# Homework 7 Solutions

This homework is to complete the code for labs 5 and 6. Below is one way of formulating and solving each problem. (There are multiple correct formulations with different conventions of indices and letterings. Make sure you define the indices for your formulation.)

## Solutions to Lab 5

### LP Formulation

**Indices:**

- $i$: fulfilment center. (The set of FCs is $I$.)
- $j$: demand region. (The set of regions is $J$.)
- $k$: item. (The set of items is $K$.)

**Decision variable:** Let $x_{ijk}$ denote the number of units of item $k$ to ship from FC $i$ to region $j$.

**Data:**

- $q_i$ is the capacity of FC $i$ (in cubit feet).
- $d_{ij}$ is the distance from FC $i$ to region $j$.
- $w_k$ is the shipping weight of item $k$ (in lbs).
- $s_k$ is the storage size of item $k$ (in cubit feet).
- $d_{jk}$ is the demand for item $k$ in region $j$.

**Linear Program:** In the following, whenever we sum over $i$, it is assumed that we sum for all $i \in I$. Similarly for indices $j$ and $k$. This is a shorthand that is convenient when writing on paper.

$$
\begin{array}{lrl}
\text{Maximize:} & 1.38 \sum_{i,j,k} w_k d_{ij} x_{ijk} & \\
\text{subject to:} & & \\
\text{(FC capacity)} & \sum_{j,k} s_k x_{ijk} \leq q_i & \text{for all fulfilment center } i. \\
\text{(Satisfying all demand)} & \sum_{i} x_{ijk} \geq d_{jk} & \text{for all regions } j \text{ and items } k. \\
\text{(Non-negativity)} & x_{ijk} \geq 0 & \text{for all } i, j, \text{ and } k.
\end{array}
$$

The LP without the short hand would be:

$$
\begin{array}{lrl}
\text{Maximize:} & 1.38 \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} w_k d_{ij} x_{ijk} & \\
\text{subject to:} & & \\
\text{(FC capacity)} & \sum_{j \in J} \sum_{k \in K} s_k x_{ijk} \leq q_i & \text{for all fulfilment center } i \in I. \\
\text{(Satisfying all demand)} & \sum_{i \in I} x_{ijk} \geq d_{jk} & \text{for all regions } j \in J \text{ and items } k \in K. \\
\text{(Non-negativity)} & x_{ijk} \geq 0 & \text{for all } i \in I, j \in J, \text{ and } k \in K.
\end{array}
$$

## Python Code

[ ]:
```python
inputFile='data.xlsx'
outputFile='output_for_'+inputFile

import pandas as pd
# Note: for older versions of pandas, you should use "sheetname" without the underscore
data=pd.read_excel(inputFile,sheet_name=None,index_col=0)
centers=data['Fulfilment Centers']
regions=data['Regions']
distances=data['Distances']
items=data['Items']
demand=data['Demand']

import gurobipy as grb
mod=grb.Model()
x={}

# Obtaining the index sets from the Dataframes directly
I=centers.index
J=regions.index
K=items.index

# Defining decision variables
for i in I:
    for j in J:
        for k in K:
            x[i,j,k]=mod.addVar(lb=0)

# Defining objective function
mod.setObjective(1.38*grb.quicksum(distances.loc[j,i]*x[i,j,k]*items.loc[k,'shipping_weight'] for i,j,k in x),
                 sense=grb.GRB.MINIMIZE)

# Defining capacity constraints
capConstr={}
for i in I:
    capConstr[i]=mod.addConstr(grb.quicksum(items.loc[k,'storage_size']*x[i,j,k] for j in J for k in K)<=
                               centers.loc[i,'capacity'])
# Defining demand constraints
for j in J:
    for k in K:
        mod.addConstr(grb.quicksum(x[i,j,k] for i in I)>=demand.loc[k,j])

mod.optimize()

# Printing objective value
print('Objective value',mod.ObjVal)

# Opening excel file
writer = pd.ExcelWriter(outputFile, engine='xlsxwriter')

# Writing summary sheet
summary=pd.DataFrame([mod.ObjVal],columns=['Objective Value'])
summary.to_excel(writer,sheet_name='Summary',index=False)

# Writing solution sheet
data=[]
for i,j,k in x:
    value=x[i,j,k].x
    if value>0:
        data.append([i,j,k,value])
shipments=pd.DataFrame(data,columns=['FC_name','region_ID','item_ID','Shipment'])
shipments.to_excel(writer,sheet_name='Solution',index=False)

# Writing shadow price sheet
data=[]
for i in I:
    data.append([i,capConstr[i].PI])
capPrice=pd.DataFrame(data,columns=['FC_name','Shadow Price'])
capPrice.to_excel(writer,sheet_name='Capacity Constraints', index=False)

writer.save()
```

# Solution to Lab 6

## IP Formulation

**Indices:**

- person $i$
- shift $j$

**Decision variables:**

- $x_{ij}$ is a binary variable for whether we assign person $i$ to shift $j$.
- $U_{all}$ and $L_{all}$ are upper and lower bounds to the total number of shifts by a nurse.
- $U_{night}$ and $L_{night}$ are upper and lower bounds to the total number of **night** shifts by a nurse.

**Data:**

- Set of people $I$.
- Preference $p_{ij} \in \{0, 1, 2\}$ of person $i$ for shift $j$. 0 denotes infeasible, 1 denotes feasible but undesirable, 2 denotes feasible and desirable.
- $f_{ij} \in \{0, 1\}$ denote whether shift $j$ is feasible for person $i$. $f_{ij} = 1$ if $p_{ij} \geq 1$.
- $n$ is the number of weeks.
- Set of shift $J = \{0, 1, 2, \cdots, 21n - 1\}$.
- Set of night shifts $J_{night} = \{2, 5, \cdots, 21n - 1\}$.
- $q_j$ is the number of nurses needed for shift $j$.

**Integer Program:**

$$
\begin{array}{lll}
\text{Maximize} & -100(U_{all} - L_{all}) - 100(U_{night} - L_{night}) + \displaystyle\sum_{i \in I, j \in J} p_{ij} x_{ij} & \\
\text{subject to:} & & \\
\text{(Shift requirement)} & \displaystyle\sum_{i \in I} x_{ij} = q_j & \text{for all } j \in J. \\
\text{(Feasibility)} & x_{ij} \leq f_{ij} & \text{for all } i \in I \text{ and } j \in J. \\
\text{(At most 6 shifts per week)} & \displaystyle\sum_{j=21w}^{21(w+1)-1} x_{ij} \leq 6 & \text{for all } i \in I, w \in \{0, 1, \cdots, n\} \\
\text{(No consecutive shifts)} & x_{ij} + x_{i(j+1)} \leq 1 & \text{for all } i \in I, j \in \{0, 1, \cdots, 21n - 2\} \\
\text{(Rest before night shift)} & x_{ij} + x_{i(j-2)} \leq 1 & \text{for all } i \in I, j \in J_{night} \\
\text{(Rest after night shift)} & x_{ij} + x_{i(j+2)} \leq 1 & \text{for all } i \in I, j \in \{2, 5, \cdots, 21n - 4\}. \\
\text{(Bounds on total shifts)} & L_{all} \leq \displaystyle\sum_{j \in J} x_{ij} \leq U_{all} & \text{for all } i \in I \\
\text{(Bounds on night shifts)} & L_{night} \leq \displaystyle\sum_{j \in J_{night}} x_{ij} \leq U_{night} & \text{for all } i \in I \\
\text{(Binary variables)} & x_{ij} \in \{0, 1\} & \text{for all } i \in I, j \in J.
\end{array}
$$

## Python Code

```
[ ]: inputData='data.xlsx'
     outputData='output for '+inputData
```

```python
import pandas as pd
prefs=pd.read_excel(inputData,sheet_name='Preferences',index_col=0)
shifts=pd.read_excel(inputData,sheet_name='Shifts',index_col=0)
feasible=(prefs>0)*1
I=prefs.index
J=shifts.index
numweeks=int(len(J)/21)
JNight=range(2,len(J),3)
import gurobipy as grb
mod=grb.Model()

# Define decision variables
x={}
for i in I:
    for j in J:
        x[i,j]=mod.addVar(vtype=grb.GRB.BINARY)
UAll=mod.addVar(lb=0)
LAll=mod.addVar(lb=0)
UNight=mod.addVar(lb=0)
LNight=mod.addVar(lb=0)

# Objective function
mod.setObjective(sum(x[i,j]*prefs.loc[i,j] for i in I for j in J)-100*(UAll-LAll)-100*(UNight-LNight),
                 sense=grb.GRB.MAXIMIZE)

# Shift requirement constraint
for j in J:
    mod.addConstr(sum(x[i,j] for i in I)==shifts.persons.loc[j])

# Scheduling constraints for each person
for i in I:
    # Feasibility
    for j in J:
        mod.addConstr(x[i,j]<=feasible.loc[i,j])

    # At most 6 shifts per week
    for week in range(numweeks):
        mod.addConstr(sum(x[i,j] for j in range(week*21,(week+1)*21)) <= 6)
    # no consecutive shifts
    for j in J[:-1]:
        mod.addConstr(x[i,j]+x[i,j+1]<=1)
    # Rest before night shift
    for j in JNight:
        mod.addConstr(x[i,j-2]+x[i,j]<=1)
    # Rest after night shift
    for j in JNight[:-1]:
        mod.addConstr(x[i,j]+x[i,j+2]<=1)

    # Upper and lower bounds for all shifts
    mod.addConstr(sum(x[i,j] for j in J)<=UAll)
    mod.addConstr(sum(x[i,j] for j in J)>=LAll)

    # Upper and lower bounds for night shifts
    mod.addConstr(sum(x[i,j] for j in JNight)<=UNight)
    mod.addConstr(sum(x[i,j] for j in JNight)>=LNight)


mod.optimize()
print(mod.ObjVal)
print('Pref score=',sum(x[i,j].x*prefs.loc[i,j] for i in I for j in J))
print('Shift inequality=',UAll.x-LAll.x)
print('Night inequality=',UNight.x-LNight.x)

result=pd.DataFrame('',index=prefs.index,columns=prefs.columns)
for i,j in x:
    if x[i,j].x>0:
        result.loc[i,j]=1
# The following line relabels the column indicies into day, time and shifts.
# (See the sample output file from the lab.) This was not required but it makes reading the output easier.
result.columns=pd.MultiIndex.from_arrays([shifts.day,shifts.time,shifts.index])
result.to_excel(outputData)
```