Optimization Final
Megan Ling

Hey Prof.Koehl,

Thank you for the excellent opportunity for me to practice my Gurobi, SQL, R, and Python skills. Didn't expect that, but it turns out it's not bad.

Without further ado, Let's get started.

PROBLEM 1.

Problem 1 is a maximum flow problem

Objective function:

Maximize FLOW = f(1,2)+f(1,3)+f(2,5) …. +f(7,8)+f(7,6)

Constraints:

1. $F(c,v) \leq c(c,v)$
2. Flow in = flow out → f(c,v) = f(v,c)
3. $\sum f(c,u)=0$

So I put f(1,2) to f(7,8) into Excel Solver and add these constraints using simplex LP :

1. Integer
2. Non-zero
3. 3 Constraints as listed

The maximum flow that can be sent from tank 1 to tank 8 per hour is 28.

PROBLEM 2

Problem 2 is basically the same as Problem 1 except using R and multiply a constant k to those flows leading out of node 1 and leading into the node 8.

Same steps from Problem 1, easy to solve using *maxFlowFordFulkerson, graph_from_data_frame,* and *graph.maxflow* functions

#The limit max flow for the graph is 62 when k=3
#Increasing the capacity of all arcs leading out of tank 1 and all arcs leading into the tank 8
#will allow it to double the maximum flow 62.

PROBLEM 4

Not much to say about this one.

PROBLEM 5

Minimize the total cost
Objective function: 0.75 * mileage * num of trailers + 200* num of trailers
Variables : go_or_not & num_of_trailers
Constraints:

- go_or_not -> binary
- num_of_trailers -> integer
- num_of_trailers = required
- each row of go_or_not = 1
- total num_of_trailers from each dc <= 12000

Business Insights:
To minimize the total cost, the company should send
208 trailers from DC 0 to Store 0
282 trailers from DC 1 to Store 0
54 trailers from DC 3 to Store 1
66 trailers from DC 3 to Store 2
The minimized cost is 190282.78

PROBLEM 6

I feel like I need to create a loop. But I decided to be straight forward about this one. Basically, I just expressed what's happening in Excel.

```
Found heuristic solution: objective 406062.31459
Presolve removed 12 rows and 32 columns
Presolve time: 0.01s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.03 seconds
Thread count was 1 (of 8 available processors)

Solution count 2: 190283 406062
```

So minimized result is $190283.

PROBLEM 7. – I AM SO FRUSTRATED; IT TOOKS ME 3 DAYS TO DEBUG! 3 DAYS!

1. Planned to use my old-fashion way in Excel: Creating a table, specifying the columns and rows, and adding constraints, simple and neat. So, I created a Matrix that contains dcid, storied, and mileage info. Failed to call the exactly Keys in the table through indexes, then I believe a Dictionary would help.
2. Want to create a dictionary directly from data frame

Use the shortcut! pandas.DataFrame.to_dict function in Pandas. - **Failed**

```
dic = dict()

for i in range(datarange):
    dic[i,2] = pd.DataFrame({'mileage':(df_mileage[i][2])},
    index=[(df_mileage[i][0],df_mileage[i][1])])

    #for j in len(range(df_mileage[1])):          'dcid': range(df_mileage[i]),
    'storeID': range(df_mileage[j])})

df_mileage.to_dict('dict')
```

3. Then I learned my lesson that never takes a "shortcut" unless you know it clearly. "In Chinese, 不作不就不会死; Never trouble trouble till trouble troubles you."
4. I used the traditional way to create a Dictionary– call keys are the tuples of *dc_id* and *store_id*

```
#Create Dictionary for mileage
trailers_dic = dict()
trailers_dic = {(dcid[i], storeid[i]):mileage_r[i] for i in range(len(mileage_r))}
print(trailers_dic)
```

```
In [148]: print(trailers_dic)
{(0, 0): 102.6070194, (0, 1): 981.5160308, (0, 2): 889.4214567, (0, 3):
720.9953667, (0, 4): 245.7530028, (0, 5): 942.612595, (0, 6):
649.6467958, (0, 7): 503.8589933, (0, 8): 271.5480614, (0, 9):
772.1073281, (0, 10): 291.1008439, (0, 11): 267.3628108, (0, 12):
138.2611935, (0, 13): 476.4816748, (0, 14): 281.6409223, (0, 15):
356.8503845, (0, 16): 613.2955227, (0, 17): 567.5278829, (0, 18):
735.8771931, (0, 19): 107.8153906, (0, 20): 65.18107451, (0, 21):
613.5449303, (0, 22): 612.800752, (0, 23): 1004.192993, (0, 24):
831.865412, (0, 25): 864.6212935, (0, 26): 849.0962312, (0, 27):
882.0693212, (0, 28): 477.2156162, (0, 29): 269.8409847, (0, 30):
396.8491551, (0, 31): 586.6489185, (0, 32): 190.0336072, (0, 33):
942.9387805, (0, 34): 844.2412754, (0, 35): 67.97545475, (0, 36):
218.5290384, (0, 37): 254.1832136, (0, 38): 897.7959159, (0, 39):
262.4859018, (0, 40): 399.7021134, (0, 41): 239.6591063, (0, 42):
70.59607733, (0, 43): 738.1607874, (0, 44): 237.3627665, (0, 45):
```

Beautiful!

5. Then I created variable dictionaries for *trailers* and *g (binary)* and encountered with problems again. The inner logic of the dictionary confused me. So, I decided to use variable lists instead of variable dictionaries. Keep the dictionary I got, I wrote my result to MySQL.
6. To add constrains, I called the indexes that accord with the data frames. (It basically creating lists for each columns/rows in the table with the same logic that sum each column and sum each rows.

```
for i in range(len(dcid)):
    constraints1 = []
    for j in range(len(trailers)):
        if trailers.index(trailers[j]) == i:
            constraints1.append(trailers[i])
    m.addConstr(quicksum(constraints1[j]*df_store[0][i] for j in range(len(constraints1))), GRB.LESS_EQUAL, 12000)

# 1 store only get 1 dc
for i in range(len(storeid)):
    constraints2 = []
    for j in range(len(trailers)):
        if trailers.index(trailers[j]) == i:
            constraints2.append(trailers[i])
    m.addConstr(quicksum(constraints2[j] for j in range(len(constraints2))), GRB.EQUAL, 1)

# trailers satisfy requirement
for i in range(len(storeid)):
    constraints3 = []
    for j in range(len(trailers)):
        if trailers.index(trailers[j]) == i:
            constraints3.append(trailers[i])
    m.addConstr(quicksum(constraints3[j] for j in range(len(constraints3))), GRB.EQUAL, store_requirements[i])
```

7. THESE CONSTRAINTS DIDN'T WORK WITH A KEY ERROR 1100 (spend 2 days. Tried to make it work!)
    i. Draw a chart that matching the steps I'm doing in the scripts. Make sure my logic is right.
    ii. Reviewed all my works and python scripts that I wrote before.
    iii. Having faith that I'm smarter than most people.
    iv. Defeated
8. I was so frustrated. Can't ask anyone for help. Wanted to cry. **T^T** Out of devastation, I record down my failure into this PDF. Then the library therapy dog showed up. I pet the dog and back to work.

9. After, I stepped back and asked myself, what's the simplest way to solve this problem. I believe the idea of adding columns and rows is the right path. Why not using the same intervals to start a new column? Since each column contains the same numbers of stores info.
10. **Five hours later…** I tried to create a list for two variables. But at the final step, my variables are not sequenced. So I back to my 'dictionary fantasy.'
11. Four hours later… I tried to create a list for two variables. But at the final step, my variables are not sequenced. So I back to my 'dictionary fantasy.'

```
g = {}
trailers = {}
for i in range(numdc):
    for j in range(numstores):
        g[(i,j)]=(m.addVar(vtype = GRB.BINARY,name= "%s%s" % (i,j)))
        trailers[(i,j)]=(m.addVar(vtype = GRB.INTEGER, name= "trailer%s%s" % (i,j)))

m.update()
```

Set Objective :

```
#Objective
m.setObjective(0.75*quicksum(g[(i,j)]*mile[(i,j)] for i in range(numdc) for j in range(numstores))+sum(g[(i,j)] for i in range(numdc) for j
```

Variables dictionary:

```
g = {}
trailers = {}
for i in range(numdc):
    for j in range(numstores):
        g[(i,j)] = m.addVar(vtype = GRB.BINARY,name= "pair%s%s" % (i,j))
```

Constraints:

```
# constrains
for i in range(numstores):
    shipments = []
    for key in g:
        if key [-1] == storeid[i]:
            shipments.append(g[key])
    m.addConstr(sum(shipments), GRB.EQUAL, 1)
m.update()

for i in range(numdc):
    y = []
    for key in g:
        if key[0] == dcid[i]:
            y.append(g[key])
    m.addConstr(quicksum(y[i]*requirement[j] for j in range(numstores)), GRB.LESS_EQUAL, 12000)
m.update()
```

12. So, I created a mileage dictionary again. (Prof.Koehl, it's 11:22 I'm running out time… I will make it short and simple); Make my decision variables as a dictionary so that I would easily call the Keys out As for constraints, to better calculate the result of each columns, for each key that match the storeid & dcid, sum the matched data and get results in sequence. For this step, I created calculate data in each columns and appended the result into the list. The list will be wiped out after calculating each column. So the constraints are like what we have in the Excel spreadsheet.
13. Populate the results into the results table

Business Insights

To minimize the cost, the company should follow the from DCid to Storeid structure in Reports table.

PROBLEM 8

DATA EXPORT – nothing much to say

PROBLEM 9

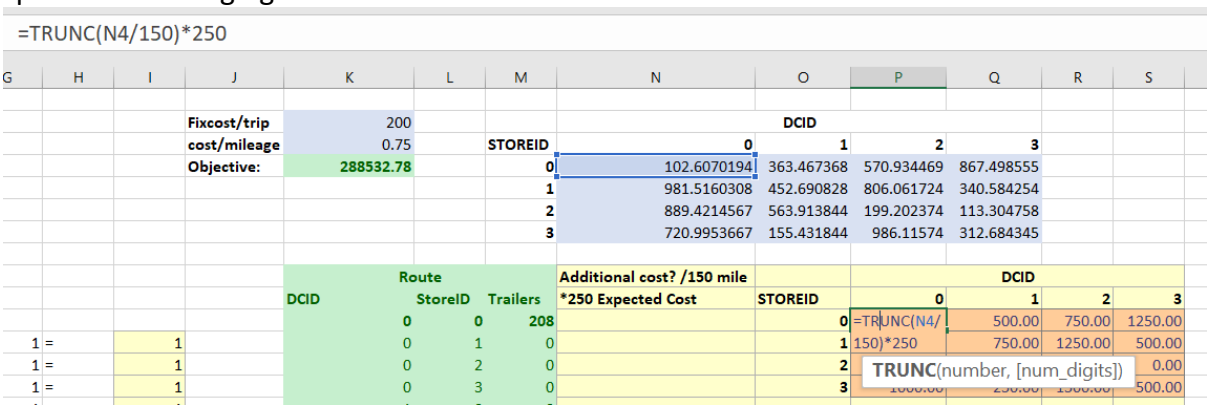
Objective Function:
0.75 * mileage * num of trailers + 200* num of trailers + **Extra Mileage cost**

Variables : go_or_not
Constraints:

- go_or_not -> binary
- num_of_trailers -> integer
- num_of_trailers = required
- each row of go_or_not = 1
- total num_of_trailers from each dc <= 12000

Extra Mileage cost:

1. Calculate every milleage/150 from each dc to each store
2. Use **Trunc()** function in Excel to drop the decimal part to get **Days of stay during the trips**; because when the drivers arrived to destination, there is no need to spend hotel&lodging fee.



3. Calculate for the cost of trips for each pair.
   CostChart -> **Days of stay during the trips * 250**;
4. **go_or_not * num of trailers * CostChart → Extra Mileage cost.**

Happens or not?		DCID			
	STOREID	0	1	2	3
	0	0.00	0.00	0.00	0.00
	1	0.00	0.00	0.00	500.00
	2	0.00	0.00	0.00	0.00
	3	0.00	250.00	0.00	0.00
		DCID			
Total Cost Under senario	STOREID	0	1	2	3
	0	0.00	0.00	0.00	0.00
97500.00	1	0.00	0.00	0.00	27000.00
	2	0.00	0.00	0.00	0.00
	3	0.00	70500.00	0.00	0.00