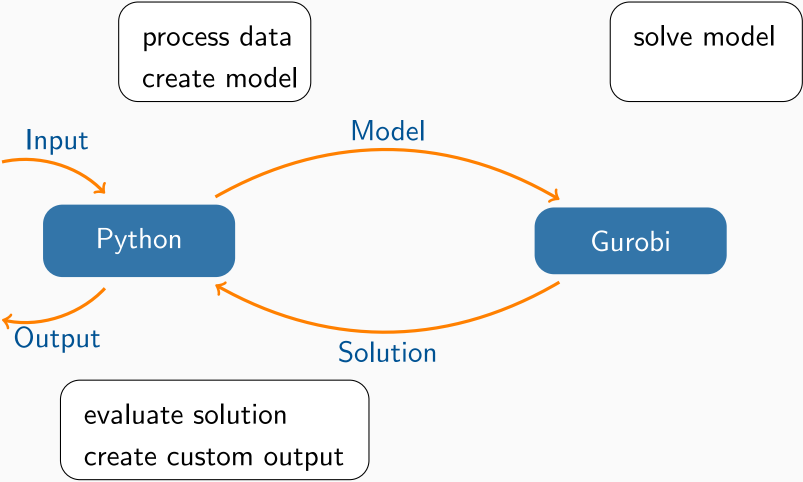
How can we use Gurobi?



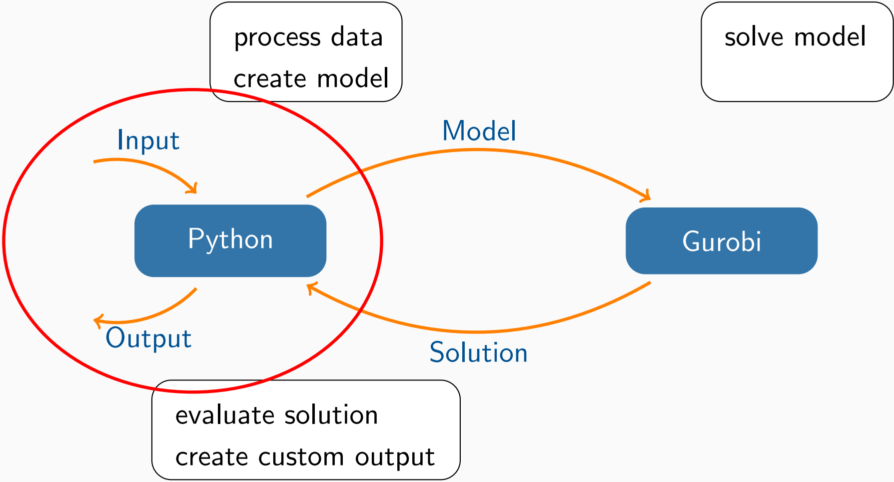
How can we use Gurobi?



How can we use Gurobi?



How can we use Gurobi?



Credits

The materials used in this course have been developed and improved by

- Melanie Herzog
- Anja Kirschbaum
- Fabian Klemm
- Michael Ritter
- Matthias Silbernagel
- Paul Stursberg
- Stefan Kober

Basics

Python

- open source
- most popular programming language
- object-oriented, procedural, functional
- interactive
- easy to learn

Advantages

- high-level
  - direct interpretation of objects
  - readable and accessible
- many useful libraries (graphs, visualization, computations, data management,...)

Limits

- slow running times
- somewhat restricted
- possibly not best choice for large object oriented project

Basic Knowledge

- Datatypes
  - integer, float, string
  - list, tuple, dict, set
- Indentation
- Output
  - print
  - formatted print
- Imports

Linear Programming

What is a linear program?

$$\begin{aligned} \min c^\top x \quad & \text{s.t.} \\ Ax \leq b \end{aligned}$$

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 2 & 1 & 1 \\ -2 & -2 & 0 \\ -2 & 0 & -3 \end{pmatrix}, \quad b = \begin{pmatrix} 4 \\ 7 \\ 1 \\ -1 \\ -1 \end{pmatrix}, \quad c = \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix}$$

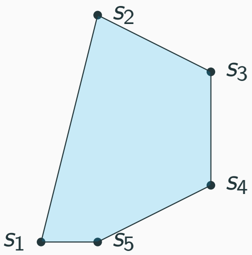
What is a linear program?

$$\begin{aligned} \min c^\top x \quad \text{s.t.} \\ Ax \leq b \end{aligned}$$

- set of variables x
- set of linear constraints $Ax \leq b$
- linear objective function $\min c^\top x$

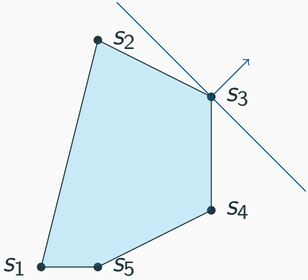
What is a linear program?

$$\min c^\top x \quad \text{s.t.}$$
$$Ax \leq b$$



What is a linear program?

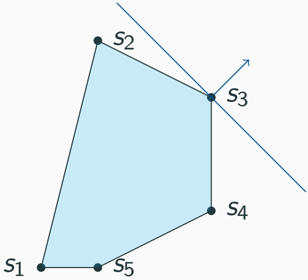
$$\min c^\top x \quad \text{s.t.}$$
$$Ax \le b$$



How to solve a linear program?

The Simplex Algorithm

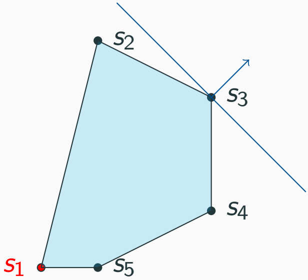
- Find a feasible solution
- Travel along improving edges
- Terminate at optimal solution



How to solve a linear program?

The Simplex Algorithm

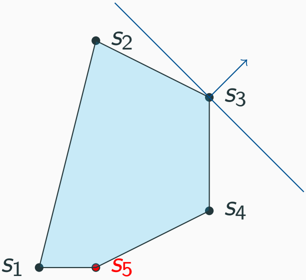
- Find a feasible solution
- Travel along improving edges
- Terminate at optimal solution



How to solve a linear program?

The Simplex Algorithm

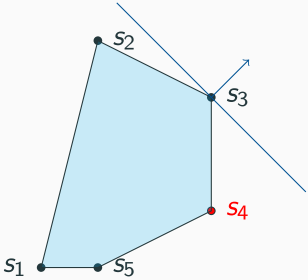
- Find a feasible solution
- Travel along improving edges
- Terminate at optimal solution



How to solve a linear program?

The Simplex Algorithm

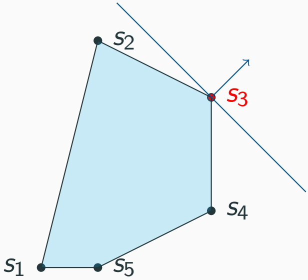
- Find a feasible solution
- Travel along improving edges
- Terminate at optimal solution



How to solve a linear program?

The Simplex Algorithm

- Find a feasible solution
- Travel along improving edges
- Terminate at optimal solution



How to solve a linear program?

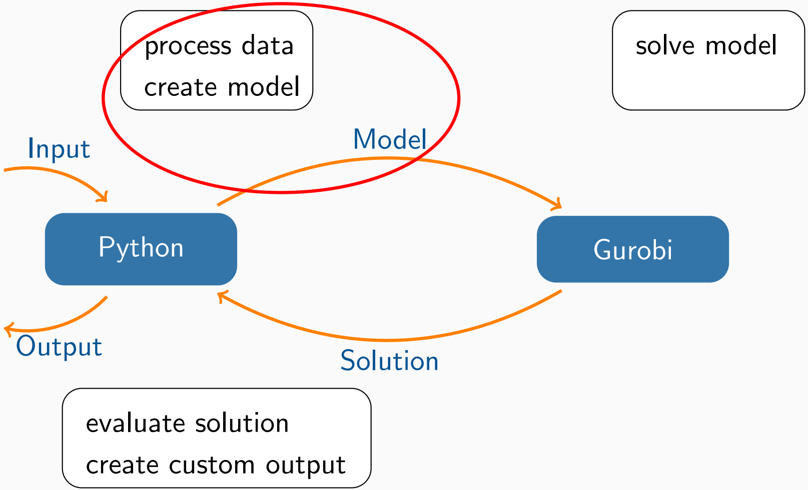
The Simplex Algorithm

- Find a feasible solution
- Travel along improving edges
- Terminate at optimal solution

Good News:
Gurobi does that for us

Modelling

Modelling


process data
create model
solve model
Input
Model
Python
Gurobi
Output
Solution
evaluate solution
create custom output

Problem: Crude Oil Refinement 1

10l crude oil

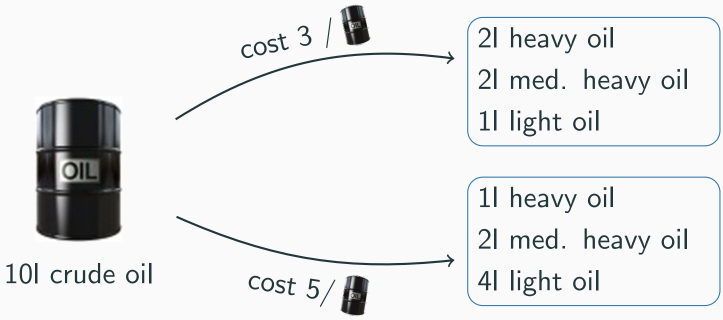
Problem: Crude Oil Refinement 1

cost 3 / 

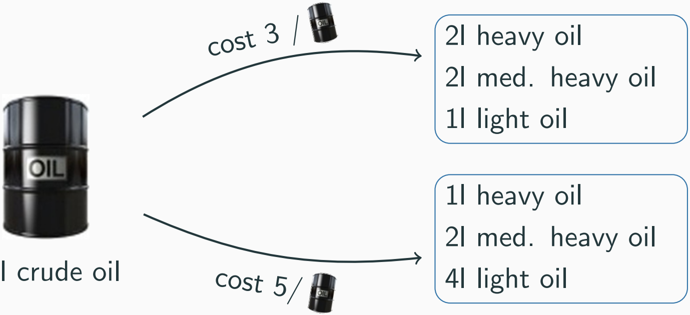
2l heavy oil
2l med. heavy oil
1l light oil

10l crude oil

Problem: Crude Oil Refinement 1

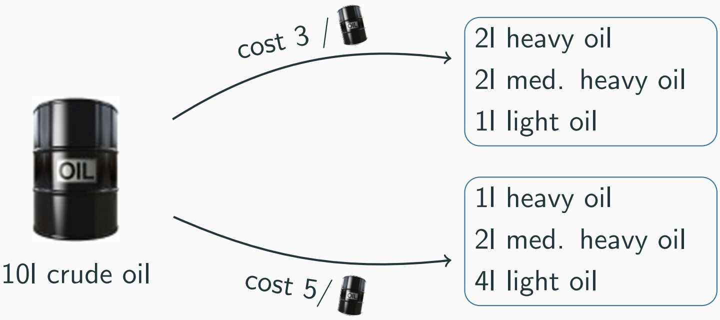


Problem: Crude Oil Refinement 1



demand: 3l heavy oil, 5l med. heavy oil, 4l light oil

Problem: Crude Oil Refinement 1



demand: 3l heavy oil, 5l med. heavy oil, 4l light oil

objective: minimize cost

LP Model Problem 1

$$\min 3x_1 + 5x_2$$

s.t.

$$\begin{aligned} 2x_1 + 1x_2 &\geq 3 \\ 2x_1 + 2x_2 &\geq 5 \\ 1x_1 + 4x_2 &\geq 4 \\ x_1, x_2 &\geq 0 \end{aligned}$$

LP Model Problem 1

$$\min 3x_1 + 5x_2$$

s.t.

$$\begin{aligned} 2x_1 + 1x_2 &\geq 3 \\ 2x_1 + 2x_2 &\geq 5 \\ 1x_1 + 4x_2 &\geq 4 \\ x_1, x_2 &\geq 0 \end{aligned}$$

LP Model Problem 1

$$\min 3x_1 + 5x_2$$

s.t.

$$2x_1 + 1x_2 \geq 3$$
$$2x_1 + 2x_2 \geq 5$$
$$1x_1 + 4x_2 \geq 4$$
$$x_1, x_2 \geq 0$$

LP Model Problem 1

$$\min 3x_1 + 5x_2$$

s.t.

$$
\begin{aligned}
2x_1 + 1x_2 &\geq 3 \\
2x_1 + 2x_2 &\geq 5 \\
1x_1 + 4x_2 &\geq 4 \\
x_1, x_2 &\geq 0
\end{aligned}
$$

LP Model Problem 1

$$\min 3x_1 + 5x_2$$

s.t.

$$\begin{aligned} 2x_1 + 1x_2 &\geq 3 \\ 2x_1 + 2x_2 &\geq 5 \\ 1x_1 + 4x_2 &\geq 4 \\ x_1, x_2 &\geq 0 \end{aligned}$$

LP Model Problem 1

$$\min 3x_1 + 5x_2$$

s.t.

$$\begin{aligned} 2x_1 + 1x_2 &\geq 3 \\ 2x_1 + 2x_2 &\geq 5 \\ 1x_1 + 4x_2 &\geq 4 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Initialize gurobipy and create set of variables x

```
from gurobipy import *

# Create a new model
m = Model()

# Create variables
x = m.addVar(vtype=GRB.CONTINUOUS)
y = m.addVar(vtype=GRB.CONTINUOUS)
```

Initialize gurobipy and create set of variables x

```
from gurobipy import *

# Create a new model
m = Model()

# Create variables
x = m.addVar(vtype=GRB.CONTINUOUS)
y = m.addVar(vtype=GRB.CONTINUOUS)
```

Variable types

- GRB.CONTINUOUS $(-\infty, \infty)$
- GRB.BINARY $\{0, 1\}$
- GRB.INTEGER $\{0, 1, 2, \ldots\}$
- GRB.SEMICONT $\{0\} \cup (a, b)$
- GRB.SEMIINT $\{0\} \cup (a, b) \cap \mathbb{Z}$

Variable types

- GRB.CONTINUOUS $(-\infty, \infty)$
- GRB.BINARY $\{0, 1\}$
- GRB.INTEGER $\{0, 1, 2, \ldots\}$
- GRB.SEMICONT $\{0\} \cup (a, b)$
- GRB.SEMIINT $\{0\} \cup (a, b) \cap \mathbb{Z}$

Variable types

- GRB.CONTINUOUS $(-\infty, \infty)$
- GRB.BINARY $\{0, 1\}$
- GRB.INTEGER $\{0, 1, 2, \ldots\}$
- GRB.SEMICONT $\{0\} \cup (a, b)$
- GRB.SEMIINT $\{0\} \cup (a, b) \cap \mathbb{Z}$

Variable types

- GRB.CONTINUOUS $(-\infty, \infty)$
- GRB.BINARY $\{0, 1\}$
- GRB.INTEGER $\{0, 1, 2, \ldots\}$
- GRB.SEMICONT $\{0\} \cup (a, b)$
- GRB.SEMIINT $\{0\} \cup (a, b) \cap \mathbb{Z}$

Variable types

- GRB.CONTINUOUS $(-\infty, \infty)$
- GRB.BINARY $\{0, 1\}$
- GRB.INTEGER $\{0, 1, 2, \ldots\}$
- GRB.SEMICONT $\{0\} \cup (a, b)$
- GRB.SEMIINT $\{0\} \cup (a, b) \cap \mathbb{Z}$

Variable types

- GRB.CONTINUOUS $(-\infty, \infty)$
- GRB.BINARY $\{0, 1\}$
- GRB.INTEGER $\{0, 1, 2, \ldots\}$
- GRB.SEMICONT $\{0\} \cup (a, b)$
- GRB.SEMIINT $\{0\} \cup (a, b) \cap \mathbb{Z}$

Initialize gurobipy and create set of variables x

```
from gurobipy import *

# Create a new model
m = Model()

# Create variables
x = m.addVar(vtype=GRB.CONTINUOUS)
y = m.addVar(vtype=GRB.CONTINUOUS)
```

Add Variables

```
addVar(lb=0,ub=GRB.INFINITY,obj=0.0,
          vtype=GRB.CONTINUOUS,name="")
```

- *lb*, *ub*: variable lower and upper bound
- *obj*: coefficient of the linear objective function
- *vtype*: variable type
- *name*: name for further referencing

Add Variables

```
addVars(indices,lb=0,ub=GRB.INFINITY,obj=0.0,
        vtype=GRB.CONTINUOUS,name="")
```

- *lb*, *ub*: variable lower and upper bound
- *obj*: coefficient of the linear objective function
- *vtype*: variable type
- *name*: name for further referencing
- *indices*: integer, range, list or dictionary used to generate set of variables

Create set of linear constraints $Ax \geq b$

```
# Add constraints
c1 = m.addConstr(2*x+y>=3)
c2 = m.addConstr(2*x+2*y>=5)
c3 = m.addConstr(x+4*y>=4)
c4 = m.addConstr(x>=0)
c5 = m.addConstr(y>=0)
```

Create set of linear constraints $Ax \geq b$

```
# Add constraints
c1 = m.addConstr(2*x+y>=3)
c2 = m.addConstr(2*x+2*y>=5)
c3 = m.addConstr(x+4*y>=4)
c4 = m.addConstr(x>=0)
c5 = m.addConstr(y>=0)
```

Add Constraints

Basic form:

```
m.addConstr(LinExpr>=a)
```

Add Constraints

Basic form:

```
m.addConstr(LinExpr>=a)
```

Linear expressions can be created by:

- $le = 2 * x + 3 * y$
- $le = x.prod([2, 3])$
- $le = x.sum()$
- $le = quicksum([2 * x, 3 * y])$

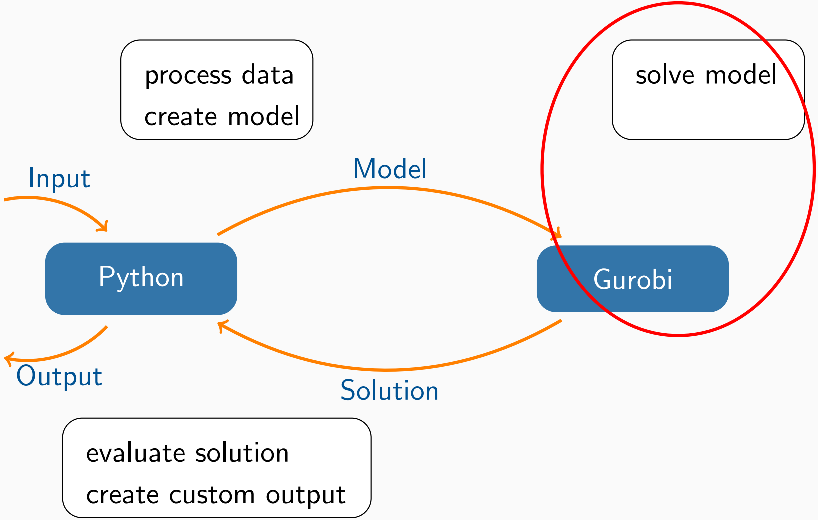
Set linear objective function $\min c^\top x$ and optimize the model

```
# Set objective function
m.setObjective(3*x+5*y,GRB.MINIMIZE)

# Optimize model
m.optimize()
```

Output Interpretation

Gurobi - LP solver



Presolve

```
Optimize a model with 1500 rows,
   2250000 columns and 4497000 nonzeros
Coefficient statistics:
  Matrix range      [1e+00, 1e+00]
  Objective range   [1e+00, 1e+02]
  Bounds range      [0e+00, 0e+00]
  RHS range         [1e+00, 2e+01]
Presolve removed 0 rows and 1611 columns
Presolve time: 4.37s
Presolved: 1500 rows, 2248389 columns, 4496778 nonzeros
```

Presolve - Example

$$
\begin{aligned}
x_1 + x_2 + x_3 &\geq 15 \\
x_1 &\leq 7 \\
x_2 &\leq 3 \\
x_3 &\leq 5
\end{aligned}
$$

Presolve - Example

$$
\begin{aligned}
x_1 + x_2 + x_3 &\geq 15 \\
x_1 &\leq 7 \\
x_2 &\leq 3 \\
x_3 &\leq 5
\end{aligned}
$$

delete constraint and variables in order to reduce the problem

LP methods

- 3 different methods
  - primal/dual simplex
    - robust
    - easy to restart after model modification
  - barrier
    - can be run on multiple cores
- concurrent optimization

Concurrent Optimization

- run simplex *and* barrier at the same time
- first one to finish reports solution
- use multiple cores
- fastest choice for general model

Primal Simplex

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	-1.2573140e+02	2.968000e+03	5.593346e+11	7s
3297	1.2166285e+05	0.000000e+00	6.532640e+07	10s
4619	8.3232592e+04	0.000000e+00	1.162234e+08	15s
7000	5.2154173e+04	0.000000e+00	4.234078e+06	25s
9189	3.7601369e+04	0.000000e+00	1.177763e+07	35s
10706	3.0986489e+04	0.000000e+00	2.645246e+06	45s
12019	2.7488411e+04	0.000000e+00	4.758955e+06	56s
13844	2.3646778e+04	0.000000e+00	3.840044e+05	65s
15403	2.1713789e+04	0.000000e+00	3.055039e+04	75s
17347	2.1105977e+04	0.000000e+00	1.777696e+01	85s
17419	2.1108270e+04	0.000000e+00	0.000000e+00	91s

Solved in 17419 iterations and 90.82 seconds
Optimal objective 2.110826998e+04

Primal Simplex

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	-1.2573140e+02	2.968000e+03	5.593346e+11	7s
3297	1.2166285e+05	0.000000e+00	6.532640e+07	10s
4619	8.3232592e+04	0.000000e+00	1.162234e+08	15s
7000	5.2154173e+04	0.000000e+00	4.234078e+06	25s
9189	3.7601369e+04	0.000000e+00	1.177763e+07	35s
10706	3.0986489e+04	0.000000e+00	2.645246e+06	45s
12019	2.7488411e+04	0.000000e+00	4.758955e+06	56s
13844	2.3646778e+04	0.000000e+00	3.840044e+05	65s
15403	2.1713789e+04	0.000000e+00	3.055039e+04	75s
17347	2.1105977e+04	0.000000e+00	1.777696e+01	85s
17419	2.1108270e+04	0.000000e+00	0.000000e+00	91s

Solved in 17419 iterations and 90.82 seconds
Optimal objective 2.110826998e+04

Primal Simplex

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	-1.2573140e+02	2.968000e+03	5.593346e+11	7s
3297	1.2166285e+05	0.000000e+00	6.532640e+07	10s
4619	8.3232592e+04	0.000000e+00	1.162234e+08	15s
7000	5.2154173e+04	0.000000e+00	4.234078e+06	25s
9189	3.7601369e+04	0.000000e+00	1.177763e+07	35s
10706	3.0986489e+04	0.000000e+00	2.645246e+06	45s
12019	2.7488411e+04	0.000000e+00	4.758955e+06	56s
13844	2.3646778e+04	0.000000e+00	3.840044e+05	65s
15403	2.1713789e+04	0.000000e+00	3.055039e+04	75s
17347	2.1105977e+04	0.000000e+00	1.777696e+01	85s
17419	2.1108270e+04	0.000000e+00	0.000000e+00	91s

Solved in 17419 iterations and 90.82 seconds
Optimal objective 2.110826998e+04

Primal Simplex

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	-1.2573140e+02	2.968000e+03	5.593346e+11	7s
3297	1.2166285e+05	0.000000e+00	6.532640e+07	10s
4619	8.3232592e+04	0.000000e+00	1.162234e+08	15s
7000	5.2154173e+04	0.000000e+00	4.234078e+06	25s
9189	3.7601369e+04	0.000000e+00	1.177763e+07	35s
10706	3.0986489e+04	0.000000e+00	2.645246e+06	45s
12019	2.7488411e+04	0.000000e+00	4.758955e+06	56s
13844	2.3646778e+04	0.000000e+00	3.840044e+05	65s
15403	2.1713789e+04	0.000000e+00	3.055039e+04	75s
17347	2.1105977e+04	0.000000e+00	1.777696e+01	85s
17419	2.1108270e+04	0.000000e+00	0.000000e+00	91s

Solved in 17419 iterations and 90.82 seconds
Optimal objective 2.110826998e+04

Primal Simplex

Iteration	Objective	Primal Inf.	Dual Inf.	Time
0	-1.2573140e+02	2.968000e+03	5.593346e+11	7s
3297	1.2166285e+05	0.000000e+00	6.532640e+07	10s
4619	8.3232592e+04	0.000000e+00	1.162234e+08	15s
7000	5.2154173e+04	0.000000e+00	4.234078e+06	25s
9189	3.7601369e+04	0.000000e+00	1.177763e+07	35s
10706	3.0986489e+04	0.000000e+00	2.645246e+06	45s
12019	2.7488411e+04	0.000000e+00	4.758955e+06	56s
13844	2.3646778e+04	0.000000e+00	3.840044e+05	65s
15403	2.1713789e+04	0.000000e+00	3.055039e+04	75s
17347	2.1105977e+04	0.000000e+00	1.777696e+01	85s
17419	2.1108270e+04	0.000000e+00	0.000000e+00	91s

```
Solved in 17419 iterations and 90.82 seconds
Optimal objective  2.110826998e+04
```

Primal Dual Relations

Primal:

$$\begin{aligned} \min c^\top x \quad & \text{s.t.} \\ Ax \leq b & \end{aligned}$$

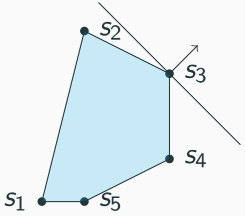
Dual:

$$\begin{aligned} \max b^\top y \quad & \text{s.t.} \\ A^\top y = c & \\ y \geq 0 & \end{aligned}$$

Primal Dual Relations

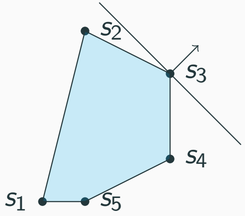
Primal:

$$\min c^\top x \quad \text{s.t.}$$
$$Ax \le b$$



Dual:

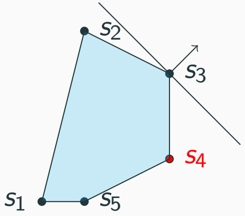
$$\max b^\top y \quad \text{s.t.}$$
$$A^\top y = c$$
$$y \ge 0$$



Primal Dual Relations

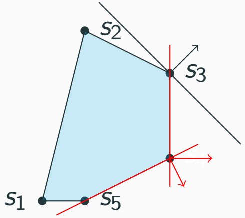
Primal:

$$\min c^\top x \quad \text{s.t.}$$
$$Ax \le b$$



Dual:

$$\max b^\top y \quad \text{s.t.}$$
$$A^\top y = c$$
$$y \ge 0$$



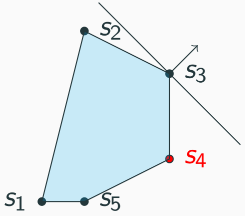
Primal Dual Relations

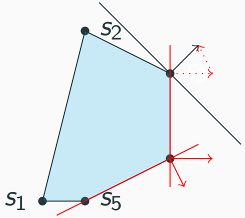
Primal:

$$\min c^\top x \quad \text{s.t.}$$
$$Ax \le b$$

Dual:

$$\max b^\top y \quad \text{s.t.}$$
$$A^\top y = c$$
$$y \ge 0$$

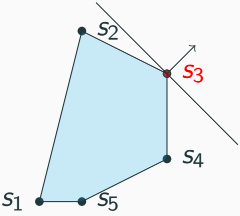




Primal Dual Relations

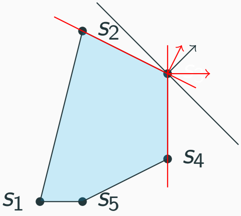
Primal:

$$\min c^\top x \quad \text{s.t.}$$
$$Ax \leq b$$



Dual:

$$\max b^\top y \quad \text{s.t.}$$
$$A^\top y = c$$
$$y \geq 0$$



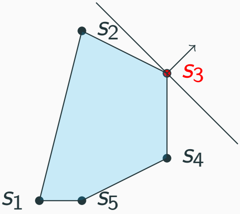
Primal Dual Relations

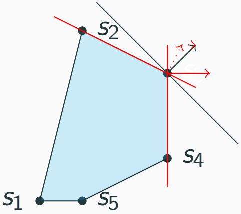
Primal:

$$\min c^\top x \quad \text{s.t.}$$
$$Ax \leq b$$

Dual:

$$\max b^\top y \quad \text{s.t.}$$
$$A^\top y = c$$
$$y \geq 0$$





Primal Dual Relations

Primal:

$$\min c^\top x \quad \text{s.t.}$$
$$Ax \leq b$$

Dual:

$$\max b^\top y \quad \text{s.t.}$$
$$A^\top y = c$$
$$y \geq 0$$

$$c^\top \bar{x} = \bar{y}^\top A\bar{x} =^{(*)} \bar{y}^\top b$$

Dual Simplex

- simplex on dual problem
- obtain primal solution as discussed before
- dominant algorithm used in MIP solving

Barrier Method

Idea:

$$\min c^\top x \quad \text{s.t.}$$
$$Ax \leq b$$

Barrier Method

Idea:

$$\min f(x) \quad \text{s.t.}$$
$$c_i(x) \leq 0$$

with f and c_i linear

Barrier Method

Idea:

$$\min f(x) \quad \text{s.t.}$$
$$c_i(x) \leq 0$$

with f and c_i linear

$$\Downarrow$$

$$\min f(x) - \mu \sum_i log(c_i(x))$$

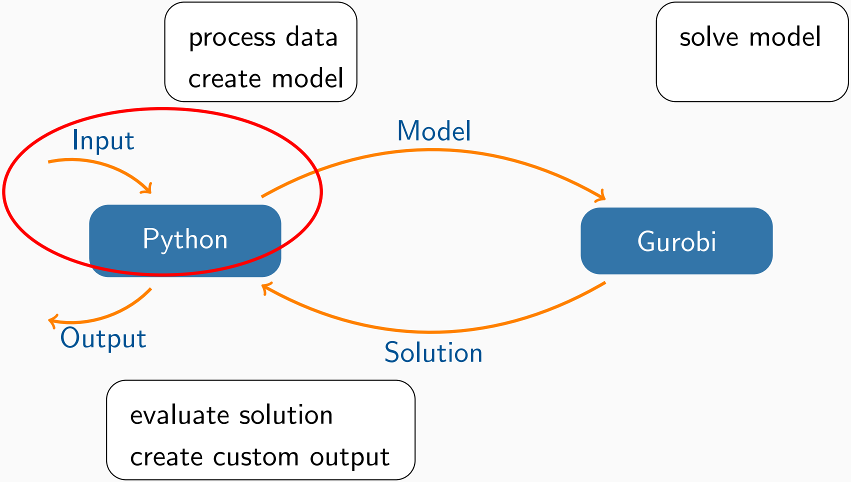
Barrier Method

$$\min f(x) - \mu \sum_i log(c_i(x))$$

- $\mu \to 0$ and solve as nonlinear optimization problem
- also known as Interior Point Method
- few, but expensive calculations
- not clear whether *warm start* is doable

Advanced Input Methods

Advanced Input



Reading Data from Files

Advantages:

- separation between data and code
- faster replacement and sharing of data
- more flexible code
- clearer code

File Formats

- excel (pandas, xlrd)
- csv (csv)
- json (json)
- basically any text-file of some custom format (write parser)

The JSON File Format


```
{
    "oil_types":[
        "heavy","medium","light"
    ],
    "processes":[0,1],
    "production":{
        "heavy":[2,1],"medium":[2,2],"light":[1,4]
    },
    "demand":{
        "heavy":3,"medium":5,"light":4
    },
    "process_cost":[3,5]
}
```

Data Types

- Boolean
- Number (integer or floating point)
- String
- Array [] (ordered list of elements of arbitrary type)
- Object {} (unordered collection of name-value pairs)

Data Types

- Boolean
- Number (integer or floating point)
- String
- Array [] (ordered list of elements of arbitrary type)
- Object {} (unordered collection of name-value pairs)

Translates straight-forward to Python!

Reading JSON files in Python

```
import json

with open("data.json") as json_file:
data = json.load(json_file)
```

Read text files

- generally, input files not given as *json* or *csv* or *xslx*
- any custom format
- solution: read file line by line according to its syntax and create list/dictionary/object

Reading general textfiles in Python

```
filename = 'data.txt'
with open(filename, "r") as file:
    line = file.readline()
    while line:
        print(line.split("separator"))
```

Write text files

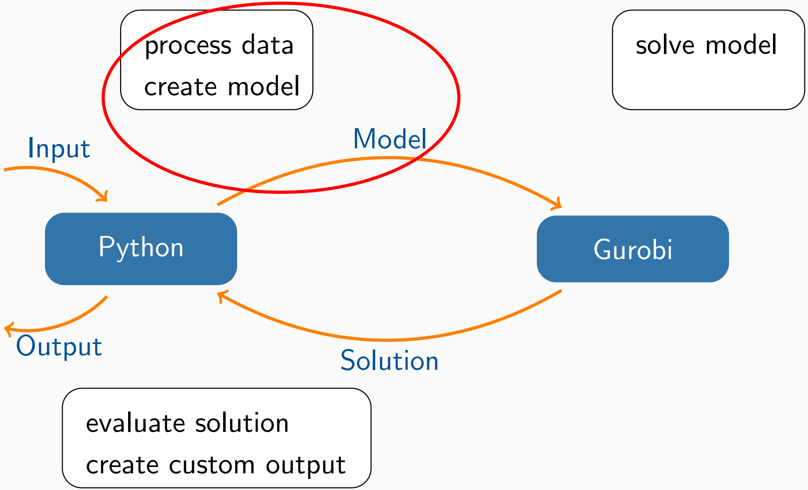
- generate different datasets to have stable collection of example data
- straightforward in python (very similar to *print*())

Writing textfiles in Python

```
filename = 'data.txt'
with open(filename, "w") as file:
    file.write("sometext\n")
    file.close()
```

Advanced Gurobi Datatypes

Advanced Datatypes



Tuplelist

- subclass of python list
- list of tuples of same size
- easy notation to find specific subsets
- same is doable with list comprehension *but* tuplelist is faster

Tuplelist

Creation
`l = tuplelist([(1, 2), (1, 3), (2, 3), (2, 4)])`

Tuplelist

Creation

```
l = tuplelist([(1, 2), (1, 3), (2, 3), (2, 4)])
```

Queries ('∗' is wildcard character)

```
l.select(1, '*')
l.select('*', [2, 4])
l.select('*', '*')
```

Tupledict

- subclass of python dictionary
- dictionary with tuplelist as keys
- usually used for variables of complex systems
- easy access via *select*
- easy constraint generation via *sum* and *prod*

Tupledict

Creation

```
l = tuplelist([(1, 2), (1, 3), (2, 3), (2, 4)])
d=model.addVars(l)
```

Tupledict

Creation

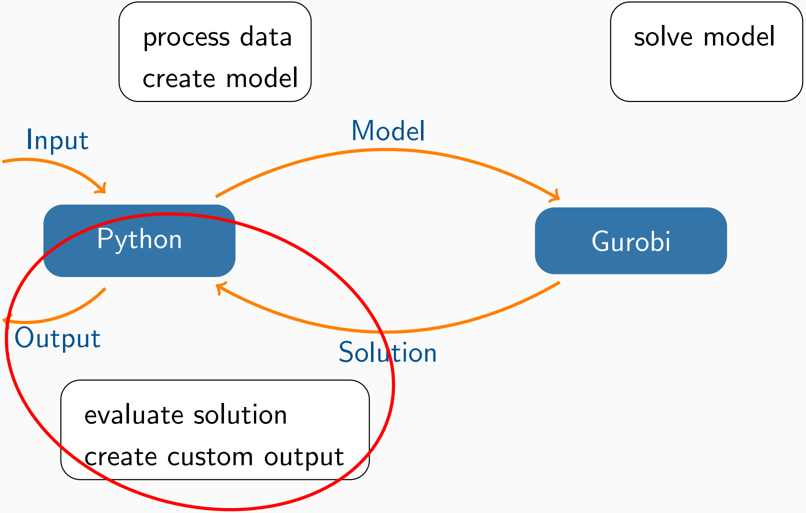
```
l = tuplelist([(1, 2), (1, 3), (2, 3), (2, 4)])
d=model.addVars(l)
```

Queries and creation of expressions

```
d.select(1, '*')
d.sum(1, '*')
coeff=[2,5]
d.prod(coeff, 1, '*')
```

Visualization

Result Visualization



Visualization of Graphs

- natural connection between many LPs and graphs
- visualizing graphs improves understanding
- matplotlib

Visualizing graphs

```
import networkx as nx
from random import randint
n = 10
position = [[randint(0,100),randint(0,100)]
    for i in range(n)]
edges = [(randint(0,n-1),randint(0,n-1))
    for i in range(3*n)]
G = nx.Graph()
G.add_nodes_from([(i, {'x': coord[0], 'y': coord[1]})
    for i, coord in enumerate(position)])
G.add_edges_from([(e[0], e[1]) for e in edges])
nx.draw_networkx_nodes(G, position, node_color="black")
nx.draw_networkx_edges(G, position,edge_color="black")
```