Solutions to Sample Final Exam B

Learning Objective:

- Create Python code to automate a given task.
- Formulate linear optimization models to inform a business decision.

Instructions:

The final exam tests your mastery of skills taught in Weeks 7-12, which culminates in creating linear optimization models to inform a given business decision. There are three questions, worth a total of 30 points. The exam is 100 minutes, and is open-notes but closed-computer. You can bring paper notes or books of any kind, but no computers, tablets, or cell phones are allowed. Do not share your solutions with a student who has not yet completed an exam and do not look at other people's solutions. **Any violation of academic integrity will result in a zero grade for the exam for everyone involved.**

As long as you fulfill all the specifications described in the problem description, it doesn't matter how you model the problem or how efficient is your code. As with the midterm, partial credits will be given for any fragments of correct solutions, or for English descriptions that would lead to a solution.

Q1. Exercise Selection in Course Design (Concrete Formulation; 11 points)

Professor Shi would like to select an optimal set of exercises for his course, so that students learn a lot from doing the exercises and the level of difficulty builds up gradually. He has in mind a fixed set of exercises to select from. For each exercise, he has assigned a difficulty score out of 10, and a learning score out of 10. A higher difficulty score means that the exercise is more difficult, and a higher learning score means that it has higher potential to teach the students useful skills for the real world. As an example, the following table lists a set of 11 exercises to select from, along with their assigned scores.

Exercise #	Difficulty Score	Learning Score
0	0	-
1	1	6
2	1	5
3	3	8
4	4	7
5	5	8
6	6	6
7	6	5
8	8	6
9	9	8
10	10	-

In the above, Exercise 0 is the easiest and corresponds to an introductory exercise that sets the baseline for the course, and Exercise 10 is the hardest and represents the final exam. Both of these must always be included in the course, so their learning scores are not relevant for the optimization.

Write a concrete formulation of a linear optimization model to decide which of the exercises 1 through 9 to select in order to maximize the total learning score of the selected exercises, subject to the following constraints:

1. Among Exercises 1 through 9, at most four can be selected due to time limits.
2. In order to ensure that there is no big jump in difficulty between two consecutive exercises that are

selected, the maximum gap in difficulty score between two consecutive selected exercises is at most 3. (Note that the difficulty score is distinct from the exercise #, so do not confuse the two.)

- For example, if the set of all selected exercises is $\{0, 1, 3, 7, 9, 10\}$, then this would satisfy this constraint, as the gap in difficulty score between the pair of consecutive exercises $(0, 1)$ is 1, between the pair $(1, 3)$ is 2, between the pair $(3, 7)$ is 3, between the pair $(7, 9)$ is 3, and between the pair $(9, 10)$ is 1.
- However, if the set of all selected exercises is $\{0, 1, 5, 7, 9, 10\}$, then this constraint would be violated as the difficulty gap between the pair $(1, 5)$ is 4. Similarly, $\{0, 1, 3, 5, 7, 10\}$ would violate this constraint, as the difficulty gap between the pair $(7, 10)$ is 4.

3. Due to overlapping content, certain pairs of exercises should not be both selected. In this example, Exercises 1 and 8 should not be both selected. Moreover, Exercises 2 and 6 should not be both selected.

Your constraints must allow for every feasible selection of exercises according to the above rules, and must disallow every infeasible selection.

Sample Solution 1

Decision Variables:

- $x_1, x_2, \cdots, x_9$: whether to use each exercise 1 through 9. (Binary)

Objective:

$$\text{Max: } 6x_1 + 5x_2 + 8x_3 + \cdots + 8x_9$$

Constraints:

$$\begin{aligned}
\text{(Pre-req 10)} \quad & 1 \leq x_8 + x_9 \\
\text{(Pre-req 9)} \quad & x_9 \leq x_6 + x_7 + x_8 \\
\text{(Pre-req 8)} \quad & x_8 \leq x_5 + x_6 + x_7 \\
\text{(Pre-req 6, 7)} \quad & x_6, x_7 \leq x_3 + x_4 + x_5 \\
\text{(Pre-req 5)} \quad & x_5 \leq x_3 + x_4 \\
\text{(Pre-req 4)} \quad & x_4 \leq x_1 + x_2 + x_3 \\
\text{(Conflicts)} \quad & x_1 + x_8 \leq 1 \\
& x_2 + x_6 \leq 1 \\
\text{(Number)} \quad & x_1 + x_2 + \cdots + x_9 \leq 4
\end{aligned}$$

Explanation

The above solution handles the key constraint of no big jumps in difficulty level by enforcing a series of pre-requisite constraints. For example,

$$\text{(Pre-req 10)} \qquad 1 \leq x_8 + x_9$$

means that in order for Exercise 10 to be included, at least one of Exercise 8 or 9 is needed, otherwise there would be a big jump to Exercise 10. Similarly,

$$\text{(Pre-req 9)} \qquad x_9 \leq x_6 + x_7 + x_8$$

enforces that if Exercise 9 is included, then at least one of Exercises 6, 7 or 8 is included, otherwise there would be a big jump to Exercise 9.

Sample Solution 2

Constraints:

$$\begin{aligned}
\text{(No Gap 1-3)} \quad & x_1 + x_2 + x_3 \geq 1 \\
\text{(No Gap 2-4)} \quad & x_3 + x_4 \geq 1 \\
\text{(No Gap 5-7)} \quad & x_5 + x_6 + x_7 \geq 1 \\
\text{(No Gap 6-8)} \quad & x_6 + x_7 + x_8 \geq 1 \\
\text{(No Gap 7-9)} \quad & x_8 + x_9 \geq 1 \\
\text{(Conflicts)} \quad & x_1 + x_8 \leq 1 \\
& x_2 + x_6 \leq 1 \\
\text{(Number)} \quad & x_1 + x_2 + \cdots + x_9 \leq 4
\end{aligned}$$

Explanation

The above solution ensures that there are no gaps in difficulty level by making sure that within each sliding window of difficulty levels of a certain width, there is at least one exercise included. For example, the "No Gap 1-3" constraint ensure that there is at least one exercise of difficulty level 1, 2 or 3. Otherwise, then there would be a jump from Exercise 0 to something with difficulty level 4 or above. Similarlly, "No Gap 2-4" ensures that there is at least one exercise of difficulty 2, 3 or 4. The above solution omits certain of these gap constraints (like "No Gap 3-5" or "No Gap 4-6"), because they are implied by other constraints, but it's not wrong to include those as well.

Other Solutions

There are at least 3 different looking solutions among those submitted. One solution is the symmetric version of Sample Solution 1, in which we start by assuming Exercise 10 is included and for each other exercise, ensure that there is at least one exercise of higher difficulty whose level is within 3 of that exercise.

Another solution is to examine every pair of exercises that cannot be consecutive, and ensure that with something like:

$$x_1 + x_9 \leq 1 + x_2 + \cdots + x_8$$

or equivalently

$$x_1 + x_9 \leq 2(x_2 + \cdots + x_8)$$

Both of the above ensure that if Exercises 1 and 9 are both included, then there needs to be something in between. One would have to do the above with every pair of exercises whose difference in difficulty is at least 4.

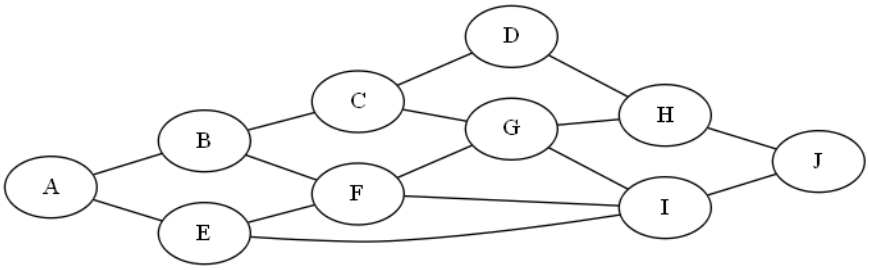
Another potential solution sets up the decision variables differently, so that x_{ij} is whether Exercise i is included as the jth smallest exercise, with $j \in \{1, 2, 3, 4\}$. This solution requires a lot of pre-requisite constraints to ensure that there's no big gap between two consecutive exercises.

Q2. Campus Security under Limited Resources (Abstract Formulation; 10 points)

USC is on holiday break and the Department of Public Safety (DPS) has a limited number of staff. Nevertheless, DPS would still like to use its resources optimally to protect the area around campus. To do so, DPS can station staff at intersections:

- An intersection is said to be **protected** if either it is staffed or if at least one of its neighboring intersections is staffed.
- An intersection is said to be **well-protected** if either it is staffed or if at least two of its neighboring intersections is staffed.
- The **total protection score** is defined to be the total number of protected intersections plus the total number of well-protected intersections.

For example, in the following map, suppose that intersections A and F are both staffed, and no other intersection is staffed. Then the set of protected intersections is $\{A, B, E, F, G, I\}$. The set of well-protected intersections is $\{A, B, E, F\}$. The total protection score is 6+4=10. (By definition, every well-protected intersection is also protected, so a well-protected intersection effectively has a weight of 2 in the total protection score, whereas a protected intersection that is not well-protected has a weight of 1.)



Write an abstract formulation of a linear optimization model to maximize the total protection score subject to staffing at most k intersections. Here, k is an input data variable and denotes the maximum number of intersections that can be staffed. Your formulation must be correct not only for the above example, but also for an arbitrary number of intersections and an arbitrary map describing which pairs of intersections are neighbors. You are free to define any data variables that can be straightforwardly obtained from the map.

Sample Solution 1

Data:

- I: the set of intersections.
- e_{ij}: whether intersections $i \in I$ and $j \in J$ are neighbors. $e_{ii} = 0$ for all i. (Binary)
- k: the maximum number of intersections staffed.

Decision Variables:

- x_i: whether to staff intersection $i \in I$. (Binary)
- y_i: whether intersection $i \in I$ is protected. (Binary)
- z_i: whether intersection $i \in I$ is well-protected. (Binary)

Objective:

$$\text{Max: } \sum_{i \in I} (y_i + z_i)$$

Constraints:

$$y_i \leq x_i + \sum_{j \in I} e_{ij} x_j \quad \text{for each intersection } i \in I.$$

$$2z_i \leq 2x_i + \sum_{j \in I} e_{ij} x_j \quad \text{for each intersection } i \in I.$$

$$\sum_{i \in I} x_i \leq k$$

Explanation

The first constraint ensures that if $y_i = 1$, then intersection i is protected. Similarly, if $z_i = 1$, then intersection i is well-protected. Note that the objective is to maximize the sum of these variables, so that we don't have to enforce the opposite direction: i.e. if intersection i is protected, then $y_i = 1$; or if intersection i is well-protected, then $z_i = 1$.

There are several variants of the above solution as well, with slight changes in the constraints/objective, but the same high-level ideal.

Sample Solution 2

Data:

- I: the set of intersections.
- N_i: the set of neighbors of intersection $i \in I$. (Does not contain intersection i itself.)

Decision Variables:

- x_i: whether to staff intersection $i \in I$. (Binary)
- s_i: the protection score for intersection $i \in I$, which should be 0 if it is unprotected, 1 if it is protected and 2 if it is well-protected. (Integer)

Objective:

$$\text{Max: } \sum_{i \in I} s_i$$

Constraints:

$$s_i \leq 2x_i + \sum_{j \in N_i} x_j \quad \text{for each intersection } i \in I.$$

$$s_i \leq 2 \quad \text{for each intersection } i \in I.$$

$$\sum_{i \in I} x_i \leq k$$

Explanation

The above solution ensures that s_i is zero if intersection i is not protected. Moreover, s_i is at most 1 if intersection i is protected but not well-protected. Finally, s_i is at most 2 if intersection i is well-protected. Since the objective is to maximize the sum of the s_i's, the value of s_i at any optimal solution would correspond to the protection score of the intersection.

Q3. Reusable Software for Exercise 11.4 (Gurobi Code; 9 points)

This problem asks you to implement the following abstract formulation for Exercise 11.4 (Assigning Consultants to Projects). The original problem description can be found in the handout for Week 11 Session 22, but you don't necessarily need it to solve this question.

Data:

- I: the set of consultants.
- J: the set of projects.
- K: the set of skills.
- a_{ik}: whether consultant i possesses skill k. (Binary)
- r_{jk}: the number of consultants of skill k needed for project j.
- S: the set of project pairs (j_1, j_2) that conflict with one another.
- c_{ij}: travel cost of consultant i to project j.

Decision Variables:

- x_{ij}: whether to assign consultant i to project j. (Binary)

Objective and Constraints:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i \in I, j \in J} c_{ij} x_{ij} \\ \text{subject to:} \quad & \sum_{i \in I} a_{ik} x_{ij} \geq r_{jk} \quad \text{for each project } j \in J \text{ and each skill } k \in K. \\ & x_{ij_1} + x_{ij_2} \leq 1 \quad \text{for each consultant } i \in I, \text{ and each pair } (j_1, j_2) \in S. \end{aligned}$$

Write a function called `assignConsultants` with one input argument:

- **inputFile**: An Excel file containing all of the input data needed for the formulation. The format is described below.

The input file is an Excel spreadsheet with four sheet. The first sheet is called "Skills" and encodes the a_{ik}'s:

	A	B	C
1	Consultant	Accounting	Operations
2	Alice	1	
3	Bob	1	1
4	Charlie		1
5	Daphne	1	1
6			

The second sheet is called "Requirements" and encodes the r_{jk}'s:

	A	B	C
1	Project	Accounting	Operations
2	P1	2	1
3	P2	1	1
4	P3	0	2

The third sheet is called "Costs" and encodes the c_{ij}'s:

	A	B	C	D
1	Costs	P1	P2	P3
2	Alice	10	0	5
3	Bob	8	15	13
4	Charlie	0	5	10
5	Daphne	10	3	0

The fourth sheet is called "Conflicts" and encodes which pair of projects are in conflict, as below:

	A	B
1	j1	j2
2	P1	P2
3	P2	P3

In the above picture, the set of conflicts is $S = \{(P1, P2), (P2, P3)\}$, as in the original description of Exercise 11.4 in Session 22. Your code must be able to handle arbitrary input files of the same format.

Your function should print the optimal objective value and return a Pandas DataFrame describing the set of consultants assigned to each project. The format of the output message and of the DataFrame must be exactly as in the sample outputs below.

[1]:
```
# Sample Solution
def assignConsultants(inputFile):
    import pandas as pd
    a=pd.read_excel(inputFile,sheet_name='Skills',index_col=0).fillna(0)
    r=pd.read_excel(inputFile,sheet_name='Requirements',index_col=0)
    c=pd.read_excel(inputFile,sheet_name='Costs',index_col=0)
    conflicts=pd.read_excel(inputFile,sheet_name='Conflicts')
    I=a.index
    J=r.index
    K=a.columns
    from gurobipy import Model, GRB
    mod=Model()
    x=mod.addVars(I,J,vtype=GRB.BINARY)
    mod.setObjective(sum(c.loc[i,j]*x[i,j] for i in I for j in J))
    for j in J:
        for k in K:
            mod.addConstr(sum(a.loc[i,k]*x[i,j] for i in I)>=r.loc[j,k])
    for i in I:
        for ind in conflicts.index:
            j1=conflicts.loc[ind,'j1']
            j2=conflicts.loc[ind,'j2']
            mod.addConstr(x[i,j1]+x[i,j2]<=1)
    mod.setParam('OutputFlag',False)
    mod.optimize()
    print('Minimum total cost:', mod.objval)
    data=[]
    for j in J:
        for i in I:
            if x[i,j].x:
                data.append([j,i])
    df=pd.DataFrame(data,columns=['Project','Consultant'])
    return df
```

[2]:
```
# Sample run
assignConsultants('sample-final-B-input.xlsx')
```

```
Minimum total cost: 36.0
```

	Project	Consultant
0	P1	Bob
1	P1	Daphne
2	P2	Alice
3	P2	Charlie
4	P3	Bob
5	P3	Daphne