Final Integration Assignment

Cheryl Ngo

10/16/2019

Prepare the Data

Import Distance Data from MySQL Workbench

In [106]:
```
import mysql.connector

from gurobipy import *

db=mysql.connector.connect(user='root',password='root',host='localhost',databa
se='sharkTank')

#Create cursor
cur=db.cursor()

#This returns all data in the table
cur.execute('select * from mileage')
mileage=cur.fetchall()

db.close()
```

Format Distance Data as a Dataframe

In [107]:
```
import pandas as pd
y=pd.DataFrame(mileage,columns=["Source","Destination","Distance"])
```

Drop Missing Values

In [108]:
```
z = y.drop(y[(y.Source=="Pearl.City..HI")|
             (y.Source=="East.Honolulu..HI")|
             (y.Source=="Honolulu..HI")|
             (y.Destination=="Pearl.City..HI")|
             (y.Destination=="East.Honolulu..HI")|
             (y.Destination=="Honolulu..HI")].index)
z=z.reset_index(drop=True)
```

Create the Gurobi Model

Set the demand

In [109]:
```
demand=[1051,940,1131,466,1301,1171,1463,1120,665,1280,615,528]
demandLoc=["Boston..MA","Chicago..IL","Dallas..TX","Denver..CO","Los.Angeles..
CA","Miami..FL",
           "New.York.City..NY","Phoenix..AZ","Pittsburgh..PA","Richmond..VA",
"San.Francisco..CA","Seattle..WA"]
```

Minimize the data frame

In [110]:
```
z=z[z['Destination'].isin(demandLoc)]
z = z.reset_index(drop=True)
```

Create a List of Cities Corresponding to the Decision Variable Indexes

In [127]:
```
cities=list(z.iloc[0:len(set(z.Source)),0])
```

Initialize the Model and Add Decision Variables

In [113]:
```
from gurobipy import *

#Create model
m=Model('hubs')

#Use array to create decision variables
dvars=[]

a=[]
for i in range(len(cities)): #Hub indicator
    a.append(m.addVar(vtype=GRB.BINARY,name=cities[i]))
dvars.append(a)

for i in range(len(cities)):
    a=[]
    for j in range(len(demandLoc)):
        a.append(m.addVar(vtype=GRB.BINARY,name="%s %s"%(cities[i],demandLoc[j
])))
    dvars.append(a)

m.update()
```

Add constraints

In [115]:
```
#Create variable names
constr_names=["TotalHubs", "Total Assignment", "Hub Assignment Constraint"]
#Constraint LHS
constr_coef=[quicksum(dvars[0]),
             [quicksum(dvars[i][j] for i in range(1,len(cities)+1)) for j in r
ange(len(demandLoc))], #All to demand center
             [quicksum(dvars[i][j] for j in range(len(demandLoc))) for i in ra
nge(1,len(cities)+1)]] #All from location

#Constraint RHS
rhs=[3,1,[dvars[0][k]*len(demandLoc) for k in range(len(cities))]]
    #Element 1 as a list [1 for i in range(len(demandLoc))]

m.addConstr(constr_coef[0],GRB.EQUAL,rhs[0],constr_names[0])
for i in range(len(demandLoc)):
    m.addConstr(constr_coef[1][i],GRB.EQUAL,rhs[1],"%s to %s" %(constr_names[1
],demandLoc[i]))
for i in range(len(cities)):
    m.addConstr(constr_coef[2][i],GRB.LESS_EQUAL,rhs[2][i],"%s from %s"%(const
r_names[2],cities[i]))

m.update()
```

Create the objective function

In [116]:
```
m.setObjective(quicksum(quicksum(z.loc[(z["Destination"]==demandLoc[j]),"Dista
nce"]*[dvars[i][j] for i in range(1,len(cities)+1)])*demand[j] for j in range(
len(demandLoc))))
```

Run the Model

Minimize the Model and Output Results

In [117]:
```
m.ModelSense=GRB.MINIMIZE
m.update()
m.optimize()
```

```
Optimize a model with 1010 rows, 12961 columns and 25922 nonzeros
Variable types: 0 continuous, 12961 integer (12961 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+01]
  Objective range  [3e+03, 7e+06]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 3e+00]
Found heuristic solution: objective 1.744363e+07
Presolve time: 0.03s
Presolved: 1010 rows, 12961 columns, 25922 nonzeros
Variable types: 0 continuous, 12961 integer (12961 binary)

Root relaxation: objective 0.000000e+00, 26 iterations, 0.00 seconds

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

     0     0    0.00000    0   10 1.7444e+07    0.00000   100%     -    0s
H    0     0                      5045678.9600    0.00000   100%     -    0s
     0     0 40663.3436    0   13 5045678.96 40663.3436  99.2%     -    0s
     0     0 65139.6718    0   13 5045678.96 65139.6718  98.7%     -    0s
     0     0 79445.5927    0   13 5045678.96 79445.5927  98.4%     -    0s
     0     0 93633.5955    0   13 5045678.96 93633.5955  98.1%     -    0s
     0     0 101227.302    0   13 5045678.96 101227.302  98.0%     -    0s
     0     0 112699.796    0   13 5045678.96 112699.796  97.8%     -    0s
     0     0 122416.636    0   13 5045678.96 122416.636  97.6%     -    0s
     0     0 136225.017    0   13 5045678.96 136225.017  97.3%     -    0s
     0     0 144552.005    0   13 5045678.96 144552.005  97.1%     -    0s
     0     0 152510.880    0   13 5045678.96 152510.880  97.0%     -    0s
     0     0 165373.880    0   13 5045678.96 165373.880  96.7%     -    0s
     0     0 177071.392    0   13 5045678.96 177071.392  96.5%     -    0s
     0     0 187798.022    0   13 5045678.96 187798.022  96.3%     -    0s
     0     0 197377.651    0   13 5045678.96 197377.651  96.1%     -    0s
     0     0 218149.435    0   13 5045678.96 218149.435  95.7%     -    0s
     0     0 245238.535    0   13 5045678.96 245238.535  95.1%     -    0s
     0     0 262080.885    0   13 5045678.96 262080.885  94.8%     -    0s
     0     0 286233.018    0   13 5045678.96 286233.018  94.3%     -    0s
     0     0 327476.557    0   13 5045678.96 327476.557  93.5%     -    0s
     0     0 372775.780    0   13 5045678.96 372775.780  92.6%     -    0s
     0     0 442291.725    0   13 5045678.96 442291.725  91.2%     -    0s
     0     0 473040.440    0   13 5045678.96 473040.440  90.6%     -    0s
     0     0 499779.811    0   13 5045678.96 499779.811  90.1%     -    0s
     0     0 529011.983    0   13 5045678.96 529011.983  89.5%     -    0s
     0     0 546943.282    0   13 5045678.96 546943.282  89.2%     -    0s
     0     0 574946.188    0   13 5045678.96 574946.188  88.6%     -    0s
     0     0 644197.948    0   13 5045678.96 644197.948  87.2%     -    0s
     0     0 676761.727    0   13 5045678.96 676761.727  86.6%     -    0s
     0     0 714851.765    0   13 5045678.96 714851.765  85.8%     -    0s
     0     0 742033.867    0   13 5045678.96 742033.867  85.3%     -    0s
     0     0 756559.237    0   13 5045678.96 756559.237  85.0%     -    0s
     0     0 788630.837    0   13 5045678.96 788630.837  84.4%     -    0s
     0     0 842533.219    0   13 5045678.96 842533.219  83.3%     -    1s
     0     0 899011.262    0   13 5045678.96 899011.262  82.2%     -    1s
     0     0 947356.036    0   13 5045678.96 947356.036  81.2%     -    1s
     0     0 974168.666    0   13 5045678.96 974168.666  80.7%     -    1s
     0     0 1016389.72    0   13 5045678.96 1016389.72  79.9%     -    1s
     0     0 1075769.35    0   13 5045678.96 1075769.35  78.7%     -    1s
```

```
         0     0 1146152.32    0   13 5045678.96 1146152.32  77.3%     -    1s
         0     0 1200037.86    0   13 5045678.96 1200037.86  76.2%     -    1s
         0     0 1227533.30    0   13 5045678.96 1227533.30  75.7%     -    1s
         0     0 1261150.25    0   15 5045678.96 1261150.25  75.0%     -    1s
         0     0 1285937.41    0   13 5045678.96 1285937.41  74.5%     -    1s
         0     0 1330038.65    0   15 5045678.96 1330038.65  73.6%     -    1s
         0     0 1357596.95    0   15 5045678.96 1357596.95  73.1%     -    1s
         0     0 1387259.71    0   13 5045678.96 1387259.71  72.5%     -    1s
         0     0 1416514.32    0   17 5045678.96 1416514.32  71.9%     -    1s
         0     0 1435680.96    0   17 5045678.96 1435680.96  71.5%     -    1s
         0     0 1456198.63    0   17 5045678.96 1456198.63  71.1%     -    1s
         0     0 1476121.19    0   15 5045678.96 1476121.19  70.7%     -    1s
         0     0 1499276.66    0   15 5045678.96 1499276.66  70.3%     -    1s
         0     0 1522096.16    0   15 5045678.96 1522096.16  69.8%     -    1s
         0     0 1543595.78    0   15 5045678.96 1543595.78  69.4%     -    1s
         0     0 1569131.29    0   17 5045678.96 1569131.29  68.9%     -    1s
         0     0 1592015.88    0   17 5045678.96 1592015.88  68.4%     -    1s
         0     0 1611183.82    0   15 5045678.96 1611183.82  68.1%     -    1s
         0     0 1632969.89    0   13 5045678.96 1632969.89  67.6%     -    1s
         0     0 1652393.83    0   15 5045678.96 1652393.83  67.3%     -    1s
         0     0 1652393.83    0   15 5045678.96 1652393.83  67.3%     -    1s
*        0     0               0    4764451.7000 4764451.70  0.00%     -    2s

Explored 1 nodes (4350 simplex iterations) in 2.30 seconds
Thread count was 8 (of 8 available processors)

Solution count 3: 4.76445e+06 5.04568e+06 1.74436e+07

Optimal solution found (tolerance 1.00e-04)
Best objective 4.764451700000e+06, best bound 4.764451700000e+06, gap 0.0000%
```

Model

The Shark Tank model required selection of three manufacturing sites that would minimize the total distance traveled. First, the distance data was read into MySQL workbench and restructured from wide to long format. Then, the data was read into Jupyter, missing values were dropped, and the demand was defined. The decision variables were created such that the first list in the array were binary variables corresponding to whether or not a certain city was a manufacturing center. The rest of the decision variables corresponded to whether or not a certain manufacturing center/demand center combination were to be used in the optimal solution. The constraints were added so that there would be exactly three manufacturing sites represented and each demand center would be assigned a manufacturing site exactly once. The objective function was represented by the sum of miles from the manufacturing center to the destination multiplied by demand for each city center. The minimization of this model was used to answer the questions below.

Question 1

In [118]:
```
print("The cities designated as manufacturing sites are:")
for var in m.getVars():
    if var.x==1:
        if var.varName in cities:
            print('\t',var.varName.replace("..",","))
```

```
The cities designated as manufacturing sites are:
         Dallas,TX
         Los.Angeles,CA
         Silver.Spring,MD
```

Question 2

In [120]:
```
print("The manufacturing site designated for each city is:","\n \n \t Manufacturing Site: Destination")
for var in m.getVars():
    if var.x==1:
        if var.varName not in cities:
            print('\t \t',var.varName.replace("..",",").replace(" ",": "))
```

```
The manufacturing site designated for each city is:

         Manufacturing Site: Destination
                 Dallas,TX: Dallas,TX
                 Dallas,TX: Denver,CO
                 Los.Angeles,CA: Los.Angeles,CA
                 Los.Angeles,CA: Phoenix,AZ
                 Los.Angeles,CA: San.Francisco,CA
                 Los.Angeles,CA: Seattle,WA
                 Silver.Spring,MD: Boston,MA
                 Silver.Spring,MD: Chicago,IL
                 Silver.Spring,MD: Miami,FL
                 Silver.Spring,MD: New.York.City,NY
                 Silver.Spring,MD: Pittsburgh,PA
                 Silver.Spring,MD: Richmond,VA
```

Question 3

In [121]:
```
print("The total number of miles traveled to satisfy demand is %s miles."%("{0:,.2f}".format(m.objVal)))
```

```
The total number of miles traveled to satisfy demand is 4,764,451.70 miles.
```

Question 4

We could use the Greedy Set Cover approximation to solve this problem. While this would certainly return a minimal total distance traveled, it is not guaranteed to produce the optimal result. Therefore, it may not result in the same recommendation that was produced above.