Schedule Distribution Linear Optimization

Author: Gihyeon Kwon

Question: MSBA example

As new cohort of USC MSBA class enters, the program office needs to evenly distribute the students into four different cores: A, B, C, D Each core represents the schedule each student will have along with their core-mates.

Each student is given a chance to rank their preferred time schedules. (Example: Student Bob: 1. Core 3, 2. Core 4, 3. Core 1, 4. Core 2) Say there are 240 students entering and 4 cores, our goal is go distribute 240 students into the 4 cores while minimizing the assigned core preference rank of all students.

Constraints:

- ~**No** student should receive a core they have ranked last~ (This will not allow for certain edge cases optimize)
- All cores should have nearly the same number of students

Sample: MSBA class of 2025 rank simulation

In [1]:
```
import pandas as pd
import numpy as np

np.random.seed(42)
cores = range(1,5)
students = range(1,241)

random_ranks = [np.random.permutation(len(cores)) + 1 for s in students]
data = pd.DataFrame(random_ranks, index = students, columns = cores)
data.to_excel('msba-input.xlsx')
```

The input data will have students as the rows and core as the columns. The values will represent the ranking for the student for each core.

In [2]:
```
data.head()
```

Out[2]:

	1	**2**	**3**	**4**
1	2	4	1	3
2	2	4	1	3
3	4	1	2	3
4	2	1	4	3
5	3	2	1	4

Abstract Formulation

Data:

- NS: Number of students
- NC: Number of cores
- S: Set of students $\in \{1, 2, \ldots, NS\}$
- C: Set of cores $\in \{1, 2, \ldots, NC\}$
- P_{sc}: Preference rank of student s and for core c
- k: $\frac{NS}{NC}$

Decision Variables:

- X_{sc}: Whether or not each student s will be placed in core c (Binary)

Objective:

$$\text{Minimize}: \sum_{s \in S, c \in C} P_{sc} X_{sc}$$

Constraints:

$$\begin{array}{lcr} \text{(One core per student)} & \displaystyle\sum_{c \in C} X_{sc} = 1 & \text{for each student } s \in S \\ \text{(Core Distribution)} & \displaystyle\lfloor k \rfloor \leq \sum_{s \in S} X_{sc} \leq \lceil k \rceil & \text{for each core } c \in C \end{array}$$

Gurobi Function

In [9]:
```
def optimize_distribution(inputFile, outputFile):
    import pandas as pd
    from gurobipy import Model, GRB
    import math
```

```python
    data = pd.read_excel(inputFile, index_col = 0)

    NS = data.shape[0]
    NC = data.shape[1]
    S = data.index
    C = data.columns
    p = data
    k = NS / NC

    mod = Model()
    X = mod.addVars(S, C, vtype= GRB.BINARY)
    mod.setObjective(sum(p.loc[s,c] * X[s,c] for s in S for c in C))

    for s in S:
        mod.addConstr(sum(X[s,c] for c in C) == 1)
    for c in C:
        mod.addConstr(sum(X[s,c] for s in S) >= math.floor(k))
        mod.addConstr(sum(X[s,c] for s in S) <= math.ceil(k))
    # for c in C:
    #     for s in S:
    #         if p.loc[s,c] == NC:
    #             mod.addConstr(X[s,c] == 0)

    mod.setParam('OutputFlag',False)
    mod.optimize()

    writer = pd.ExcelWriter(outputFile)

    summary = pd.DataFrame([[NS, NC, mod.objVal, mod.objVal / NS]], columns = (['Number of Students', 'Number of Cores',
                                                            'Total Preference Rank', 'Average Preference Rank']))
    summary.to_excel(writer, sheet_name = 'Summary', index=False)

    out = pd.DataFrame(index=S, columns= C)
    for s in S:
        for c in C:
            if X[s,c].x == 1:
                out.loc[s, c] = 1
            else:
                out.loc[s, c] = 0

    out.to_excel(writer, sheet_name='Results')

    distribution = pd.DataFrame(out.sum()).T
    distribution.to_excel(writer, sheet_name='Core Distribution')

    writer.close()
```

In [10]:

```python
#Test 1

import os
output_file = 'msba-output.xlsx'
if os.path.exists(output_file):
    os.remove(output_file)
optimize_distribution('msba-input.xlsx',output_file)
display(pd.read_excel(output_file,sheet_name='Summary'))
display(pd.read_excel(output_file,sheet_name='Results', index_col=0).head())
display(pd.read_excel(output_file,sheet_name='Core Distribution', index_col=0))
```

	Number of Students	Number of Cores	Total Preference Rank	Average Preference Rank
0	240	4	247	1.029167

	1	2	3	4
1	0	0	1	0
2	0	0	1	0
3	0	1	0	0
4	0	1	0	0
5	0	1	0	0

	1	2	3	4
0	60	60	60	60

In [11]:

```python
#Test 2 input

input2_file = 'test2_input.xlsx'
np.random.seed(42)
cores = range(1,11)
students = range(1,562)

random_ranks = [np.random.permutation(len(cores)) + 1 for s in students]
data = pd.DataFrame(random_ranks, index = students, columns = cores)
data.to_excel(input2_file)
data.head()
```

Out[11]:

	1	2	3	4	5	6	7	8	9	10
1	9	2	6	1	8	3	10	5	4	7
2	1	2	9	6	4	5	8	10	7	3
3	10	3	1	7	9	6	4	8	2	5
4	2	8	7	3	9	1	4	5	6	10
5	2	6	5	9	1	8	7	4	3	10

In [12]:

```
#Test 2 output
import os
output_file = 'test2-output.xlsx'
if os.path.exists(output_file):
    os.remove(output_file)
optimize_distribution(input2_file,output_file)
display(pd.read_excel(output_file,sheet_name='Summary'))
display(pd.read_excel(output_file,sheet_name='Results', index_col=0).head())
display(pd.read_excel(output_file,sheet_name='Core Distribution', index_col=0))
```

	Number of Students	Number of Cores	Total Preference Rank	Average Preference Rank
0	561	10	594	1.058824

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	1	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0
4	0	0	0	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0

	1	2	3	4	5	6	7	8	9	10
0	56	56	56	56	56	56	56	56	57	56

In [13]:

```
#Test 3 Edge input: All students have same preference

input3_file = 'test3_input.xlsx'
np.random.seed(42)
cores = range(1,5)
students = range(1,241)

ranks = [[1,2,3,4] for s in students]
data = pd.DataFrame(ranks, index = students, columns = cores)
data.to_excel(input3_file)
data.head()
```

Out[13]:

	1	2	3	4
1	1	2	3	4
2	1	2	3	4
3	1	2	3	4
4	1	2	3	4
5	1	2	3	4

In [14]:

```
#Test 3 Edge output, Model could not optimize

import os
output_file = 'test3-output.xlsx'
if os.path.exists(output_file):
    os.remove(output_file)
optimize_distribution(input3_file,output_file)
display(pd.read_excel(output_file,sheet_name='Summary'))
display(pd.read_excel(output_file,sheet_name='Results', index_col=0).head())
display(pd.read_excel(output_file,sheet_name='Core Distribution', index_col=0))
```

	Number of Students	Number of Cores	Total Preference Rank	Average Preference Rank
0	240	4	600	2.5

	1	2	3	4
1	0	0	1	0
2	0	0	1	0
3	0	0	1	0
4	1	0	0	0
5	0	0	1	0

	1	2	3	4
0	60	60	60	60