Introduction to Modeling with Gurobi Python Interface

Wenbo Ma

December, 2022

Agenda

- Installations
  - Installations of Anaconda(Python) and gurobipy
  - License Request
- Example:
  - Multiple searchers route optimization problem
    - Parameters, Decision Variables, Objective Function, Constraints
- Running the program
  - Interactive mode (Spyder)
  - Command Line in Linux
- Jupyter Notebook

Installation

- Install Python via Anaconda
  - https://www.anaconda.com/products/distribution
- Install Gurobi (i.e. gurobipy) via Conda
  - https://support.gurobi.com/hc/en-us/articles/360044290292-How-do-I-install-Gurobi-for-Python-
- Obtain a Gurobi Academic License

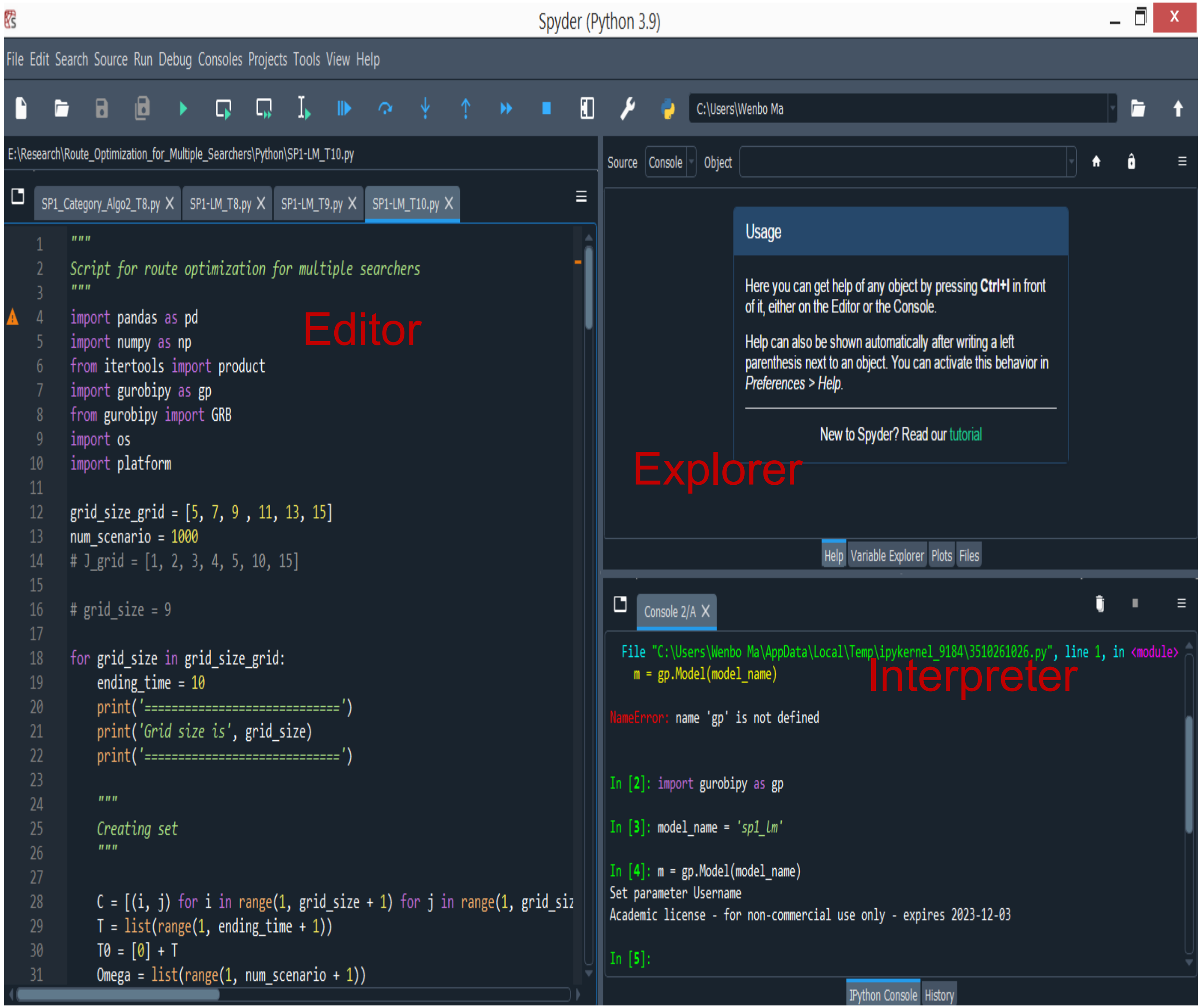
- Register an account using your edu email
- Request an academic license on Gurobi's official website (under MyAccount – MyLicenses)
- Request a "Academic Named-User License" (https://www.gurobi.com/downloads/free-academic-license/)
- Run Anaconda Prompt: grbgetkey c26d1936-7698-11ed-8ee8-0242ac190003 (You will get a different key from your own request. You need to be on campus's wifi or VPN when running the command)

```
(base) PS C:\Users\Wenbo Ma> grbgetkey c26d1936-7698-11ed-8ee8-0242ac190003
info  : grbgetkey version 10.0.0, build v9.5.2rc0-1468-g91169db51
info  : Contacting Gurobi license server...
info  : License file for license ID 910292 was successfully retrieved
info  : License expires at the end of the day on 2023-12-03
info  : Saving license file...

In which directory would you like to store the Gurobi license file?
[hit Enter to store it in C:\Users\Wenbo Ma]:

info  : License 910292 written to file C:\Users\Wenbo Ma\gurobi.lic
```

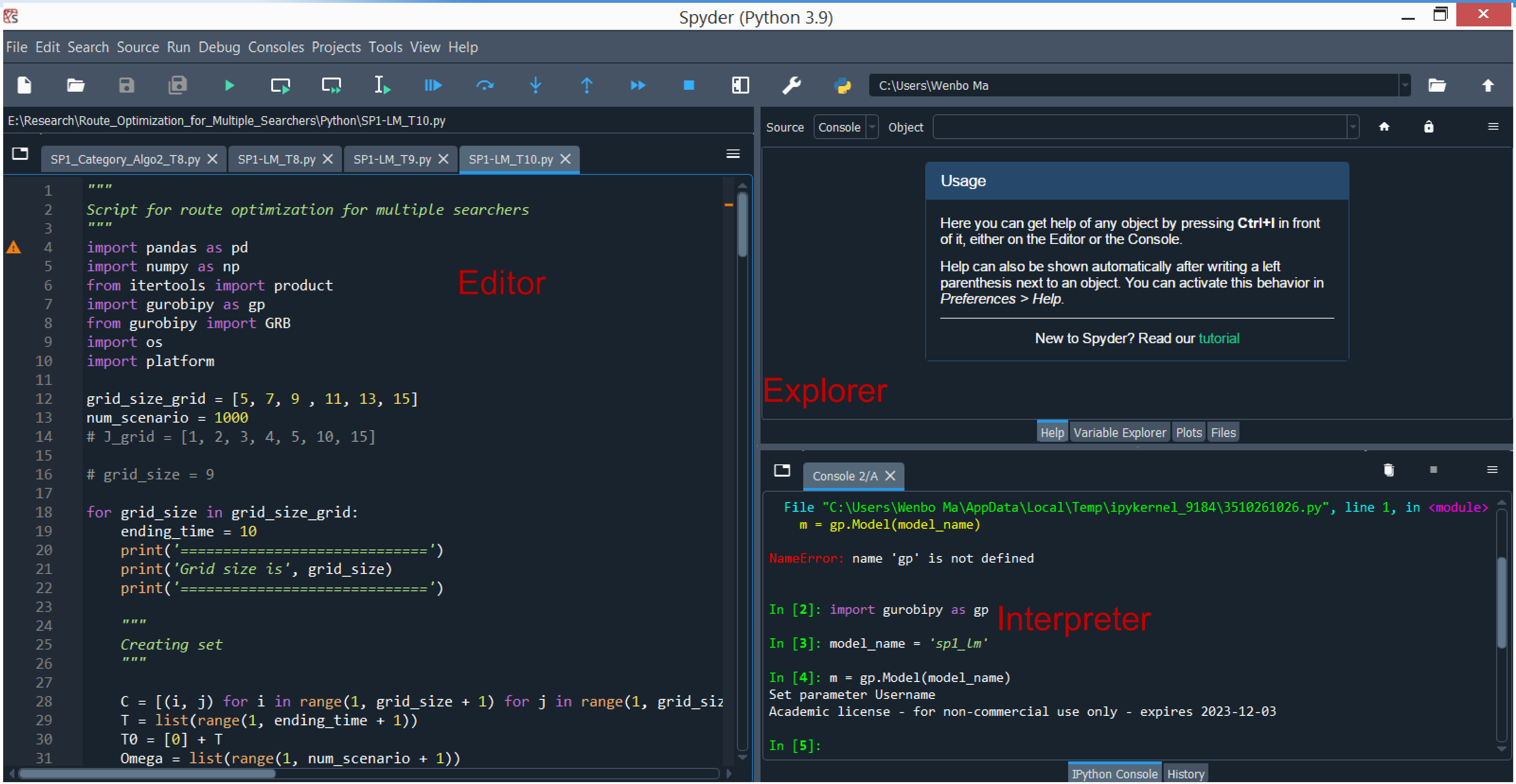

Figure sourced from Hsuan-Tien Lin's Machine Learning Techniques Course

Writing a Model in Python

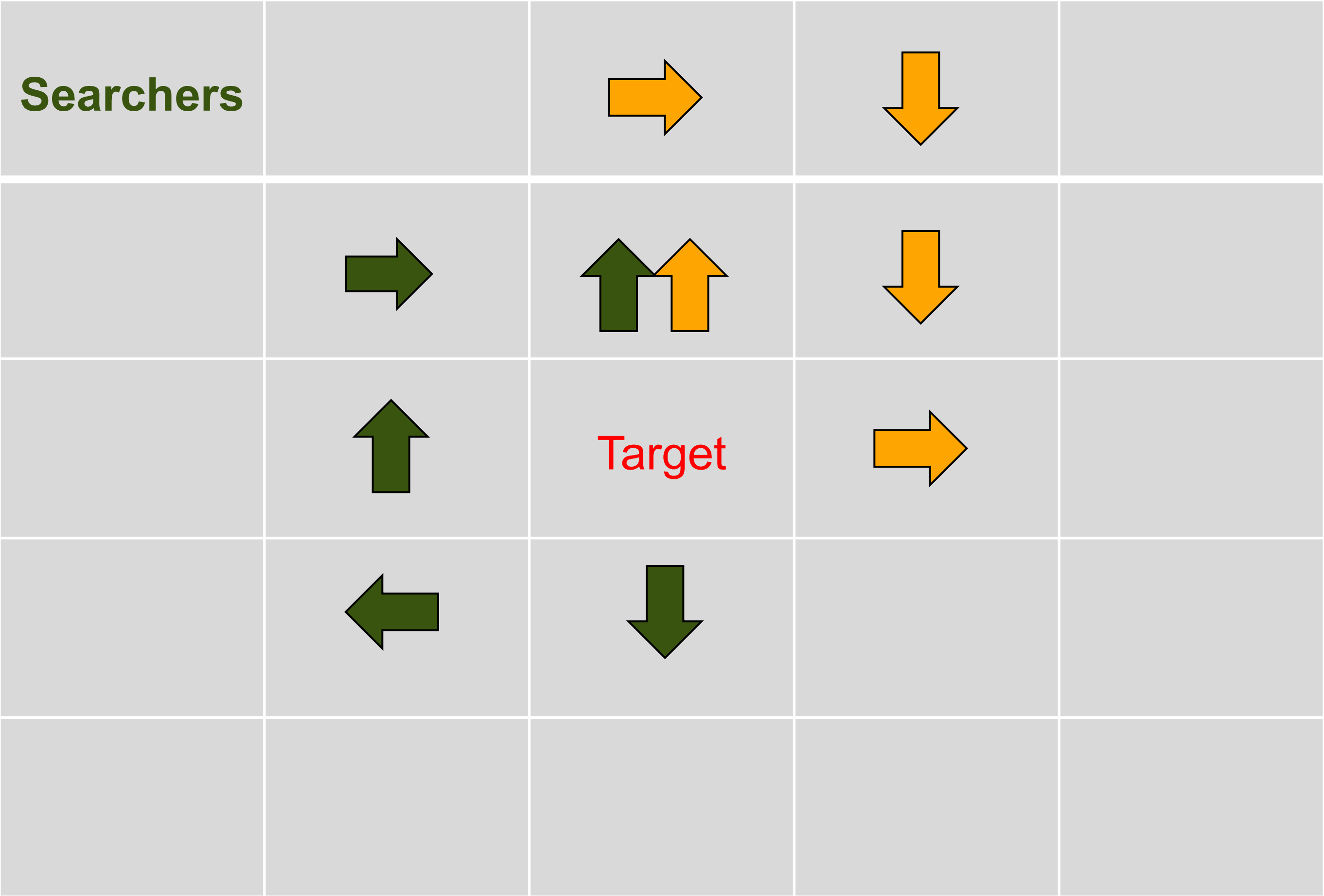
- Environment: Spyder (included in the Anaconda distribution)
  - Spyder is an integrated development environment (IDE):
  - Code editor (syntax checking), interpreter (running interactively), debugger.
  - Similar to Rstudio, MATLAB, Eclipse(Java).
  - Other Popular IDE for Python:
    - PyCharm(Popular among Software Engineer, Not Lightweight),
    - VSCode,
    - Jupyter notebook (Easy for demo)



Writing a Model in Python



Writing a Model in Python



Writing a Model in Python

Formulation

$$\min \sum_{\omega \in \Omega} q(\omega) U_\omega$$

$$\text{s.t. } e^{-i\alpha}(1 + i - ie^{-\alpha})$$

$$+\frac{1}{\alpha} e^{-i\alpha}(e^{-\alpha} - 1) \sum_{c,t \in \mathcal{T}} \zeta_{c,t}(\omega) \alpha Z_{c,t} \le U_\omega \quad \forall\, \omega, i \qquad (19)$$

$$(14) - (18)$$

$$\text{s.t. } \sum_{c' \in \mathcal{R}(c)} X_{c',c,t-d_{c',c}} = \sum_{c' \in \mathcal{F}(c)} X_{c,c',t} \quad \forall\, c, t \in \mathcal{T} \qquad (14)$$

$$\sum_{c' \in \mathcal{F}(c)} X_{c,c',0} = x_{c,0} \quad \forall\, c \qquad (15)$$

$$\sum_{c' \in \mathcal{R}(c)} X_{c',c,t-d_{c',c}} = Z_{c,t} \quad \forall\, c, t \in \mathcal{T} \qquad (16)$$

$$X_{c,c',t} \ge 0 \quad \forall\, c, c', t \qquad (17)$$

$$Z_{c,t} \in \{0, 1, 2, \ldots, J\} \quad \forall\, c, t \in \mathcal{T} \qquad (18)$$

Context: There is a two-dimensional space denoted by C where there is a target moving according to a path ω from t = 1 to T. Our goal is to place several searchers in the space and obtain a search plan that minimize the expected non-detection probability.

Sets and index:
- C and c: cell index. c = (1, 2) means row 1 and columns 2.
- T and t: time
- Ω and ω: a path taken by the target
- J: number of searchers
- I and i: I = J * T

Parameters:
- α: Detection rate when the searchers and the target encounter at the same cell
- $\zeta_{c,t}(\omega)$: Whether the target appears at cell c at time t for scenario ω.
- $q(\omega)$: The probability that the target takes path ω

Decision Variables:
- U_ω: Auxiliary variable equals to non-detection probability when the target takes path ω
- Z_{ct} : Number of searchers at cell c for time t.
- $X_{c,c',t}$: Number of searchers at cell from time t who will move to cell c' in the next time period

Constraints:
- (19) LHS is the non-detection probability for scenario ω.
- (14) Making sure the searchers' move is continuous and cannot jump between periods.
- (15) Setting initial conditions for where the searchers are.
- (16) Linking X to Z

Import Python Library and Set Up the Environment

```python
"""
Script for route optimization for multiple searchers
"""


"""
######### Import libraries
"""


import pandas as pd
import numpy as np
from itertools import product
import gurobipy as gp
from gurobipy import GRB
import os
import platform
```

```python
"""
    Setting parameters for the searching environment
"""

grid_size = 9
ending_time = 7
num_scenario = 1000

if platform.system() == 'Windows':
    data_folder = 'E:\\Research\\Route_Optimization_for_Multiple_Searchers\\Python\\'
else:
    data_folder = os.path.dirname(os.path.realpath(__file__))
```

Formulation

$$\min \sum_{\omega \in \Omega} q(\omega) U_\omega$$

$$\text{s.t. } e^{-i\alpha}(1 + i - ie^{-\alpha})$$

$$+\frac{1}{\alpha} e^{-i\alpha}(e^{-\alpha} - 1) \sum_{c,t \in \mathcal{T}} \zeta_{c,t}(\omega) \alpha Z_{c,t} \leq U_\omega \quad \forall\, \omega, i \tag{19}$$

$$(14) - (18)$$

$$\text{s.t. } \sum_{c' \in \mathcal{R}(c)} X_{c',c,t-d_{c',c}} = \sum_{c' \in \mathcal{F}(c)} X_{c,c',t} \quad \forall\, c, t \in \mathcal{T} \tag{14}$$

$$\sum_{c' \in \mathcal{F}(c)} X_{c,c',0} = x_{c,0} \quad \forall\, c \tag{15}$$

$$\sum_{c' \in \mathcal{R}(c)} X_{c',c,t-d_{c',c}} = Z_{c,t} \quad \forall\, c, t \in \mathcal{T} \tag{16}$$

$$X_{c,c',t} \geq 0 \quad \forall\, c, c', t \tag{17}$$

$$Z_{c,t} \in \{0, 1, 2, \ldots, J\} \quad \forall\, c, t \in \mathcal{T} \tag{18}$$

Create Sets and Parameters

```
"""
Creating set for the optimization model
"""
C = [(i, j) for i in range(1, grid_size + 1) for j in range(1, grid_size + 1)]
T = list(range(1, ending_time + 1))
T0 = [0] + T
Omega = list(range(1, num_scenario + 1))
J = 3
I = list(range(0, J * ending_time + 1))

xx = {} # this is the variable x in the formulation
for c in C:
    for t in T0:
        if c == (1, 1) and t == 0:
            xx[c, t] = J
        else:
            xx[c, t] = 0
```

- range: starting and ending
- list comprehension

Formulation

$$\min \sum_{\omega\in\Omega} q(\omega)U_\omega$$

$$\text{s.t. } e^{-i\alpha}(1+i-ie^{-\alpha})$$

$$+\frac{1}{\alpha}e^{-i\alpha}(e^{-\alpha}-1)\sum_{c,t\in\mathcal{T}} \zeta_{c,t}(\omega)\alpha Z_{c,t} \le U_\omega \quad \forall\, \omega, i \qquad (19)$$

$$(14)-(18)$$

$$\text{s.t.} \sum_{c'\in\mathcal{R}(c)} X_{c',c,t-d_{c',c}} = \sum_{c'\in\mathcal{F}(c)} X_{c,c',t} \quad \forall\, c, t\in\mathcal{T} \qquad (14)$$

$$\sum_{c'\in\mathcal{F}(c)} X_{c,c',0} = x_{c,0} \quad \forall\, c \qquad (15)$$

$$\sum_{c'\in\mathcal{R}(c)} X_{c',c,t-d_{c',c}} = Z_{c,t} \quad \forall\, c, t\in\mathcal{T} \qquad (16)$$

$$X_{c,c',t} \ge 0 \quad \forall\, c, c', t \qquad (17)$$

$$Z_{c,t} \in \{0, 1, 2, \ldots, J\} \quad \forall\, c, t\in\mathcal{T} \qquad (18)$$

Creating Sets and Parameters

```python
# zeta_raw = pd.read_csv(r'C:\Users\Wenbo Ma\Desktop\Route Optimization\Python\SP1-L\Zeta.csv', he
zeta_raw = pd.read_csv(data_folder + '/Zeta.csv', header = None, index_col = 0)
Zeta = {}
for path in range(1, zeta_raw.shape[0] + 1):
    for t in range(1, ending_time + 1):
        all_cells = C
        for cell in all_cells:
            Zeta[(cell, t, path)] = 0 # set Zeta equal to 0
        cell_one_dim = zeta_raw.loc[path, 3 * (t - 1) + 1] # extract the occupied cell loc from Ze
        cell_two_dim = (cell_one_dim // grid_size + 1, np.mod(cell_one_dim, grid_size))
        Zeta[(cell_two_dim, t, path)] = 1 # set Zeta equal to 1 for occupied cell

np.random.seed(2022)
q = np.random.uniform(low = 0, high = 1, size = num_scenario)
q = q / sum(q) # normalize to a probablity distribution summing up to 1
q = dict(zip(Omega, q))
alpha = -3 * np.log(0.4) / J
```

- Python dictionary – key value pairs

```
In [20]: Zeta
Out[20]:
{((1, 1), 1, 1): 0,
 ((1, 2), 1, 1): 0,
 ((1, 3), 1, 1): 0,
 ((1, 4), 1, 1): 0,
 ((1, 5), 1, 1): 0,
 ((1, 6), 1, 1): 0,
 ((1, 7), 1, 1): 0,
 ((1, 8), 1, 1): 0,
 ((1, 9), 1, 1): 0,
```

Formulation

$$\min \sum_{\omega\in\Omega} q(\omega)U_\omega$$

$$\text{s.t. } e^{-i\alpha}(1+i-ie^{-\alpha})$$

$$+\frac{1}{\alpha}e^{-i\alpha}(e^{-\alpha}-1)\sum_{c,t\in\mathcal{T}} \zeta_{c,t}(\omega)\alpha Z_{c,t} \le U_\omega \quad \forall\, \omega, i \qquad (19)$$

$$(14)-(18)$$

$$\text{s.t. } \sum_{c'\in\mathcal{R}(c)} X_{c',c,t-d_{c',c}} = \sum_{c'\in\mathcal{F}(c)} X_{c,c',t} \quad \forall\, c, t\in\mathcal{T} \qquad (14)$$

$$\sum_{c'\in\mathcal{F}(c)} X_{c,c',0} = x_{c,0} \quad \forall\, c \qquad (15)$$

$$\sum_{c'\in\mathcal{R}(c)} X_{c',c,t-d_{c',c}} = Z_{c,t} \quad \forall\, c, t\in\mathcal{T} \qquad (16)$$

$$X_{c,c',t} \ge 0 \quad \forall\, c, c', t \qquad (17)$$

$$Z_{c,t} \in \{0,1,2,\ldots,J\} \quad \forall\, c, t\in\mathcal{T} \qquad (18)$$

Creating a New Model

```
"""
"""
model_name = 'sp1_l_new_formul'
m = gp.Model(model_name)
m.setParam(GRB.Param.TimeLimit, 15 * 60)
m.setParam(GRB.Param.Threads, 1)
m.setParam(GRB.Param.LogFile, model_name)
    # m.setParam(GRB.Param.NumericFocus,1)
```

- model name, time limit, threads…

Formulation

$$\min \sum_{\omega \in \Omega} q(\omega) U_\omega$$

$$\text{s.t. } e^{-i\alpha}(1 + i - ie^{-\alpha})$$

$$+\frac{1}{\alpha} e^{-i\alpha}(e^{-\alpha} - 1) \sum_{c,t \in \mathcal{T}} \zeta_{c,t}(\omega) \alpha Z_{c,t} \leq U_\omega \quad \forall\, \omega, i \qquad (19)$$

$$(14) - (18)$$

$$\text{s.t. } \sum_{c' \in \mathcal{R}(c)} X_{c',c,t-d_{c',c}} = \sum_{c' \in \mathcal{F}(c)} X_{c,c',t} \quad \forall\, c, t \in \mathcal{T} \qquad (14)$$

$$\sum_{c' \in \mathcal{F}(c)} X_{c,c',0} = x_{c,0} \quad \forall\, c \qquad (15)$$

$$\sum_{c' \in \mathcal{R}(c)} X_{c',c,t-d_{c',c}} = Z_{c,t} \quad \forall\, c, t \in \mathcal{T} \qquad (16)$$

$$X_{c,c',t} \geq 0 \quad \forall\, c, c', t \qquad (17)$$

$$Z_{c,t} \in \{0, 1, 2, \ldots, J\} \quad \forall\, c, t \in \mathcal{T} \qquad (18)$$

Defining Decisions Variables

```python
"""
Defining decision variables
"""
sub_X = list(product(C, C, T0))
sub_Z = list(product(C, T))
# ==========================================================================
#     sub_WW = list(product(C, T))
#     sub_WW = [each for each in sub_WW if W[each] == 1]
# ==========================================================================

X = m.addVars(sub_X, lb = 0, name = 'X')
Z = m.addVars(sub_Z, lb = 0, ub = J, vtype = GRB.INTEGER, name = 'Z')
# Z_New = m.addVars(sub_WW, lb = 0, ub = J, vtype = GRB.INTEGER, name = 'Z')
U = m.addVars(Omega, lb = 0, name = 'U')
```

set/index, bounds, name, type

Formulation

$$\min \sum_{\omega \in \Omega} q(\omega) U_\omega$$

$$\text{s.t. } e^{-i\alpha}(1 + i - ie^{-\alpha}) + \frac{1}{\alpha} e^{-i\alpha}(e^{-\alpha} - 1) \sum_{c,t \in \mathcal{T}} \zeta_{c,t}(\omega) \alpha Z_{c,t} \leq U_\omega \quad \forall\, \omega, i \tag{19}$$

$$(14) - (18)$$

$$\text{s.t. } \sum_{c' \in \mathcal{R}(c)} X_{c',c,t-d_{c',c}} = \sum_{c' \in \mathcal{F}(c)} X_{c,c',t} \quad \forall\, c, t \in \mathcal{T} \tag{14}$$

$$\sum_{c' \in \mathcal{F}(c)} X_{c,c',0} = x_{c,0} \quad \forall\, c \tag{15}$$

$$\sum_{c' \in \mathcal{R}(c)} X_{c',c,t-d_{c',c}} = Z_{c,t} \quad \forall\, c, t \in \mathcal{T} \tag{16}$$

$$X_{c,c',t} \geq 0 \quad \forall\, c, c', t \tag{17}$$

$$Z_{c,t} \in \{0, 1, 2, \ldots, J\} \quad \forall\, c, t \in \mathcal{T} \tag{18}$$

Defining Objective and Constraints

```
"""
Defining objective function
"""
m.setObjective(sum(q[omega] * U[omega] for omega in Omega), GRB.MINIMIZE)

"""
Defining constraints
"""

m.addConstrs((np.exp(-i * alpha) * (1 + i - i * np.exp(-alpha)) + np.exp(-i * alpha) * (np.exp(-alpha) - 1) * sum(Zeta[c,
m.addConstrs((sum(X[c_prime, c, t - 1] for c_prime in C if is_nearby_cell(c, c_prime)) == sum(X[c, c_prime, t] for c_prime
m.addConstrs((sum(X[c, c_prime, 0] for c_prime in C if is_nearby_cell(c, c_prime)) == xx[c, 0] for c in C), name = '15') #
m.addConstrs((sum(X[c_prime, c, t - 1] for c_prime in C if is_nearby_cell(c, c_prime)) == Z[c, t] for c in C for t in T),
# m.addConstrs((sum(X[c_prime, sub[0], sub[1] - 1] for c_prime in C if is_nearby_cell(sub[0], c_prime)) == Z_New[sub] for
```

Objective and constraints

Formulation

$$\min \sum_{\omega \in \Omega} q(\omega) U_\omega$$

$$\text{s.t. } e^{-i\alpha}(1 + i - ie^{-\alpha})$$

$$+\frac{1}{\alpha} e^{-i\alpha}(e^{-\alpha} - 1) \sum_{c,t \in \mathcal{T}} \zeta_{c,t}(\omega) \alpha Z_{c,t} \leq U_\omega \quad \forall\, \omega, i \qquad (19)$$

$$(14) - (18)$$

$$\text{s.t. } \sum_{c' \in \mathcal{R}(c)} X_{c',c,t-d_{c',c}} = \sum_{c' \in \mathcal{F}(c)} X_{c,c',t} \quad \forall\, c, t \in \mathcal{T} \qquad (14)$$

$$\sum_{c' \in \mathcal{F}(c)} X_{c,c',0} = x_{c,0} \quad \forall\, c \qquad (15)$$

$$\sum_{c' \in \mathcal{R}(c)} X_{c',c,t-d_{c',c}} = Z_{c,t} \quad \forall\, c, t \in \mathcal{T} \qquad (16)$$

$$X_{c,c',t} \geq 0 \quad \forall\, c, c', t \qquad (17)$$

$$Z_{c,t} \in \{0, 1, 2, \ldots, J\} \quad \forall\, c, t \in \mathcal{T} \qquad (18)$$

Running and Extracting Results

```python
""" Solving
"""
m.optimize()

""" Extracting optimal solution, bound, objective value
"""
for key, val in Z.items():
    if val.X != 0:
        print('key is', key, 'value is', val.X)

print('best objective value is', m.objVal)
print('best bound is', m.objBound)
```

```
Root relaxation: objective 5.987964e-01, 829 iterations, 0.03 seconds (0.03 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    0.59880    0   13    1.00000    0.59880  40.1%     -    0s
H    0     0                       0.6013103    0.59880  0.42%     -    0s
     0     0    0.59955    0   10    0.60131    0.59955  0.29%     -    0s
     0     0    0.59955    0    7    0.60131    0.59955  0.29%     -    0s
*    0     0               0         0.6008911    0.60089  0.00%     -    0s

Cutting planes:
  Gomory: 3

Explored 1 nodes (1170 simplex iterations) in 0.20 seconds (0.10 work units)
Thread count was 1 (of 4 available processors)

Solution count 3: 0.600891 0.60131 1

Optimal solution found (tolerance 1.00e-04)
Best objective 6.008910521752e-01, best bound 6.008910521752e-01, gap 0.0000%
```

```
In [24]: m.optimize()
Gurobi Optimizer version 10.0.0 build v10.0.0rc2 (win64)

CPU model: Intel(R) Core(TM) i5-4300U CPU @ 1.90GHz, instruction set [SSE2|AVX|AVX2]
Thread count: 2 physical cores, 4 logical processors, using up to 1 threads

Optimize a model with 5615 rows, 53255 columns and 43885 nonzeros
Model fingerprint: 0x4a2eb4a5
Variable types: 52688 continuous, 567 integer (0 binary)
Coefficient statistics:
  Matrix range     [3e-09, 1e+00]
  Objective range  [4e-05, 1e-02]
  Bounds range     [3e+00, 3e+00]
  RHS range        [6e-08, 3e+00]
Found heuristic solution: objective 1.0000000
Presolve removed 4185 rows and 52646 columns
Presolve time: 0.06s
Presolved: 1430 rows, 609 columns, 5415 nonzeros
Variable types: 473 continuous, 136 integer (0 binary)
```

```
key is ((1, 1), 1) value is 1.0
key is ((1, 2), 1) value is 2.0
key is ((1, 2), 2) value is 1.0
key is ((1, 3), 2) value is 2.0
key is ((1, 3), 3) value is 1.0
key is ((1, 4), 3) value is 2.0
key is ((2, 3), 4) value is 1.0
key is ((2, 4), 4) value is 2.0
key is ((2, 4), 5) value is 2.0
key is ((2, 4), 6) value is 1.0
key is ((2, 4), 7) value is 1.0
key is ((2, 5), 6) value is 1.0
key is ((2, 6), 7) value is 1.0
key is ((3, 3), 5) value is 1.0
key is ((3, 4), 6) value is 1.0
key is ((3, 5), 7) value is 1.0
best objective value is 0.6008910521751948
best bound is 0.6008910521751948
```

Interactive Mode and Command Line Mode

Demo:

- Interactive Mode (Line by Line): Code developing and debugging
- Command Line Mode: Generating testing results
- Jupyter Notebook