Operations Research II: Algorithms

Gurobi and Python for Linear Programming

Serena Chen

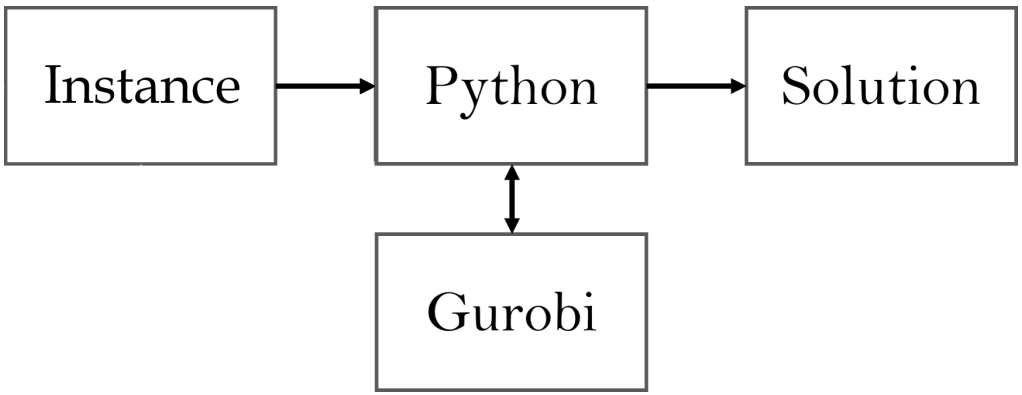
Department of Information Management
National Taiwan University

Road map

- **Gurobi Optimizer**.
- Producing desks and tables.
- Model-data decoupling.

Gurobipy

- **Gurobi Optimizer** is a well-known optimization solver.
  - Companies must pay, but academic use is free.
- We will use `gurobipy`, which is a Python package for a Python program to invoke Gurobi Optimizer.
- **Python** is used as an interface, and **Gurobi Optimizer** is a solver.



To obtain Gurobi Optimizer

- Official website: `https://www.gurobi.com/`.
- In this course, we use the academic license.
- Get Gurobi Optimizer license from `https://www.gurobi.com/downloads/end-user-license-agreement-academic/`.
  - Remember to access the web page in an **academic institution**, e.g., your university.
  - Remember to **copy your license code**.
- Download Gurobi Optimizer from `https://www.gurobi.com/downloads/gurobi-optimizer-eula/`.
- After the installation, open your Command Prompt or Terminal, enter

  ```
  grbgetkey [your Gurobi license code]
  ```

  to activate your Gurobi Optimizer license.
- After the activation, you may use Gurobi Optimizer anywhere.

To obtain Python

- Official website: `https://www.python.org/`.
- Download Python 3 from `https://www.python.org/downloads/`.
- Notice that Gurobi Optimizer **does not** support Python 3.8, please install other versions, e.g., Python 3.7.
- Use any Python environment you like (in the example, it is Jupyter Lab).
- Try it!
- Here our introduction is based on the MS Windows version.
  - The way to write the code is the same on MS Windows and Mac.
  - The only major difference is the screen shots.

Road map

- Gurobi Optimizer.
- **Producing desks and tables**.
- Model-data decoupling.

Producing desks and tables

- Consider the problem we have introduced in *Operations Research I: Modeling and Application*. Let

$$x_1 = \text{number of desks produced in a day and}$$
$$x_2 = \text{number of tables produced in a day.}$$

- The formulation of this example is

$$\begin{array}{lrcrcll} \max & 700x_1 & + & 900x_2 & & & \\ \text{s.t.} & 3x_1 & + & 5x_2 & \leq & 3600 & \text{(wood)} \\ & x_1 & + & 2x_2 & \leq & 1600 & \text{(labor)} \\ & 50x_1 & + & 20x_2 & \leq & 48000 & \text{(machine)} \\ & x_1 & & & \geq & 0 & \\ & & & x_2 & \geq & 0. & \end{array}$$

Producing desks and tables

- To build a new model with **gurobipy**, open your development environment and type these codes into a new python file:

```
from gurobipy import *

eg1 = Model("eg1")
x1 = eg1.addVar(lb = 0, vtype = GRB.CONTINUOUS, name = "x1")
x2 = eg1.addVar(lb = 0, vtype = GRB.CONTINUOUS, name = "x2")

eg1.setObjective(700 * x1 + 900 * x2 , GRB.MAXIMIZE)
eg1.addConstr(3 * x1 + 5 * x2 <= 3600, "resource_wood")
eg1.addConstr(x1 + 2 * x2 <= 1600, "resource_labor")
eg1.addConstr(50 * x1 + 20 * x2 <= 48000, "resource_machine")

eg1.optimize()
```

- Save the file then run it. Let's try it first and explain the codes later.

Producing desks and tables

```
Gurobi Optimizer version 9.0.2 build v9.0.2rc0 (win64)
Optimize a model with 3 rows, 2 columns and 6 nonzeros
Model fingerprint: 0xa395d65c
Coefficient statistics:
  Matrix range     [1e+00, 5e+01]
  Objective range  [7e+02, 9e+02]
  Bounds range     [0e+00, 0e+00]
  RHS range        [2e+03, 5e+04]
Presolve time: 0.01s
Presolved: 3 rows, 2 columns, 6 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.0000000e+32   3.593750e+30   2.000000e+02      0s
       3    7.8947368e+05   0.000000e+00   0.000000e+00      0s

Solved in 3 iterations and 0.02 seconds
Optimal objective  7.894736842e+05
```

Producing desks and tables

- To see the solution, type

```
for var in eg1.getVars():
  print(var.varName, '=', var.x)
print("objective value =", eg1.objVal)
```

  Save the file and then run it.
- You may customize the output format by writing your own programs.
  - You may even save the solution into a file.
- The values are displayed.
  - $x^* = (884.21, 189.47)$.
  - Objective value $z^* = 789473.68$.

Code revisited

- Let's explain the code.

```
# build a new model
eg1 = Model("eg1")  # build a new model, name it as "eg1"

x1 = eg1.addVar(lb = 0, vtype = GRB.CONTINUOUS, name = "x1")
x2 = eg1.addVar(lb = 0, vtype = GRB.CONTINUOUS, name = "x2")
```

  - Use the constructor `Model` and the function `addVar` defined in `gurobipy`.
  - In `addVar`, we use `lb` to set the lower bound, `vtype` to set the type of the variable, and `name` to set the name of this variable.
  - The type `GRB.CONTINUOUS` means this variable is a coutinuous number, which may not be an integer.
  - Do not forget to use `#` to write **comments**.

Code revisited

- Let's explain the code.

```
# setting the objective function
# use GRB.MAXIMIZE for a maximization problem
eg1.setObjective(700 * x1 + 900 * x2, GRB.MAXIMIZE)

# add constraints and name them
eg1.addConstr(3 * x1 + 5 * x2 <= 3600, "resource_wood")
eg1.addConstr(x1 + 2 * x2 <= 1600, "resource_labor")
eg1.addConstr(50 * x1 + 20 * x2 <= 48000, "resource_machine")
```

  - Used Gurobi function: `setObjective` and `addConstr`.
  - Give all constraints, variables, and the model **distinct names**.

Code revisited

- Let's explain the code.

```
eg1.optimize()
```

  - Used Gurobi function: `optimize` to **run and solve** the model.

- Let's explain the code.

```
for var in eg1.getVars():
  print(var.varName, '=', var.x)
print('objective value =', eg1.objVal)
```

  - Use Gurobi function: `getVars` to get the list of all variables in the model, and then use `for` loop to show **the value of all variables**.
  - Get **objective value**: `objVal`.

Road map

- Gurobi Optimizer.
- Producing desks and tables.
- **Model-data decoupling**.

Decoupling the data from a model

- To make our Python programs flexible and extendible, we should **decouple** the data from a model.
- To do this, we will prepare several **lists** to store the data.
  - The list contains the instance parameters.
  - The model part only contains an abstract model.
- In this example, the lists are in our Python program **before** the model part.

The data part

```
products = range(2)  # 2 products
resources = range(3)  # 3 resources

prices = [700, 900]
resource_consumptions = [[3 , 5 ],
                         [1 , 2 ],
                         [50, 20]]
resource_limitations = [3600, 1600, 48000]
```

- We use lists to store the data.

The model part

```
eg1_de = Model("eg1_decoupling")

x = []
for i in products:
  x.append(eg1_de.addVar(lb = 0, vtype = GRB.CONTINUOUS,
    name = 'x' + str(i)))
```

- The length of list and matrix must be consistent. Use `.append()` to add a new item into a list.
- Be aware of those `[]`, `:`, and space and the timing of using them.

The model part

```
eg1_de.setObjective(
  quicksum(prices[i] * x[i] for i in products),
    GRB.MAXIMIZE)

eg1_de.addConstrs(
  (quicksum(resource_consumptions[j][i] * x[i]
    for i in products)
      <= resource_limitations[j] for j in resources)
      , "Resource_limitation")
```

- In `setObjective`, use `GRB.MAXIMIZE` for maximization problems.
- Use `quicksum` and `for` to sum up a group of variables.
- Use `addConstrs` and `for` to add multiple constraints in one command.
- Be aware of the length of lists.

The model part

- To solve the problem and get the solution, type

```
eg1_de.optimize()

for var in eg1_de.getVars():
  print(var.varName, '=', var.x)
print("objective value =", eg1_de.objVal)
```

- The solution is exactly the same.

Some Remarks

- Python is **case-sensitive**.
- Do model-data decoupling to make your model more extendible.
  - There should be no hard-coded number in the model part.
- If you have not learned Python yet, you may find lots of Python tutorials on the Internet.