Lab 3: Shift Scheduling

Learning Objectives:

- Modeling a scheduling problem using mixed-integer programming (MIP). (Model)
- Expressing logical constraints using binary variables and index notation. (Analyze)
- Linearizing non-linear constraints. (Analyze)
- Implementing a large scale integer program using Gurobi. (Code)

1. Problem

Trevon is in charge of scheduling shifts for all nurses at the emergency department at Trojan community hospital. In order to help the hospital retain quality staff, Trevon would like to create schedules that treat all nurses fairly and satisfy their preferences as much as possible. In the past, he spent many painstaking hours scheduling by hand, but he feels that the schedules can still be improved. Recently, Trevon learned about mixed-integer programming (MIP) in a course he is taking at USC. He would like to apply his knowledge to create better schedules.

The shifts are scheduled n weeks at a time (beginning on a Sunday and ending on a Saturday). (n is a parameter that can be inferred from the input data.)

Shift requirements: There are three nurse shifts in a day: morning, evening, and nights. Each shift should have exactly a certain number of nurses, which is specified in the input file in the sheet called "Requirements".

Scheduling constraints: Based on hospital and union regulations, no nurse can be assigned more than 6 shifts in a calendar week (Sunday to Saturday). Moreover, no nurse may be scheduled to consecutive shifts. For example, it is not allowed to assign a certain nurse both the morning and the evening shift of the same day. In addition, if a nurse works in a night shift, he/she must take the two prior shifts off as well as the two next shifts off. For example, if a nurse works Tuesday night, then he or she cannot work that Tuesday's morning or evening, nor can she work on Wednesday morning or evening. In other words, you must give her two shifts worth of time to sleep before and after the night shift.

Finally, each nurse can blackout a certain number of shifts, which means that he/she cannot make those shifts under any circumstances (due to vacations and personal conflicts). To blackout a shift, the nurse would specify "0" for the corresponding shift in the nurse's preferences, which is given in the sheet called "Preferences" and is described below.

Nurse Preferences: For each shift, a nurse may indicate a preference score of 0, 1, or 2. A score of 0 indicates that the shift is blacked-out, and he/she cannot be assigned to it. A score of 1 indicates that the time is not preferred, but the nurse is willing to work if needed. A score of 2 indicates a preferred time slot.

Objective: Trevon would like to maximize the following objective, subject to satisfying all shift requirements and scheduling constraints,

$$(\text{Sum of preference scores}) - 100 \times (\text{shift inequality}) - 150 \times \text{night inequality}.$$

- **Sum of preference scores**: for each shift, add up the preference scores of all nurses assigned to the shift. Then sum over all shifts in the 9 week scheduling period.
- **Shift inequality**: the maximum number of shifts worked by any nurse in the entire period represented by the input data file, minus the minimum number of shifts worked by any nurse.
- **Night inequality**: the maximum number of night shifts worked by any nurse in the entire period, minus the minimum number of night shifts.

Help Trevon formulate a mixed-integer program (MIP) using proper mathematical formulation and find an optimal schedule for his input data.

2. Data

The following files are associated with this lab and can be downloaded from Blackboard.

- **data.xlsx:** the main input file for this lab. There are two sheets. The sheet "Preferences" contains the preference score submissions of all nurses for each shift. The sheet "Requirements" contains the number of nurses required for each shift.
- **small_data.xlsx:** a smaller dataset of the same format as the above.
- **output_for_small_data.xlsx:** a correct optimization output using the input file "small_data.xlsx". (There may be other correct optimal schedules that achieve the same objective but with different assignments.) See Section 3.2 for a description.
- **grade_lab3.py:** the grading script that will be used to assess your schedule (see instructions in Section 4).

2.1 Reading and writing data with MultiIndex

The following code illustrates how to do the proper file input and output for this lab. The code first reads the "Preferences" sheet from "small_data.xlsx", which has multiple rows representing a column header (represented as a MultiIndex object in Python). The code then replace the MultiIndex by a numerical index, which makes it more manageable. Finally, the code creates a partial schedule using the MultiIndex, as well as fake summary statistics and save to "tmp.xlsx". This gives you an idea of how to create the output files so that it has the required format.

```
[1]: import pandas as pd
     prefs=pd.read_excel('small_data.xlsx',header=[0,1,2],sheet_name='Preferences')
     prefs
```

```
day         2019-03-30                2019-03-31                2019-04-01  ...
time           Morning Evening Night    Morning Evening Night    Morning  ...
shift_id             0       1     2          3       4     5          6   ...
name
Alexis               1       1     1          1       2     1          1   ...
Alyssa               2       2     0          1       2     1          2   ...
Anthony              1       1     2          1       1     2          2   ...
Brandon              1       1     2          2       2     1          1   ...
Brianna              0       2     1          1       1     2          0   ...
Caleb                1       2     1          1       2     2          1   ...
Cameron              2       2     0          0       0     0          0   ...
Chloe                2       0     0          0       0     1          2   ...
Christopher          1       1     1          1       1     1          2   ...

[9 rows x 21 columns]
```

```
[2]: names=prefs.index
     shifts=prefs.columns
     shift_id=shifts.get_level_values(2)
     shift_id
```

```
Int64Index([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
            19, 20],
           dtype='int64', name='shift_id')
```

[3]:
```
# Remove the MultiIndex
prefs.columns=shift_id
prefs
```

```
shift_id     0   1   2   3   4   5   6   7   8   9  ...
name                                                ...
Alexis       1   1   1   1   2   1   1   0   2   1 ...
Alyssa       2   2   0   1   2   1   2   2   2   1 ...
Anthony      1   1   2   1   1   2   2   1   0   2 ...
Brandon      1   1   2   2   2   1   1   2   1   0 ...
Brianna      0   2   1   1   1   2   0   0   1   2 ...
Caleb        1   2   1   1   2   2   1   2   2   1 ...
Cameron      2   2   0   0   0   0   0   0   0   0 ...
Chloe        2   0   0   0   0   1   2   1   2   2 ...
Christopher  1   1   1   1   1   1   2   1   2   2 ...

[9 rows x 21 columns]
```

[7]:
```
schedule=pd.DataFrame('',index=names,columns=shift_id)
schedule.loc['Alexis',0]=1
schedule.loc['Alyssa',1]=1
schedule.columns=shifts
schedule
```

```
day         2019-03-30                2019-03-31                2019-04-01  ...
time           Morning Evening Night    Morning Evening Night    Morning  ...
shift_id             0       1     2          3       4     5          6   ...
name                                                                       ...
Alexis                1                                                    ...
Alyssa                        1                                            ...
Anthony                                                                    ...
Brandon                                                                    ...
Brianna                                                                    ...
Caleb                                                                      ...
Cameron                                                                    ...
Chloe                                                                      ...
Christopher                                                                ...

[9 rows x 21 columns]
```

[5]:
```
summary=pd.Series(name='Value')
summary['Objective']=0
summary['Shifts scheduled']=2
summary
```

```
Objective           0
Shifts scheduled    2
Name: Value, dtype: int64
```

[6]:
```
writer=pd.ExcelWriter('tmp.xlsx',datetime_format='m/dd')
schedule.to_excel(writer,sheet_name='Schedule')
summary.to_excel(writer,sheet_name='Summary')
writer.save()
```

3. Deliverables (due at 11am one week after the lab is assigned)

Each individual must submit the following two files on Blackboard (individuals from the same team may submit the same files, but each individual is responsible for his/her own submission and will be graded separately.)

3.1 formulation.ipynb

A Jupyter notebook containing an abstract formulation of the MIP, defining all data variables, decision variables, the objective and all constraints. The formulation must use correct mathematical notation and be typeset using Latex in a Markdown cell. To express sums, the formulation should use the correct summation notation with Σ.

3.2 output_for_data.xlsx

An optimal schedule corresponding to the input file "data.xlsx", which must have the EXACT SAME FORMAT as the provided "output_for_small_data.xlsx". Precisely speaking, there must be two sheets, named "Schedule" and "Summary" respectively.

The first sheet "Schedule" should contain one row for each nurse and one column for each shift. There should be a "1" if the nurse is scheduled for that shift, and the cell should be blank otherwise. The first column should correspond to the morning shift of the first day, the second column the evening shift, and so on. The table must be readable using `pandas.read_excel`.

The second sheet "Summary" should contain two columns. The first column corresponds to labels and should be identical to the "small_data.xlsx" and the second column should contain the corresponding value. Below are the description of the labels in the first column:

- Objective: the final objective value.
- Total preference score: the total preference score of your schedule.
- Shift inequality: the shift inequality corresponding to your schedule.
- Night inequality: the night inequality corresponding to your schedule.

4. Grading Rubric

The lab will be graded out of a total of 4 points, with one point in each of the following 4 categories. The grading is binary, meaning that a perfectly correct solution for a category would obtain 1 point, while any mistake (however minor) would result in 0 for that category.

i) Mathematical Formulation

The abstract formulation in the Jupyter notebook is complete and correct, and uses proper mathematical notation formatted nicely in a Markdown cell using Latex. To receive the point, you must

- define every data variable or decision variable you use;
- use proper summation notation, including denotation of what set each sum is going over and how constraints are repeated.
- express all constraints linearly (no non-linear formulation allowed).
- encode all necessary relationships between the decision variables and any auxiliary variables.
- render the formulation in a readable fashion using Latex.

For examples of how to format using Latex, you should download the .ipynb files from recent session handouts as well as solutions to homeworks.

Note: Categories ii) through iv) will be autograded using the `grade_lab3.py` file attached. You should grade your submission yourself to see what you will get by running the following. The grading script will point out whether you get a category correct and indicate where you may have an error, you should repeatedly run the grading script and update your code until you are satisfied with the grade.

```
python grade_lab3.py output_for_data.xlsx data.xlsx
```

For development purposes, you should get your code working on "small_data.xlsx" before trying the "data.xlsx" (for which the optimization may take up to several minutes each time). The `grade_lab3.py` script is also designed to evaluate your output to the "small_data.xlsx". If you saved your output to "small_output.xlsx", you can simply run the following to check.

```
python grade_lab3.py small_output.xlsx small_data.xlsx
```

ii) Correct Summary Statistics

The summary statistics in your output file (the values in the "Summary" sheet of the excel file you create) are correctly calculated for the schedule you create (given in the "Schedule" sheet of your output file).

iii) Feasible Schedule for Actual Data

The schedule you submit satisfies all requisite constraints.

iv) Optimal Schedule for Actual Data

You have full points for ii) and iii) (feasible schedule with correct summary statistics), and your objective value is optimal. Note that there can be many equally optimal but different schedules, and any of them would receive full points.