Heimadæmi3 (1)

September 28, 2023

[62]:

```
import gurobipy as gp
import numpy as np
from gurobipy import GRB
import pandas as pd

#gögn frá repository team 1
dico = {0:"Rome",1:"Venice",2:"Madrid",3:"Barcelona",4:"Lisbon",5:"London",6:
 ↪"Berlin",7:"Hamburg",8:"Budapest",9:"Amsterdam",10:"Paris",11:"Vienna"}
nb_cities = len(dico)

infi=10000
enjoyement = np.array([4.5, 5.9, 8.4, 5.8, 10, 2.7, 7.4, 5.2, 9.3, 7, 4.1, 6])
hostel = np.array([44, 52, 29, 24, 16, 23, 30, 25, 10, 27, 44, 36])
flights = np.array([[93, infi, infi, 105, infi, 64, 137, infi, infi, 94, 110,␣
 ↪infi],
                    [280, infi, infi, 260, infi, 180, 210, infi, infi, 190,␣
 ↪210, infi]])



nb_days = 7
budget = 1500
rest_time = 6
transit = 3
same=100000
travel_time = np.array([[same, 280, infi, infi, 260, infi, 180, 210, infi,␣
 ↪infi, 190, 210, infi],
                 [280, same, 239, 1045, 879, 1808, 868, 919, 984, 933, 930,␣
 ↪654, 732],
                 [infi, 239, same, 1002, 836, 1765, 834, 769.0, 941, 667, 887,␣
 ↪611, 466],
                 [infi, 1045, 1002, same, 150, 1005, 829, 1188, 1172, 1581,␣
 ↪891, 615, 1365],
                 [260, 879, 836, 150, same, 795, 619, 978, 953, 1346, 681, 405,␣
 ↪1164],
                 [infi, 1808, 1765, 1005, 795, same, 1943, 1684, 1608, 2412,␣
 ↪1336, 1064, 1756],
                        [180, 868, 834, 829, 619, 1943, same, 569, 553, 1207, 251,␣
 ↪139, 1006],
                        [210, 919, 769, 1188, 978, 1684, 569, same, 106, 737, 397,␣
 ↪532, 536],
                        [infi, 984, 941, 1172, 953, 1608, 553, 106, same, 902, 350,␣
 ↪507, 701],
                        [infi, 933, 667, 1581, 1346, 2412, 1207, 737, 902, same, 919,␣
 ↪797, 146],
                        [190, 930, 887, 891, 681, 1336, 251, 397, 350, 919, same, 205,␣
 ↪763],
                        [210, 654, 611, 615, 405, 1064, 139, 532, 507, 797, 205, same,␣
 ↪608],
                        [infi, 732, 466, 1365, 1164, 1756, 1006, 536, 701, 146, 763,␣
 ↪608, same]])
```

[63]:

```
# Create the model
m = gp.Model()

x = {}
y = {}

# Add decision variables
for day in range(nb_days):
    for city in range(nb_cities):
        x[day, city] = m.addVar(vtype=GRB.BINARY, name=f"x{day}{city}")
        y[day, city] = m.addVar(lb = 0.0, vtype=GRB.CONTINUOUS,␣
 ↪name=f"t{day}{city}")



# Set the objective

#ákvað að margfalda enjoyment með tímanum sem er eitt á staðnum
m.setObjective(gp.quicksum((enjoyement[city]*x[day,␣
 ↪city]*(24-rest_time-(travel_time[city][city-1])/24) + y[day, city])
                               for day in range(nb_days)
                               for city in range(nb_cities)), GRB.MAXIMIZE)
# Add constraints



m.addConstr(gp.quicksum(flights[0][city]*x[0, city] for city in␣
 ↪range(nb_cities)) +
                gp.quicksum(flights[1][city]*x[nb_days-1, city] for city in␣
 ↪range(nb_cities)) +
                gp.quicksum(hostel[city]*x[day, city] for day in range(nb_days)␣
 ↪for city in range(nb_cities)) <= budget, name="Budget")
```

```python
m.addConstr(gp.quicksum(x[day, city] for day in range(nb_days) for city in range(nb_cities)) == nb_days, name=f"Visit on day{day}")

# One time maximum in a city
for city in range(nb_cities):
    m.addConstr(gp.quicksum(x[day, city] for day in range(nb_days)) <= 1, name="One time max per city")

# Exactly one city per day
for day in range(nb_days):
    m.addConstr(gp.quicksum(x[day, city] for city in range(nb_cities)) == 1, name="One city per day")
for day in range(nb_days):
    for city in range(nb_cities):
        m.addConstr(y[day, city] <= (24-rest_time)*60, name=f"Visit time max on {day} in {city}")

for day in range(nb_days):
    for city in range(nb_cities):
        if day == 0:
            m.addConstr(y[day, city] + flights[0][city]*x[0, city] <= (24-rest_time-transit)*60)
        elif day == 6:
            m.addConstr(y[day, city] + flights[1][city]*x[0, city] <= (24-rest_time-transit)*60 )
        else:
            m.addConstr(y[day, city] + gp.quicksum(travel_time[city][city_yest] * x[day, city] * x[day-1, city_yest] for city_yest in range(nb_cities)) <= (24-rest_time)*60)


# Optimize model
m.optimize()

if m.status == GRB.OPTIMAL:
    for day in range(nb_days):
        for city in range(nb_cities):
            if x[day, city].x > 0.1:
                time = y[day, city].x
                loc = dico[city]
                print(f"Day {day+1} in {loc} for {time/60:.1f} hours.")
```

```
Gurobi Optimizer version 10.0.3 build v10.0.3rc0 (win64)

CPU model: AMD Ryzen 7 5800H with Radeon Graphics, instruction set
[SSE2|AVX|AVX2]
```

```
Thread count: 8 physical cores, 16 logical processors, using up to 16 threads

Optimize a model with 129 rows, 168 columns and 468 nonzeros
Model fingerprint: 0xf06fb107
Model has 60 quadratic constraints
Variable types: 84 continuous, 84 integer (84 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+04]
  QMatrix range    [1e+02, 1e+05]
  QLMatrix range   [1e+00, 1e+00]
  Objective range  [1e+00, 2e+03]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 2e+03]
  QRHS range       [1e+03, 1e+03]
Presolve removed 110 rows and 36 columns
Presolve time: 0.00s
Presolved: 727 rows, 780 columns, 2796 nonzeros
Variable types: 0 continuous, 780 integer (720 binary)
Found heuristic solution: objective 82927.383333

Root relaxation: objective 8.641213e+04, 19 iterations, 0.00 seconds (0.00 work
units)

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

     0     0 86412.1250    0   17 82927.3833 86412.1250  4.20%     -    0s
H    0     0                    83106.941667 86343.5708  3.89%     -    0s
     0     0 86266.4444    0   15 83106.9417 86266.4444  3.80%     -    0s
H    0     0                    83718.470833 86266.4444  3.04%     -    0s
     0     0 85681.3750    0   21 83718.4708 85681.3750  2.34%     -    0s
H    0     0                    84370.108333 85681.3750  1.55%     -    0s
H    0     0                    84781.108333 85681.3750  1.06%     -    0s
     0     0 85442.6917    0   26 84781.1083 85442.6917  0.78%     -    0s
     0     0 85442.1639    0   24 84781.1083 85442.1639  0.78%     -    0s
     0     0 85431.6344    0   33 84781.1083 85431.6344  0.77%     -    0s
     0     0 85413.0490    0   47 84781.1083 85413.0490  0.75%     -    0s
     0     0 85273.4509    0   52 84781.1083 85273.4509  0.58%     -    0s
     0     0 85273.4509    0   18 84781.1083 85273.4509  0.58%     -    0s
     0     0 85273.4509    0   33 84781.1083 85273.4509  0.58%     -    0s
     0     0 85238.6083    0   21 84781.1083 85238.6083  0.54%     -    0s
     0     0 85171.6083    0   22 84781.1083 85171.6083  0.46%     -    0s
     0     0 85078.5847    0   36 84781.1083 85078.5847  0.35%     -    0s
     0     0 85077.7750    0   27 84781.1083 85077.7750  0.35%     -    0s
     0     0 85044.1083    0   22 84781.1083 85044.1083  0.31%     -    0s
     0     0 84991.8017    0   39 84781.1083 84991.8017  0.25%     -    0s
H    0     1                    84828.108333 84991.8017  0.19%     -    0s
```

```
Cutting planes:
  Gomory: 1
  Implied bound: 1
  Clique: 3
  MIR: 4
  Zero half: 2
  RLT: 45
  BQP: 2

Explored 1 nodes (543 simplex iterations) in 0.23 seconds (0.05 work units)
Thread count was 16 (of 16 available processors)

Solution count 6: 84828.1 84781.1 84370.1 … 82927.4

Optimal solution found (tolerance 1.00e-04)
Best objective 8.482810833333e+04, best bound 8.482810833333e+04, gap 0.0000%
Day 1 in Barcelona for 13.2 hours.
Day 2 in Lisbon for 15.5 hours.
Day 3 in Vienna for 11.2 hours.
Day 4 in Paris for 14.6 hours.
Day 5 in Budapest for 12.2 hours.
Day 6 in Hamburg for 16.2 hours.
Day 7 in London for 15.0 hours.
```