Optimization Code

October 26, 2023

1 Optimization Project

[ ]:
```
from gurobipy import *
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

1.1 Question 2

1.1.1 (a)

[ ]:
```
# importing data, used small datasets
distance_data = pd.read_csv('cities_small.csv')
packages_data = pd.read_csv('packages_small.csv')
```

[ ]:
```
distance_data
```

[ ]:
```
packages_data.head()
```

[ ]:
```
# using euclidean formula for computing distances between each city
def distance(lat1, lon1, lat2, lon2):
    dis = np.sqrt((lat1 - lat2)**2 + (lon1 - lon2)**2)
    return(dis)

# computing the distance matrix
cities = len(distance_data)
distance_matrix = np.zeros((cities, cities))

for i in range(cities):
    for j in range(cities):
        distance_matrix[i][j] = distance(distance_data.iloc[i]['lat'],␣
 ↪distance_data.iloc[i]['lon'],
                                         distance_data.iloc[j]['lat'],␣
 ↪distance_data.iloc[j]['lon'])

len(distance_matrix)
d = distance_matrix
d[7]
```

[ ]:
```
# creating a matrix that stores the number of packages between each city
packages_matrix = np.zeros((cities, cities))

# populating the 0 matrix with the number of packages in each trip using␣
 ↪cities' id
for i, row in packages_data.iterrows():
    origin_city = row['origin']
    destination_city = row['destination']
    packages_matrix[origin_city][destination_city] = row['packages']

len(packages_matrix)
f = packages_matrix
```

[ ]:
```
# create a model
m = Model("hub_and_spoke")

# number of hubs
k = 2

# discount factor alpha
a = 0.75

# decision variables
x = m.addVars(cities, cities, vtype = GRB.BINARY, name = "x")  # Spoke-to-Hub␣
 ↪and Hub-to-Spoke
h = m.addVars(cities, cities, vtype = GRB.BINARY, name = "h")  # Hub-to-Hub

# objective function
obj = LinExpr()

spoke_to_hub = sum(f[i][j] * d[i][j] * x[i,j] for i in range(cities) for j in␣
 ↪range(cities))
hub_to_hub = a * sum(f[i][j] * d[i][j] * h[i,j] for i in range(cities) for j in␣
 ↪range(cities))
hub_to_spoke = sum(f[i][j] * d[i][j] * x[i,j] for i in range(cities) for j in␣
 ↪range(cities))

# obj = sum(f[i][j] * d[i][j] * x[i,j] for i in range(cities) for j in␣
 ↪range(cities)) + \
#       a * sum(f[i][j] * d[i][j] * h[i,j] for i in range(cities) for j in␣
 ↪range(cities))

obj = spoke_to_hub + hub_to_hub + hub_to_spoke

m.setObjective(obj, GRB.MINIMIZE)

### constraints
```

```
# each city is connected to only one hub
for i in range(cities):
    m.addConstr(sum(x[i,j] for j in range(cities)) == 1)

# a city can only be connected to a hub if that city is designated as a hub
for i in range(cities):
    for j in range(cities):
        m.addConstr(x[i,j] <= h[j,j])

# ensure that there's a direct path between two hubs
for i in range(cities):
    for j in range(cities):
        m.addConstr(h[i,j] <= h[i,i])
        m.addConstr(h[i,j] <= h[j,j])

# ensure that a city can not be connected to itself
for i in range(cities):
    m.addConstr(x[i, i] == 0)

# set the number of hubs = 2
m.addConstr(sum(h[i,i] for i in range(cities)) == k)

# solve the model
m.optimize()
```

1.1.2 (b)

[ ]:
```
hub_city_ids = [i for i in range(cities) if h[i,i].x > 0]
hub_city_ids
```

[ ]:
```
connections = [[i, j] for i in range(cities) for j in range(cities) if x[i,j].x␣
↪> 0]
connections
```

[ ]:
```
plt.figure(figsize=(10, 6))

# plot all cities
plt.scatter(distance_data['lon'], distance_data['lat'], label='spoke')

# plot the hub with a different color
label = 1
for i in hub_city_ids:
    hub_row = distance_data[distance_data['id'] == i].iloc[0]
    if label == 1:
        plt.scatter(hub_row['lon'], hub_row['lat'], c='red', label='hub')
        label -= 1
```

```
    else:
        plt.scatter(hub_row['lon'], hub_row['lat'], c='red')

# mark the ids on plot
for i, row in distance_data.iterrows():
    plt.text(row['lon'], row['lat'], str(row['id']))

# plot the links between hobs and spokes
for connection in connections:
    spoke, hub = connection
    spoke_row = distance_data[distance_data['id'] == spoke].iloc[0]
    hub_row = distance_data[distance_data['id'] == hub].iloc[0]
    plt.plot([spoke_row['lon'], hub_row['lon']], [spoke_row['lat'],␣
 ↪hub_row['lat']], c='gray', linestyle='--')

plt.xlabel('longitude')
plt.ylabel('latitude')
plt.legend()
plt.grid(True)
plt.show()
```

[ ]: