Session 21: Optimization Modelling III (Reusable Software)

Example (Abstract formulation from last session's example)

Data:

- I: the set of books.
- J: the set of genres.
- a_{ij}: a binary variable denoting whether book i is of genre j. (These corresponds to the checkmarks in the original question.)
- q_j: how many books do we need of genre j.

Decision Variables: For each book $i \in I$, let x_i be a binary decision variable denoting whether to carry the book.

Objective and constraints:

$$\begin{aligned} \text{Minimize:} \quad & \sum_{i \in I} x_i \\ \text{subject to:} & \\ \text{(Enough books in genre)} \quad & \sum_{i \in I} a_{ij} x_i \geq q_j \quad \text{for each genre } j \in J. \end{aligned}$$

Step 4a. Plan how to store all required data in an input spreadsheet.

See `21-data1.xlsx` and `21-data1b.xlsx` that were emailed to everyone.

Digression: Using the Command line to run Python scripts

Open Anaconda Prompt (in Windows) or a Terminal (in Mac). Use the following commands to navigate to the folder containing the unzipped files `21-data1.xlsx`, `21-data2.xlsx`, `books.py` (emailed to everyone).

Obtain current directory (Windows):

```
cd
```

Obtain current directory (Mac):

```
pwd
```

Changing directory:

```
cd directoryName
```

Changing drive (Windows):

```
D:\
```

Moving up one level:

```
cd ..
```

Listing content of current directory (Windows):

```
dir
```

Listing content of current directory (Mac):

```
ls
```

Execute the following commands to run the `books.py` script:

```
python books.py
python books.py 21-data1.xlsx output1.xlsx
python books.py 21-data1.xlsx output1b.xlsx
```

Examine the output files created: output1.xlsx and output1b.xlsx.

Step 4b. Implement the abstract formulation using Python and Gurobi.

Loading data

[4]:
```
import pandas as pd
genres=pd.read_excel('21-data1.xlsx',sheet_name='genres',index_col=0)\
                                                    .fillna(0).astype(int)
genres
```

book	Literary	Sci-Fi	Romance	Thriller
1	1	0	0	0
2	0	1	0	1
3	0	0	1	1
4	1	0	1	0
5	1	0	0	0
6	0	0	1	0
7	0	1	0	0
8	0	0	0	1
9	1	1	0	0
10	0	0	1	0

[5]:
```
requirements=pd.read_excel('21-data1.xlsx',sheet_name='requirements',index_col=0)
requirements
```

genre	required
Literary	2
Sci-Fi	2
Romance	2
Thriller	2

Gurobi Optimization

[30]:
```
from gurobipy import Model, GRB
mod=Model()
I=genres.index
J=genres.columns

x=mod.addVars(I,vtype=GRB.BINARY)
mod.setObjective(sum(x[i] for i in I))
for j in J:
    mod.addConstr(sum(genres.loc[i,j]*x[i] for i in I)>=requirements.loc[j])
mod.setParam('outputflag',False)
mod.optimize()

carry=[]
for i in I:
    if x[i].x:
        carry.append(i)
carry
```

```
[2, 3, 4, 9]
```

Step 4c. Make the code runnable as a function

[29]:
```
import pandas as pd
from gurobipy import Model, GRB

def optimize(inputFile,outputFile):
    genres=pd.read_excel(inputFile,sheet_name='genres',index_col=0).fillna(0)
    requirements=pd.read_excel(inputFile,sheet_name='requirements',index_col=0)

    mod=Model()
    I=genres.index
    J=genres.columns

    x=mod.addVars(I,vtype=GRB.BINARY)
    mod.setObjective(sum(x[i] for i in I))
    for j in J:
        mod.addConstr(sum(genres.loc[i,j]*x[i] for i in I)>=requirements.loc[j])
    mod.setParam('outputflag',False)
    mod.optimize()

    writer=pd.ExcelWriter(outputFile)
    carry=[]
    for i in I:
        if x[i].x:
            carry.append(i)
    pd.DataFrame(carry,columns=['books'])\
        .to_excel(writer,sheet_name='optimal_decision')
    pd.DataFrame([mod.objVal],columns=['books_needed'])\
        .to_excel(writer,sheet_name='objective',index=False)
    writer.save()

optimize('21-data1.xlsx','21-data1-output.xlsx')
optimize('21-data1b.xlsx','21-data1b-output.xlsx')
```

Step 4d. Make the code runnable as a standalone module

The `books.py` file that was emailed to the class contains the above code as well as the following addendum at the end. This allows the code to be run using the command line (in Anaconda Prompt in Windows or in a Terminal in Mac) using the command

```
python books.py inputFile outputFile
```

```
[ ]: if __name__=='__main__':
         import sys, os
         if len(sys.argv)!=3:
             print('Correct syntax: python books.py inputFile outputFile')
         else:
             inputFile=sys.argv[1]
             outputFile=sys.argv[2]
             if os.path.exists(inputFile):
                 optimize(inputFile,outputFile)
                 print(f'Successfully optimized. Results in "{outputFile}"')
             else:
                 print(f'File "{inputFile}" not found!')
```

For example, one can process the first input file by running in command line.

```
python books.py 21-data1.xlsx 21-data1-output.xlsx
```

Exercise (Transportation Planning)

Abstract formulation from lass session's Q2

Data:

- I: the set of office branches.
- J: the set of conventions.
- s_i: the number of available representatives at office branch $i \in I$.
- d_j: the number of representatives needed at convention $j \in J$.
- c_{ij}: the roundtrip airfare from branch i to convention j.

Decision variables: Let X_{ij} denote how many representatives to send from branch $i \in I$ to convention $j \in J$. (Integer)

Objective and constraints:

$$\begin{aligned}
\text{Minimize} \quad & \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} \\
\text{s.t.} \quad & \\
\text{(Supply)} \quad & \sum_{j\in J} x_{ij} \le s_i \quad \text{for each branch } i \in I \\
\text{(Demand)} \quad & \sum_{i\in I} x_{ij} \ge d_j \quad \text{for each convention } j \in J \\
\text{(Non-negativity)} \quad & x_{ij} \ge 0 \quad \text{for each } i \in I, j \in J
\end{aligned}$$

Step 4a. Plan how to store all required data in an input spreadsheet.

See `21-data2.xlsx` and `21-data2b.xlsx`.

Step 4b. Implement the abstract formulation using Python and Gurobi.

Load the data and implement the optimization in Jupyter notebook. Don't worry about outputing format, writing a function or copying into a Python module for now.

[33]:

```
             Los Angeles  St. Louis  Detroit
Little Rock          250        150      200
Urbana               300        200      150
```

[34]:

```
             available
branch
Little Rock          6
Urbana               6
```

[35]:

```
             needed
convention
Los Angeles       2
St. Louis         5
Detroit           4
```

[39]:

```
Minimal cost: 1900.0
        x[Little Rock , Los Angeles]=1.0
        x[Little Rock , St. Louis]=5.0
        x[Little Rock , Detroit]=0.0
        x[Urbana , Los Angeles]=1.0
        x[Urbana , St. Louis]=0.0
        x[Urbana , Detroit]=4.0
```

Step 4c. Make the code runnable as a function

After verifying that the code in step 4b works, store the output in pandas DataFrames (so that you can export into excel easily). Then write the optimal objective and the optimal plan for sending representatives into an excel workbook (in different sheets). Finally, indent everything and put it in a function.

Step 4d. Make the code runnable as a standalone module

Copy the working code from step 4c into a Python module `transportation.py` using Spyder, and add the corresponding code as in the example to make it runnable from the command line. Verify by running

```
python transportation.py 21-data2.xlsx 21-data2-output2.xlsx
python transportation.py 21-data2b.xlsx 21-data2b-output2.xlsx
```