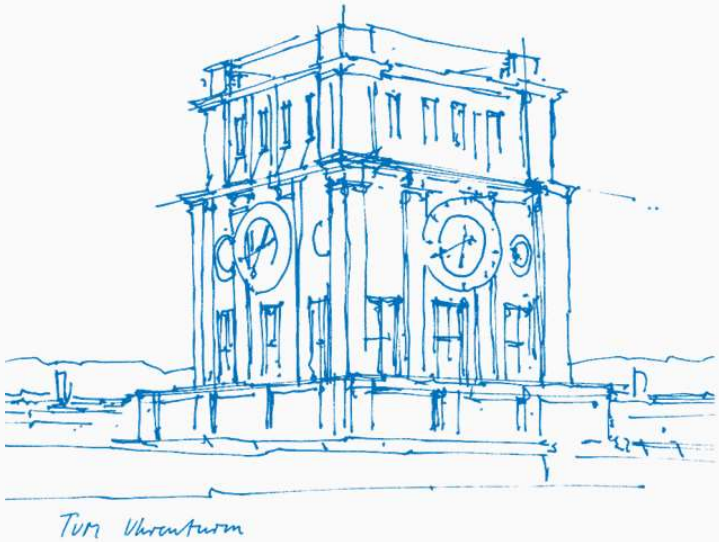
Computer Course Linear Programming

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

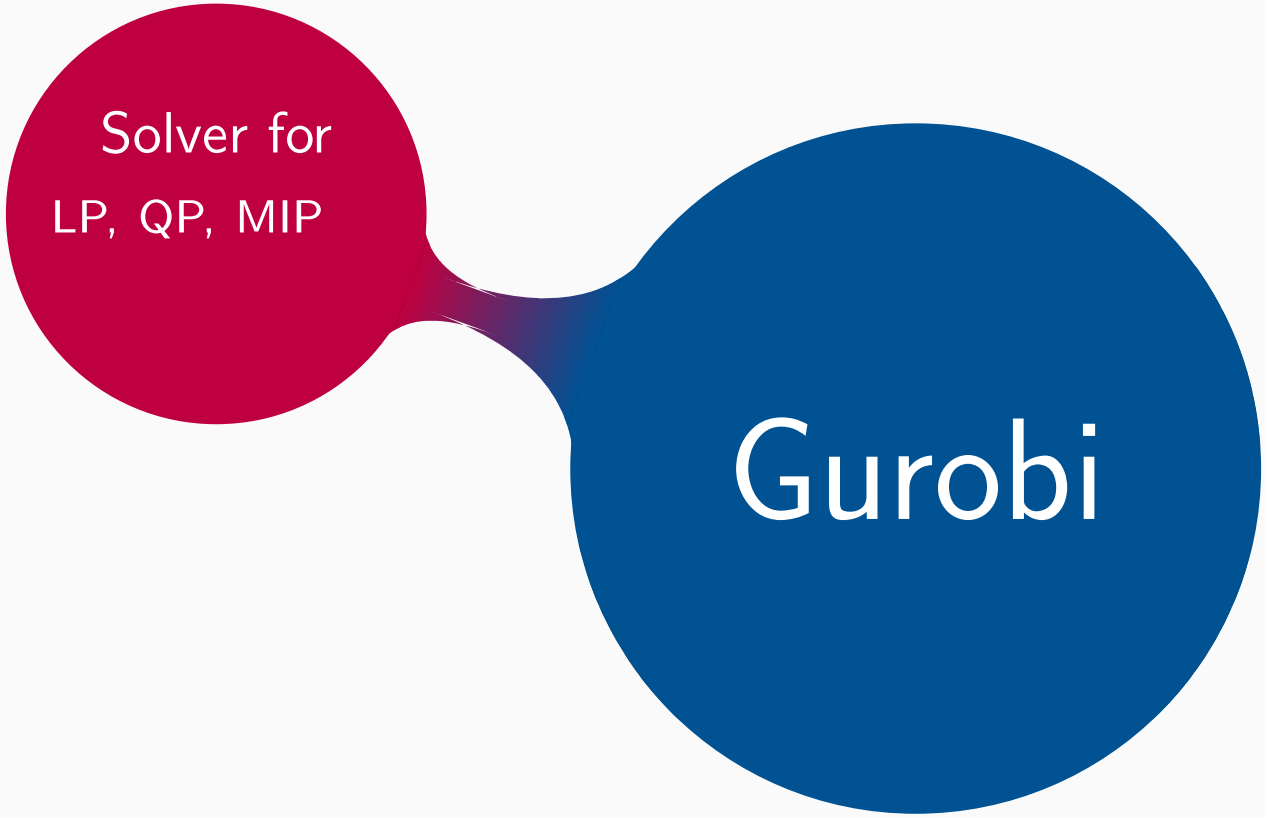
Advanced Input Methods

Advanced Gurobi Datatypes

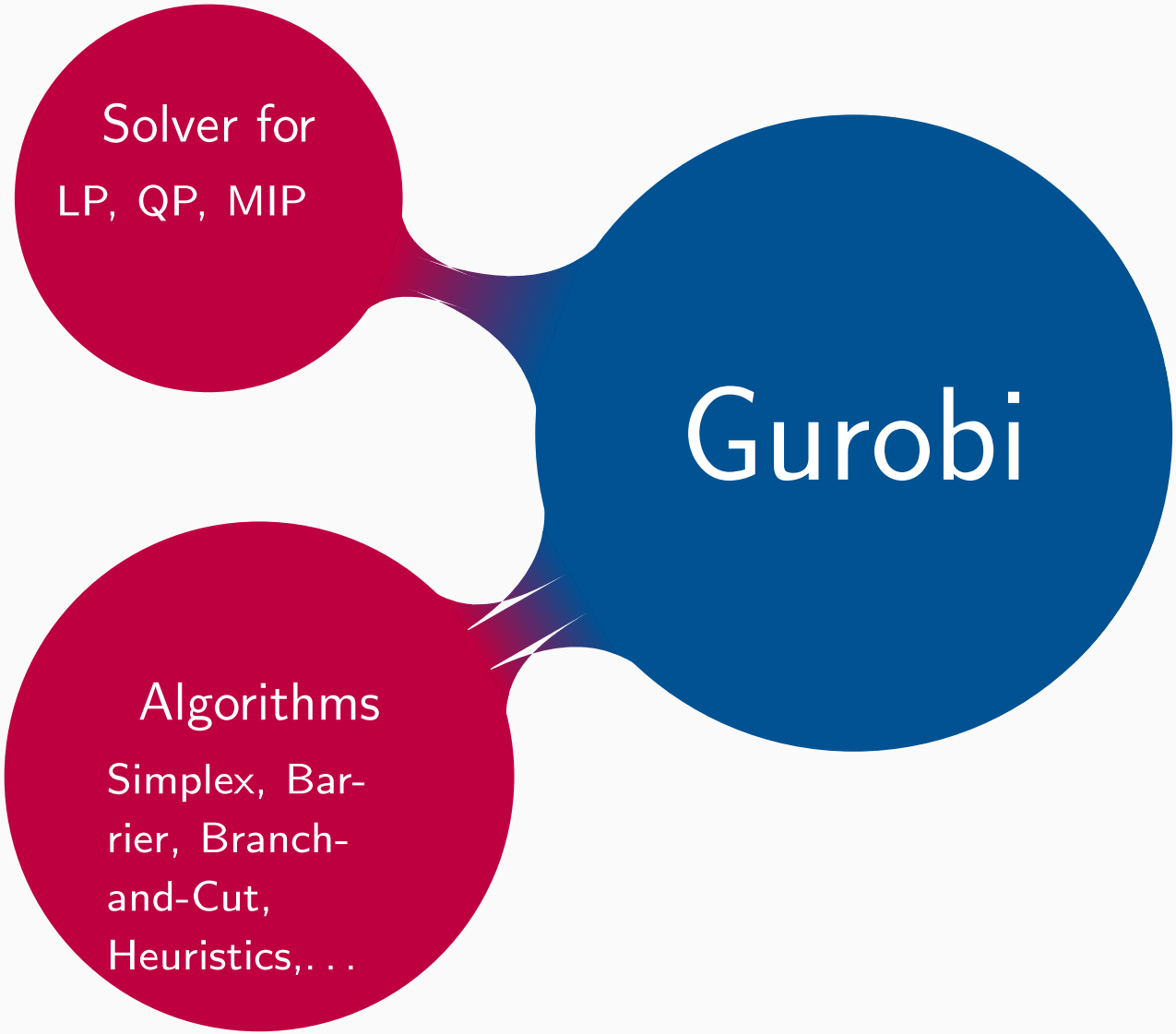
Visualization

Structure of Gurobi

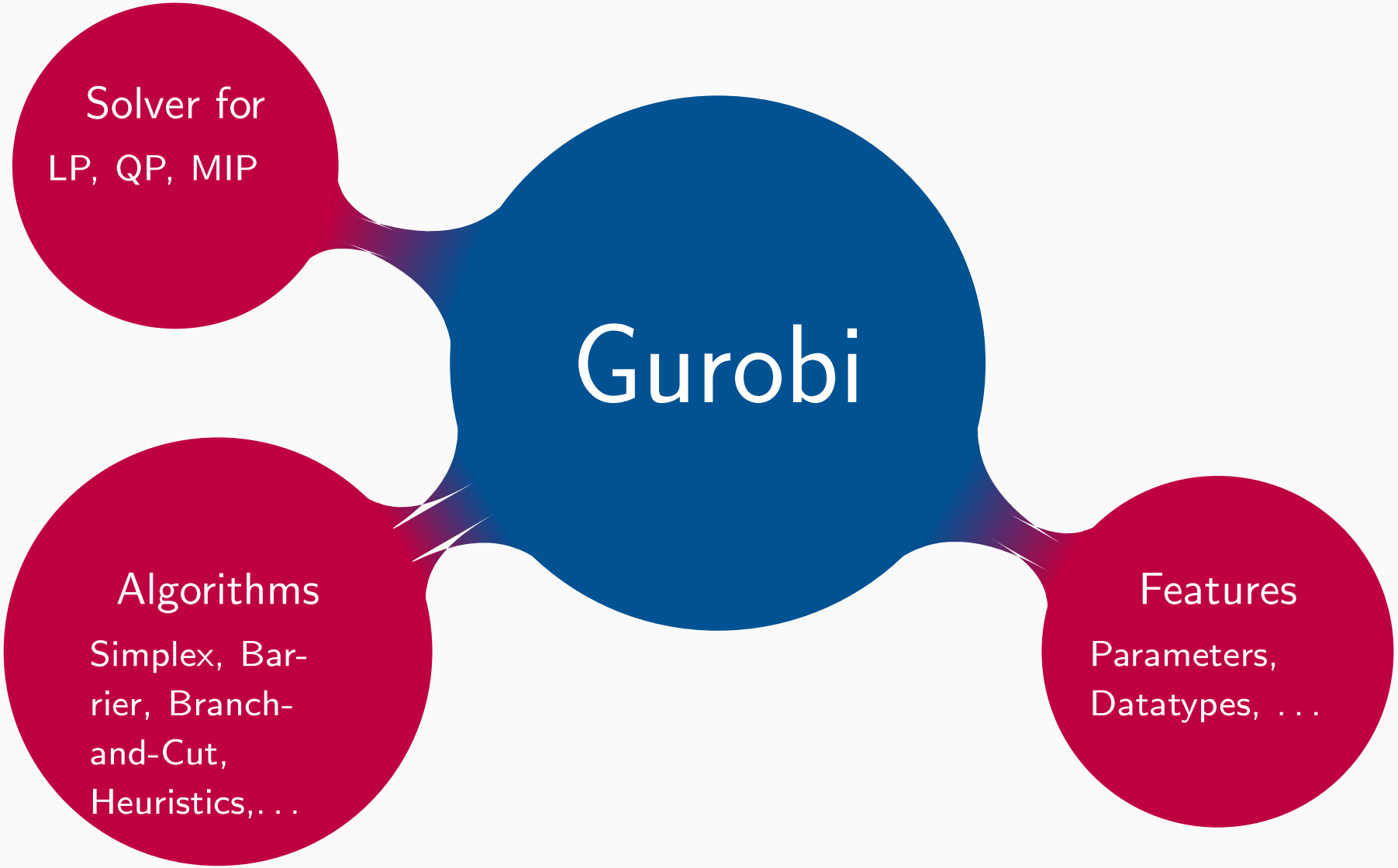
What is Gurobi?



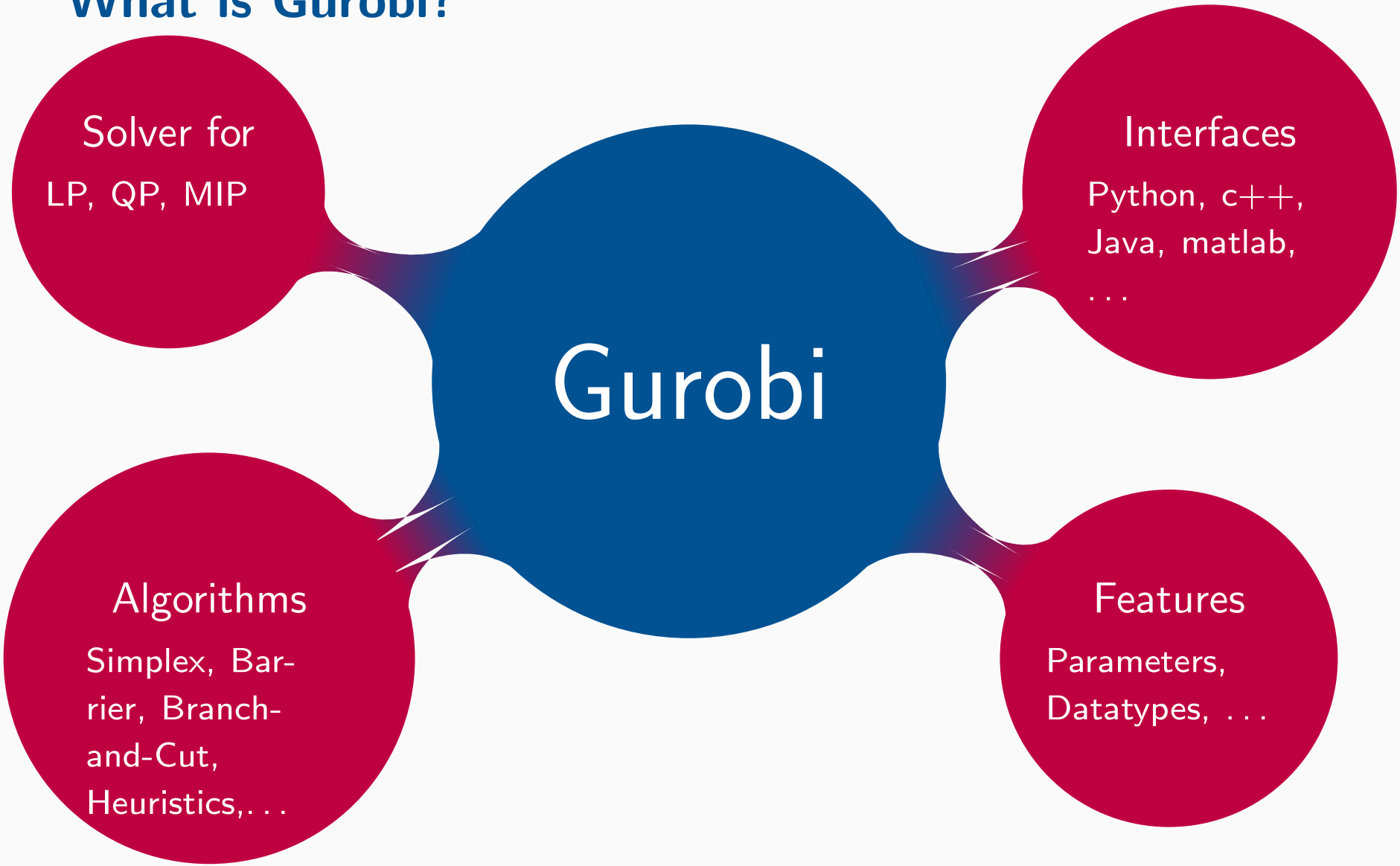
What is Gurobi?



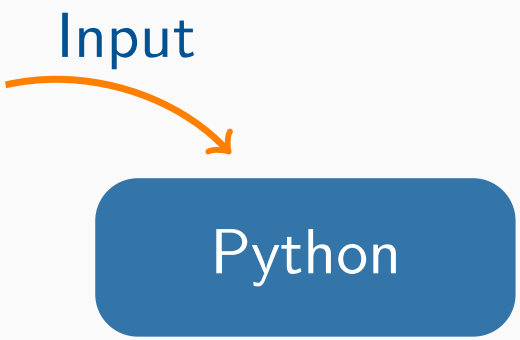
What is Gurobi?



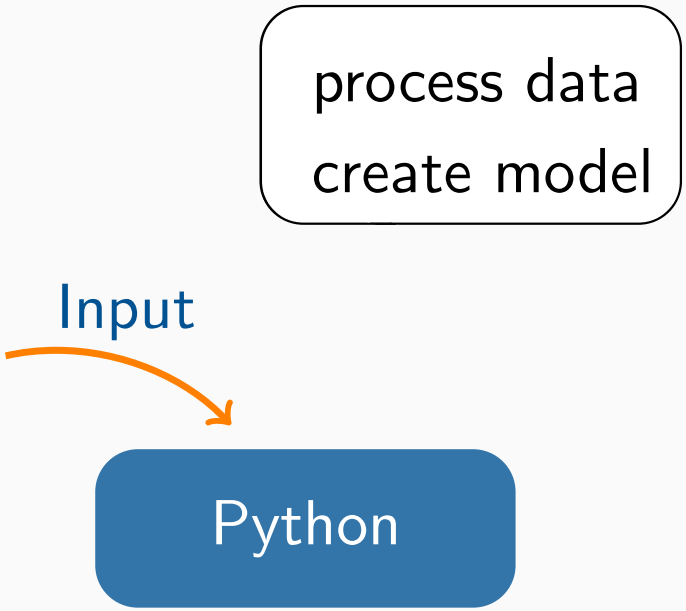
What is Gurobi?



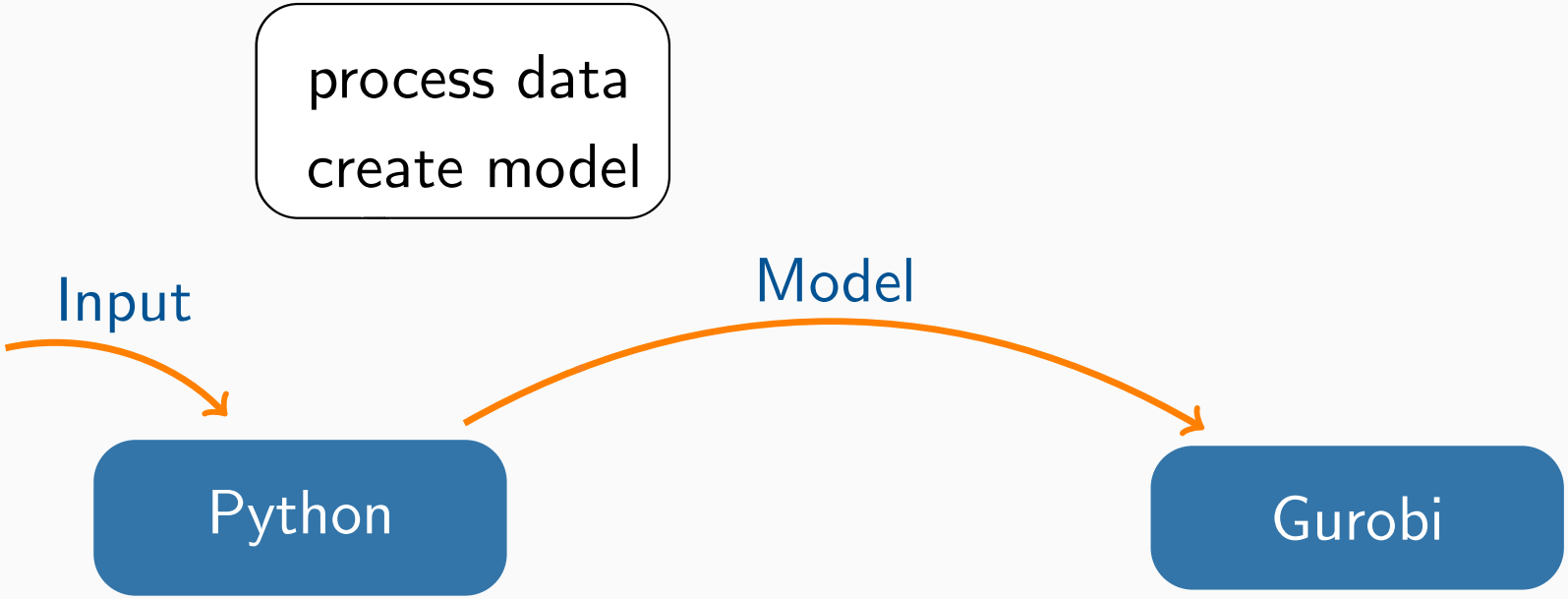
How can we use Gurobi?



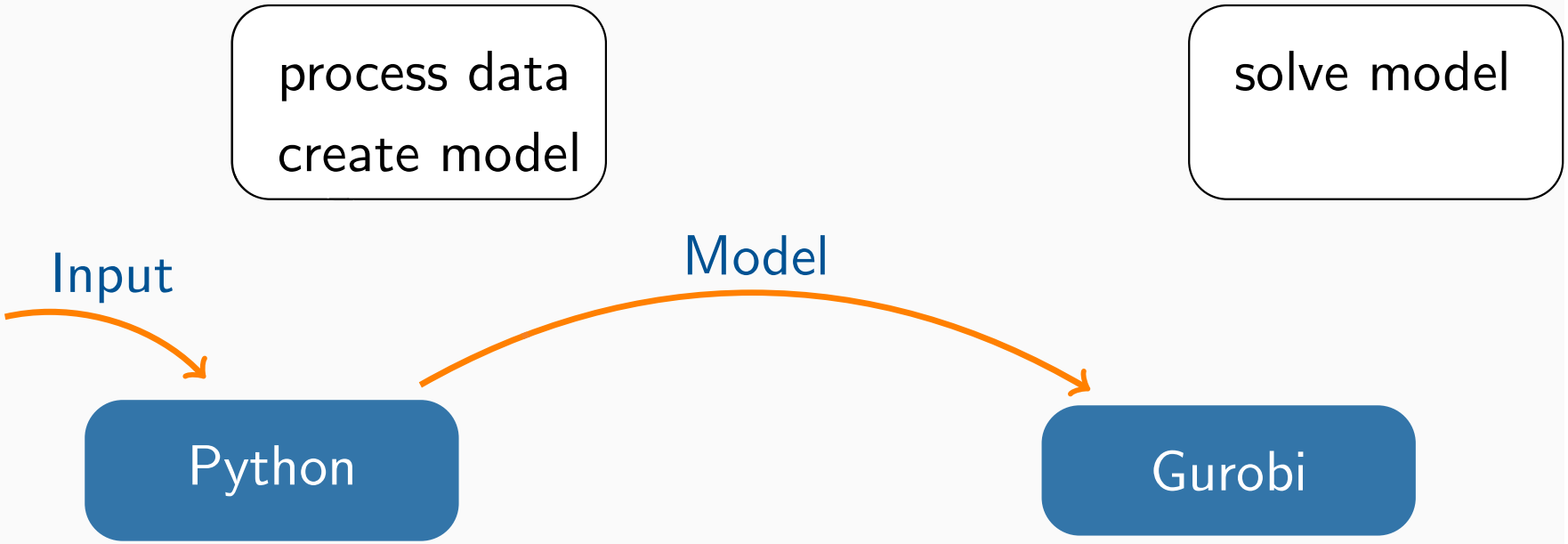
How can we use Gurobi?



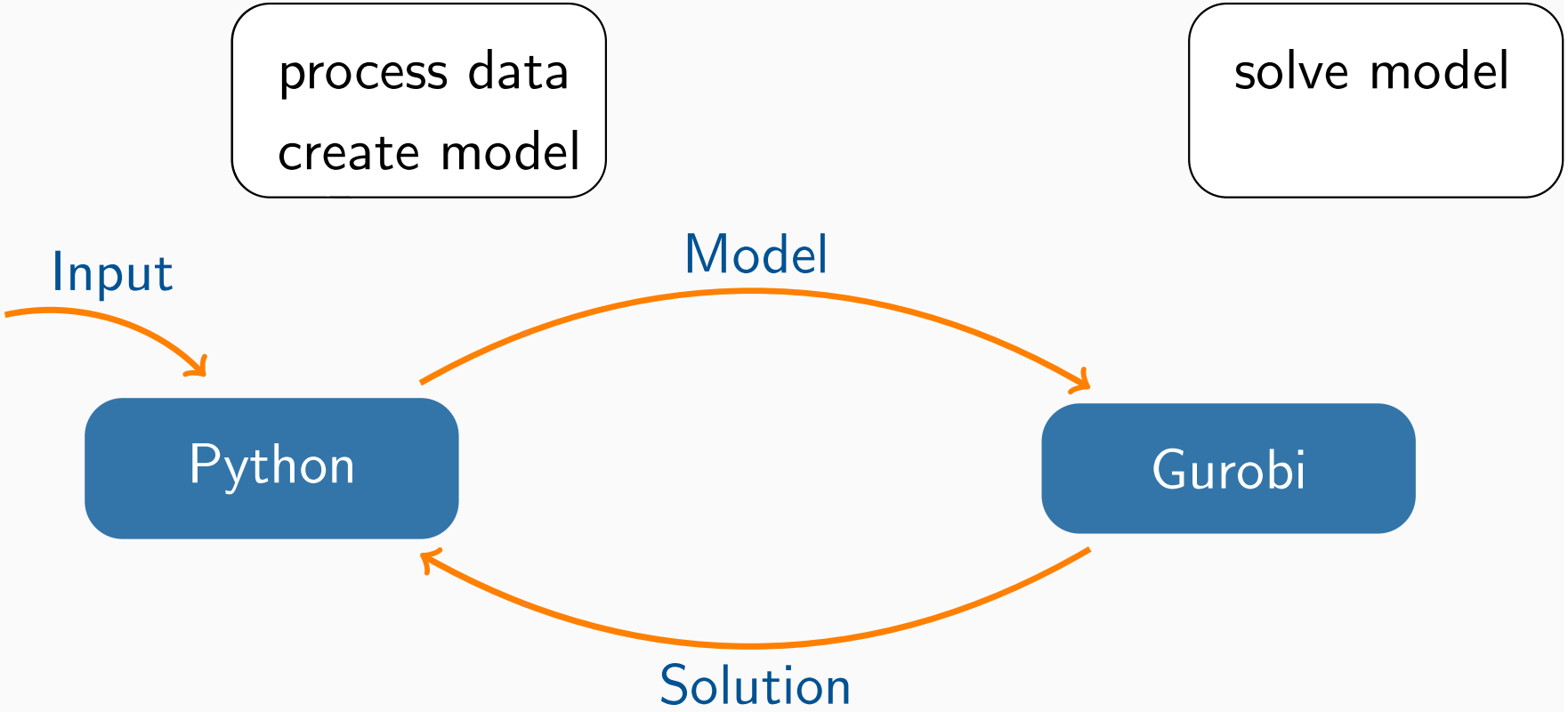
How can we use Gurobi?



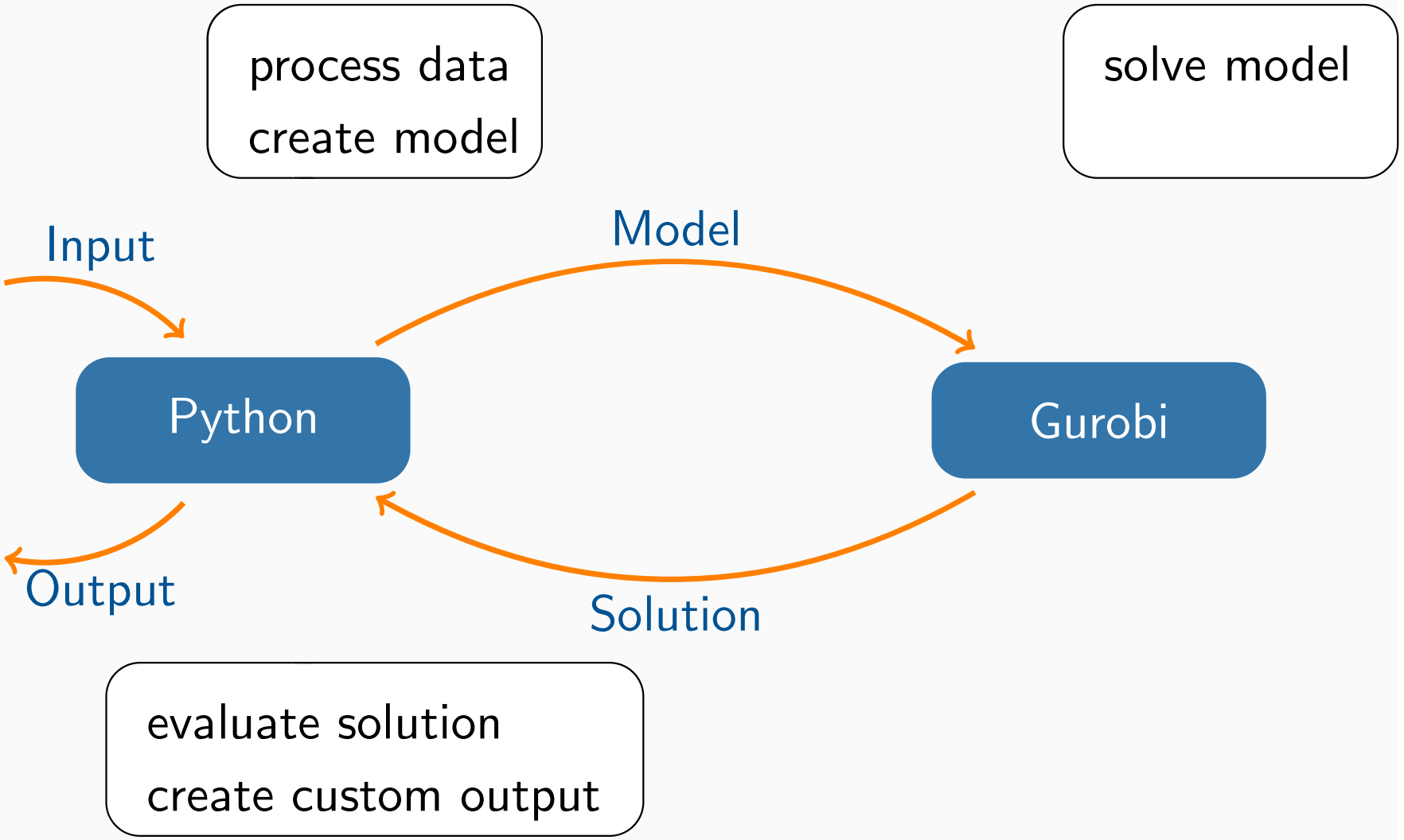
How can we use Gurobi?



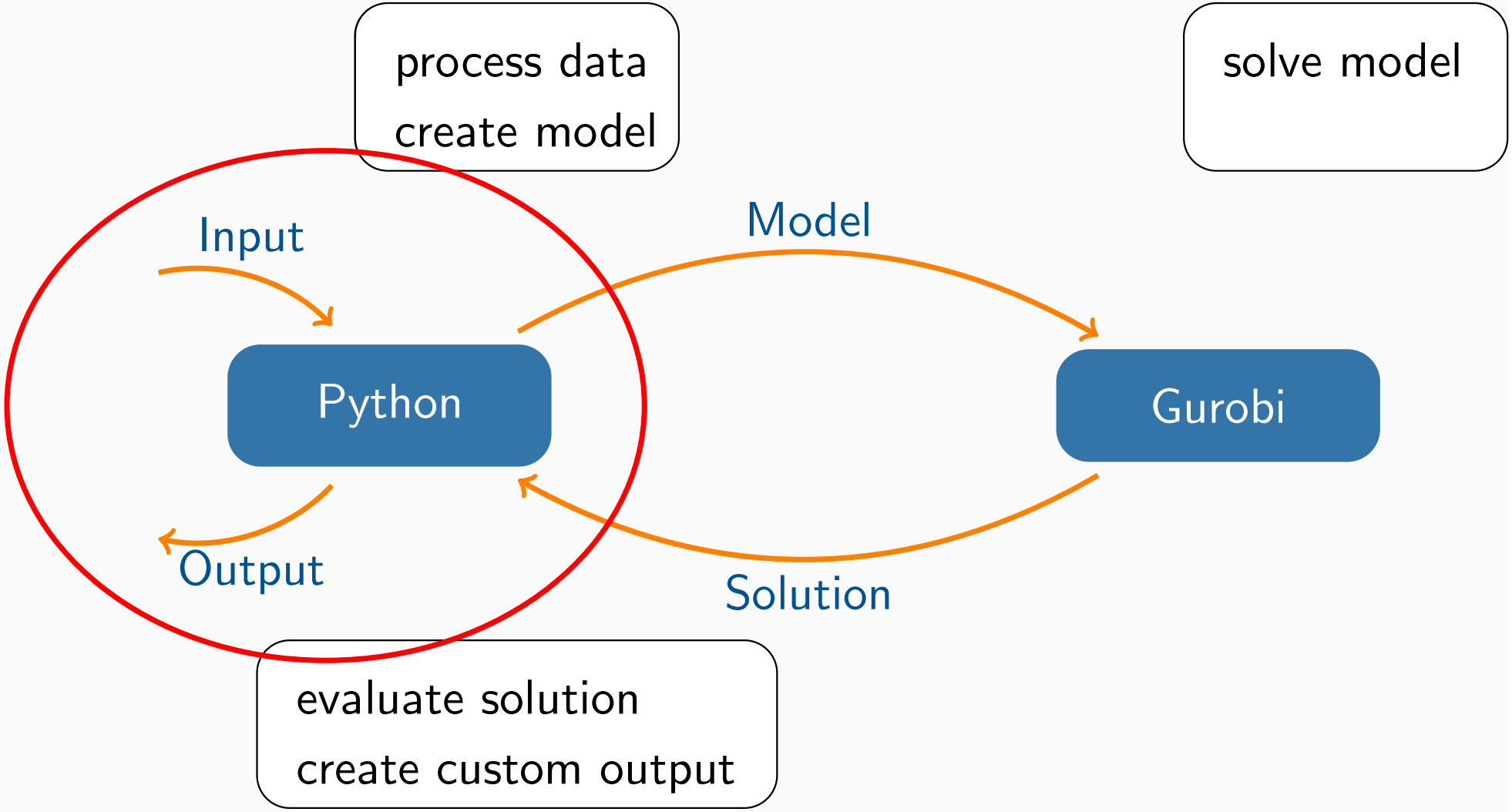
How can we use Gurobi?



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?

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

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

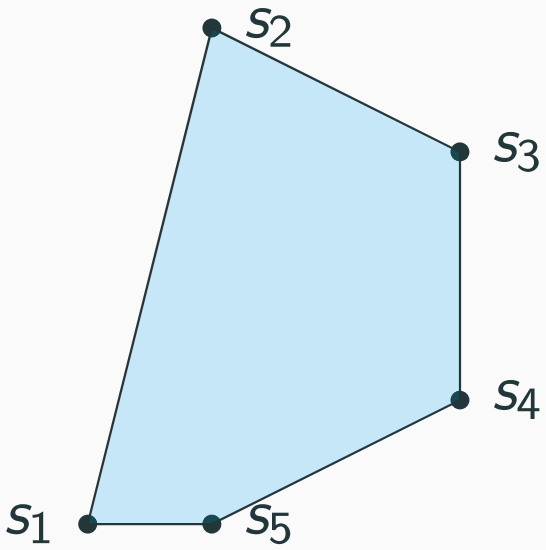
What is a linear program?

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

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

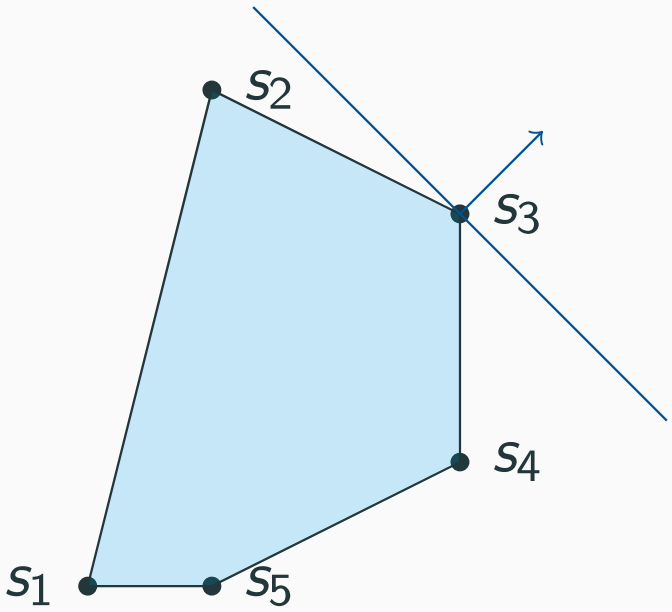
What is a linear program?

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



What is a linear program?

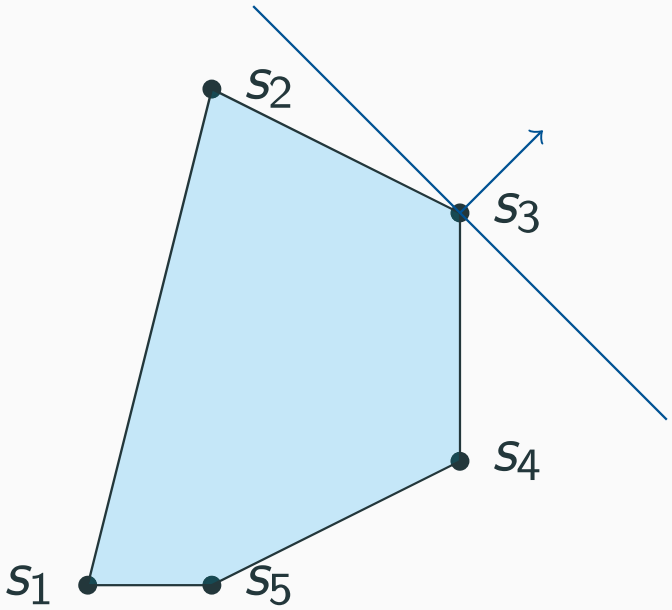
$$\min c^\top x \quad \text{s.t.}$$
$$Ax \leq b$$



How to solve a linear program?

The Simplex Algorithm

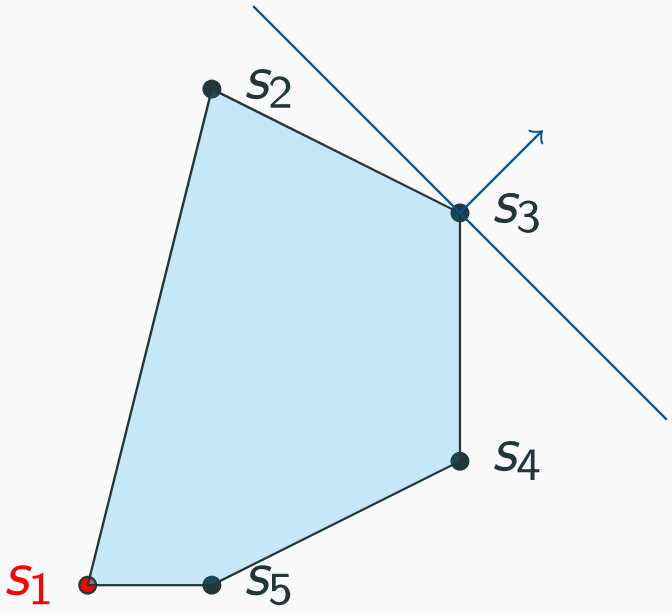
- Find a feasible solution
- Travel along improving edges
- Terminate at optimal solution



How to solve a linear program?

The Simplex Algorithm

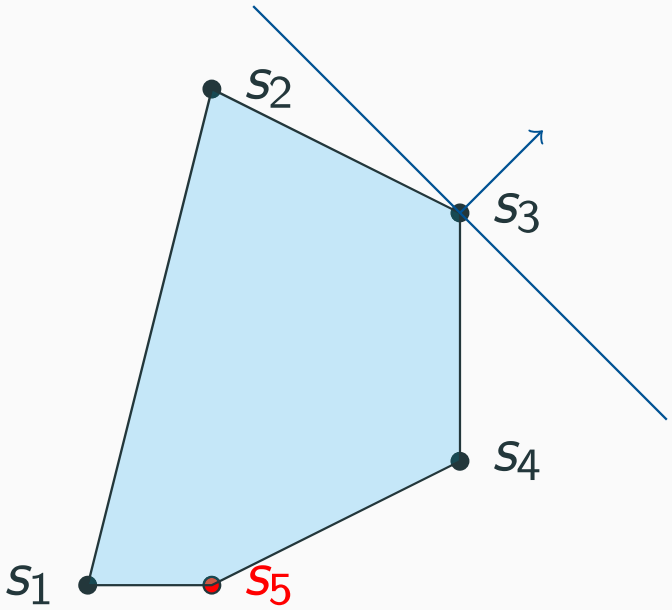
- Find a feasible solution
- Travel along improving edges
- Terminate at optimal solution



How to solve a linear program?

The Simplex Algorithm

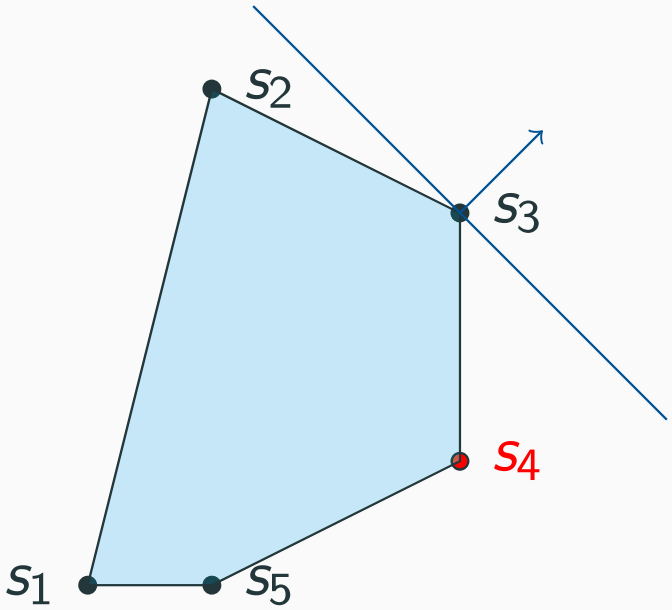
- Find a feasible solution
- Travel along improving edges
- Terminate at optimal solution



How to solve a linear program?

The Simplex Algorithm

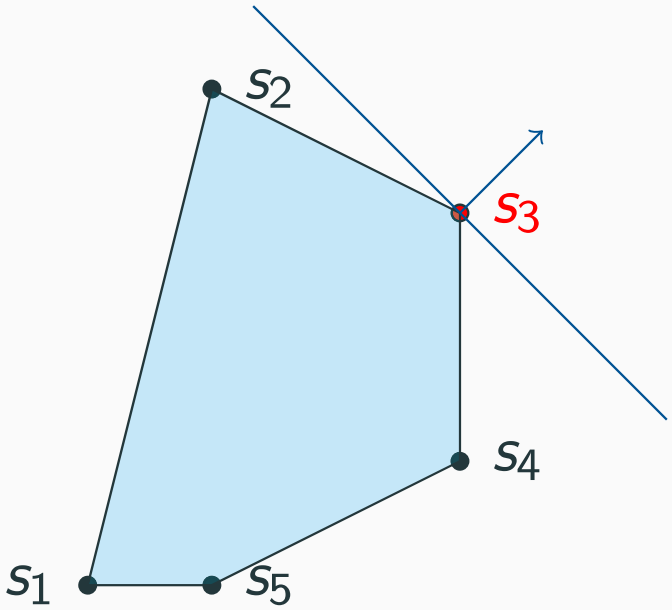
- Find a feasible solution
- Travel along improving edges
- Terminate at optimal solution



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?

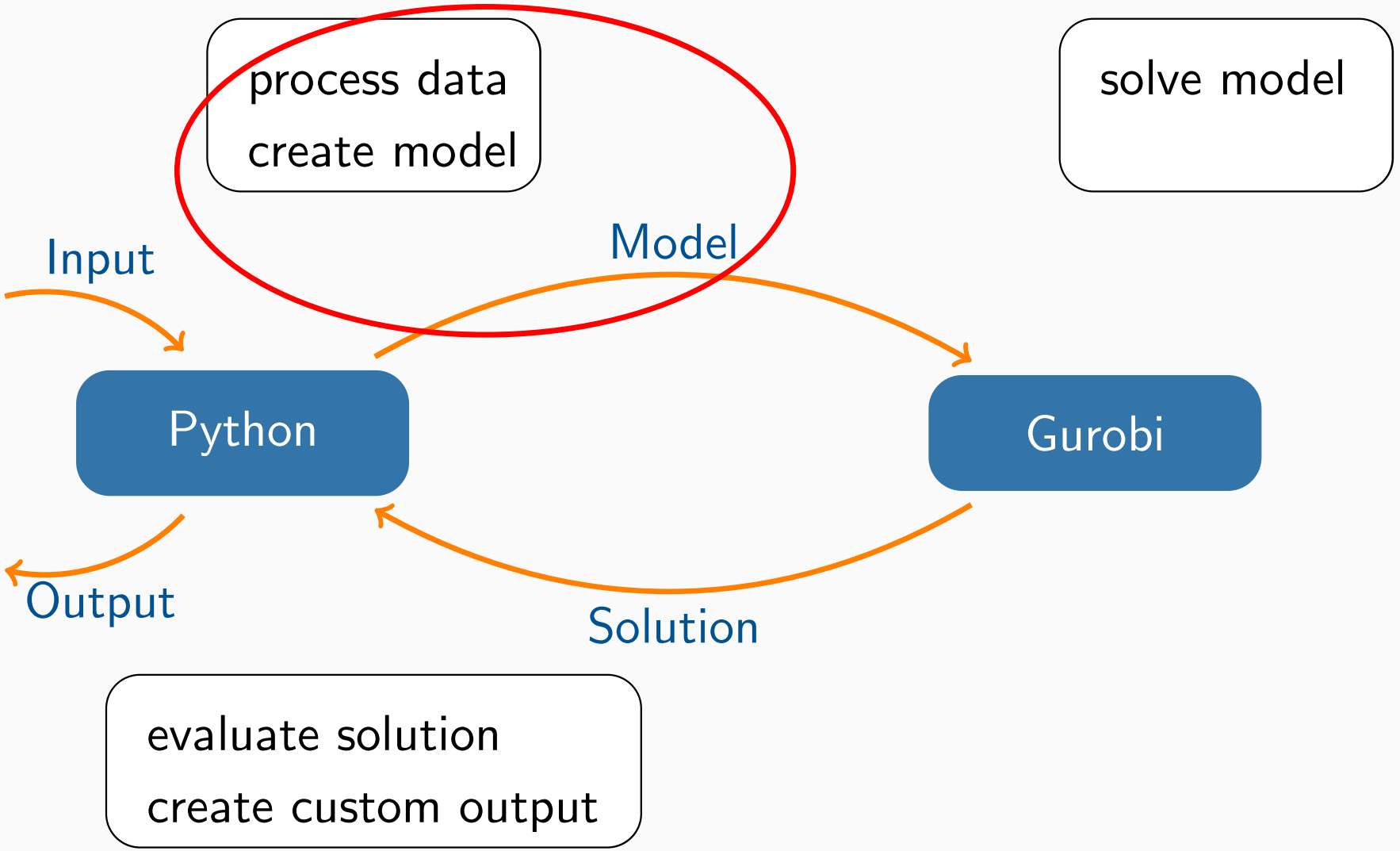
The Simplex Algorithm

- 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

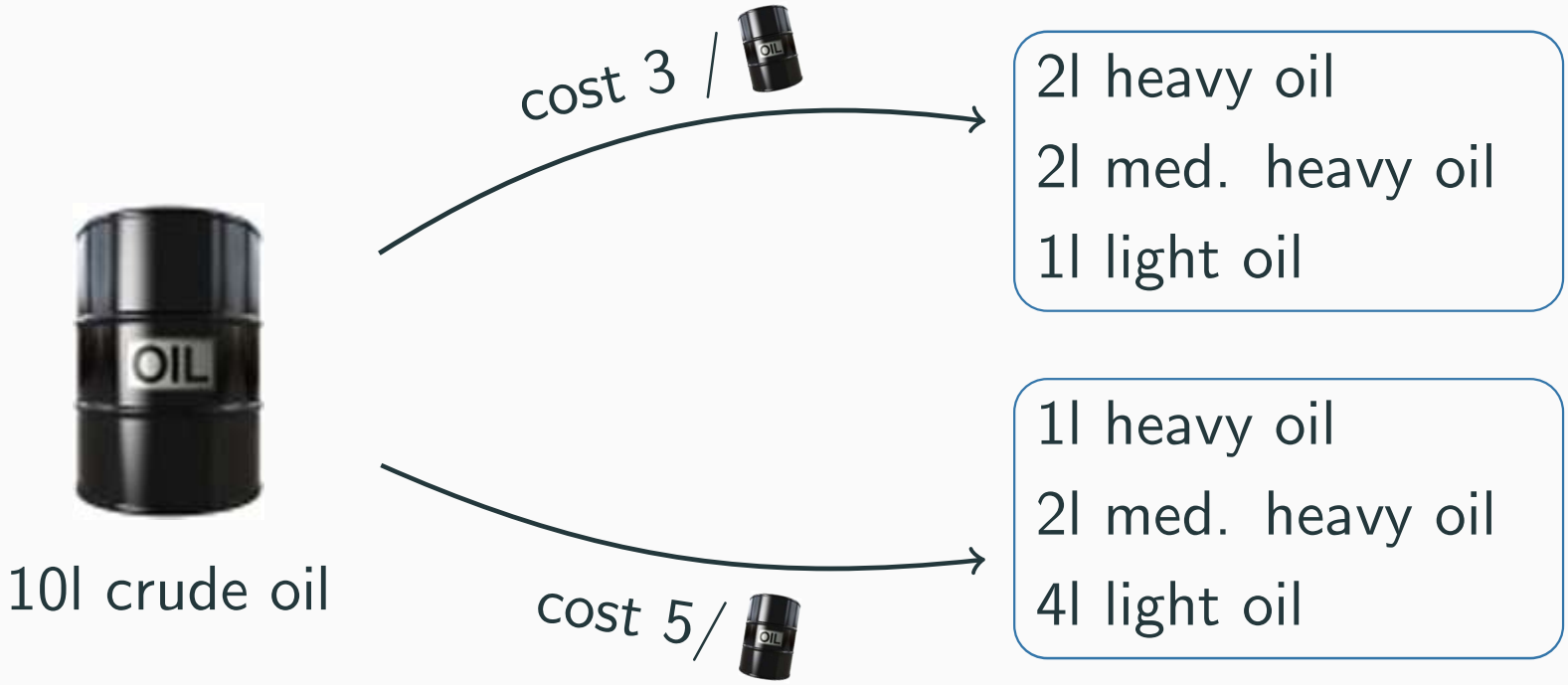
Problem: Crude Oil Refinement 1

10l crude oil

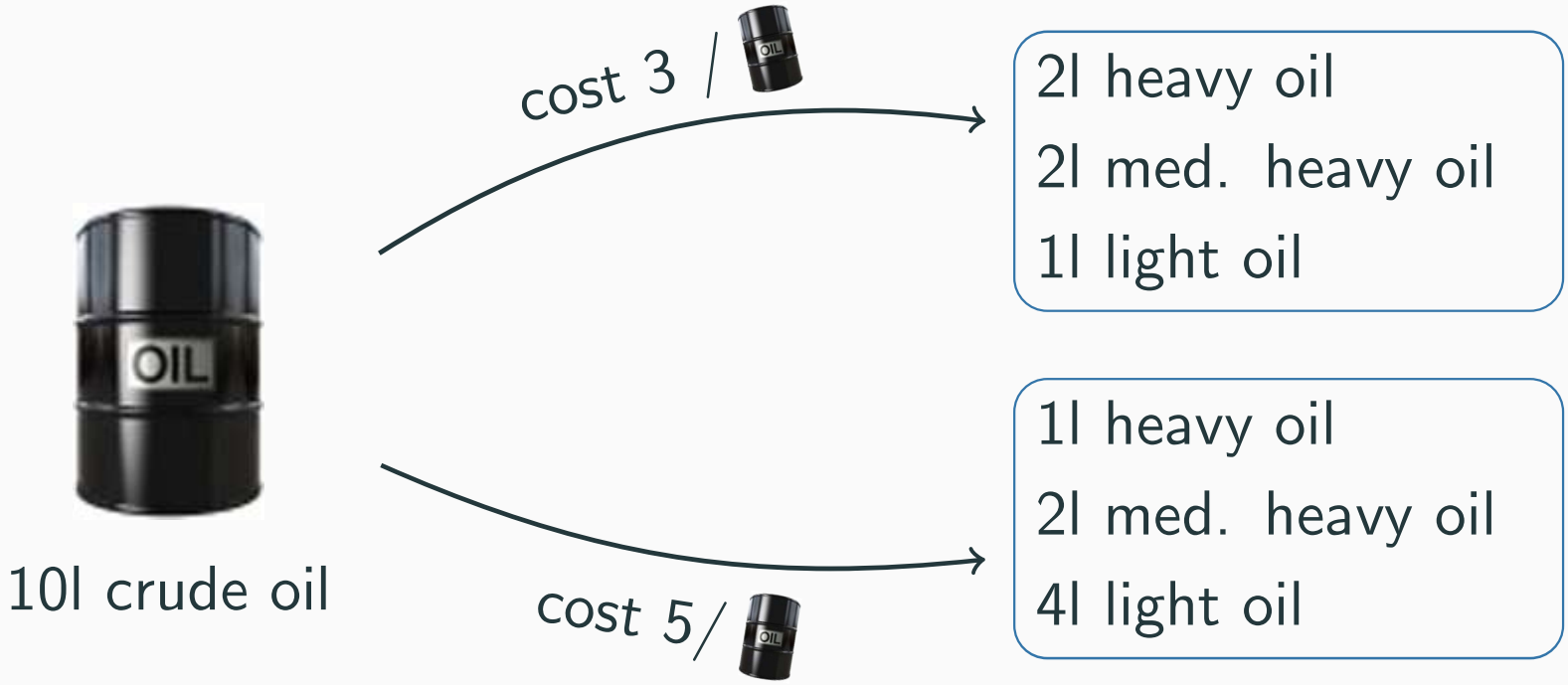
cost 3 /

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

Problem: Crude Oil Refinement 1

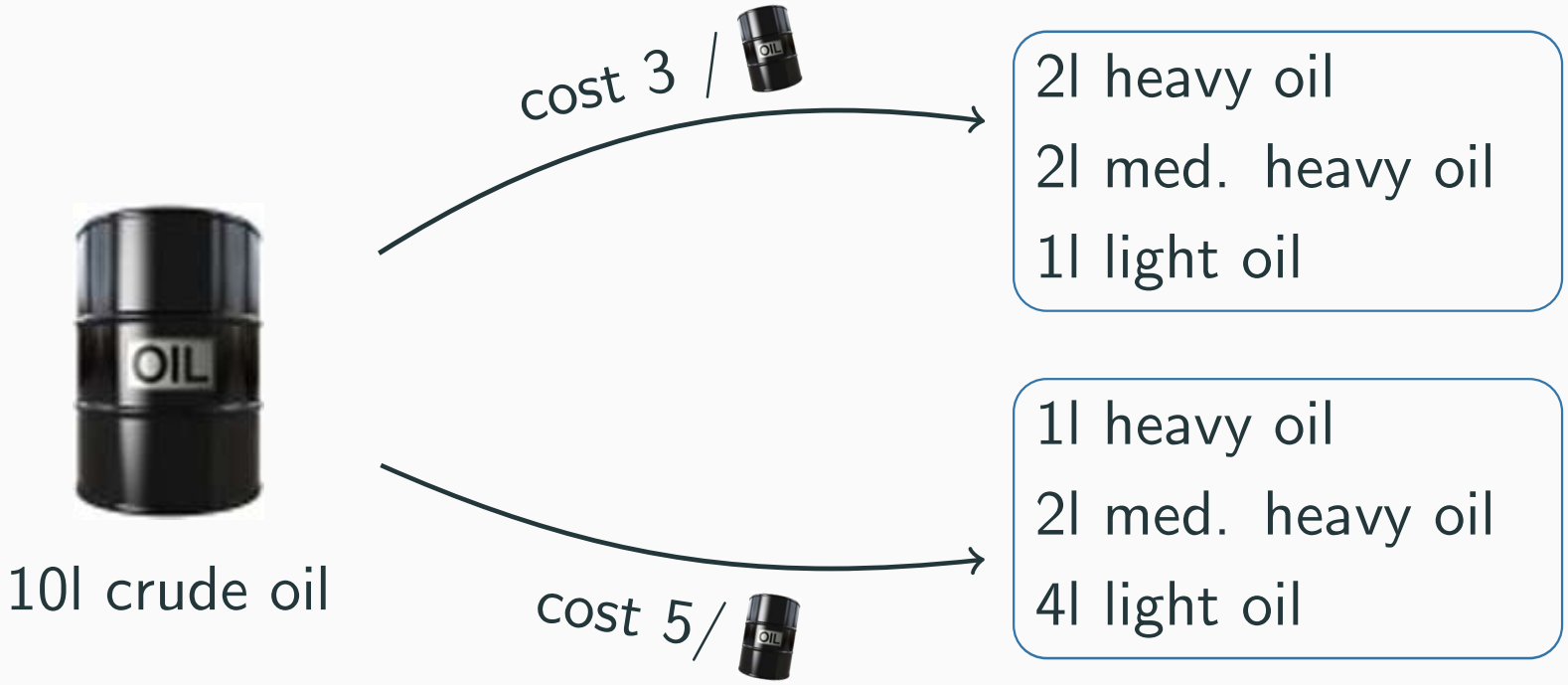


Problem: Crude Oil Refinement 1



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.

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

LP Model Problem 1

$$\min 3x_1 + 5x_2$$

s.t.

$$2x_1 + 1x_2 \geq 3$$

$$2x_1 + 2x_2 \geq 5$$

$$1x_1 + 4x_2 \geq 4$$

$$x_1, x_2 \geq 0$$

LP Model Problem 1

$$\min 3x_1 + 5x_2$$

s.t.

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

LP Model Problem 1

$$\min 3x_1 + 5x_2$$

s.t.

$$2x_1 + 1x_2 \geq 3$$

$$2x_1 + 2x_2 \geq 5$$

$$1x_1 + 4x_2 \geq 4$$

$$x_1, x_2 \geq 0$$

LP Model Problem 1

$$\min 3x_1 + 5x_2$$

s.t.

$$2x_1 + 1x_2 \geq 3$$

$$2x_1 + 2x_2 \geq 5$$

$$1x_1 + 4x_2 \geq 4$$

$$x_1, x_2 \geq 0$$

LP Model Problem 1

$$\min 3x_1 + 5x_2$$

s.t.

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

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()
```