Solutions to Sample Final Exam D

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. Optimizing Cooking Order (Concrete Formulation; 11 points)

Raj is a skilled cook and would like to apply optimiztaion to make his time in the kitchen more efficient. He is attending a potluck later today and would like to cook as much as he can to contribute to the potluck. Since he does not want to spend all day cooking, Raj gave himself a time limit of 1 hour to finish all the preparation and cooking. Below are four dishes he is good at, along with the requisite time for preparing the dish, as well as the time it takes to cook it. Ideally, he wants to bring all of the dishes to the potluck if he can finish all on time. But if he cannot finish all, Raj estimated an interest score for each dish, representing how much his friends would appreciate the dish, and he would like to maximize the total interest scores of the dishes he brings to the potluck.

Dish	Preparation Time	Cooking Time	Interest Score
Baked Salmon	10 Min	10 Min	6
Chicken Curry	20 Min	30 Min	10
Lamb Vindaloo	20 Min	40 Min	8
Salad	10 Min	0	4

As can be seen, each dish has a preparation phase and a cooking phase. For any given dish, the preparation for that dish must be completed before cooking for that dish begins. Raj has enough kitchen supplies and oven/stovetop space such that the cooking portion of all of the dishes can take place simultaneously, and he is so good at multi-tasking that he can manage all of the ongoing cooking at the same time, while possibly preparing for another dish. However, the preparation phase requires his active attention, so Raj canot prepare two dishes at the same time. Moreover, for any given dish, he would like to finish all of the preparations in a consecutive chunk of time and start its cooking before moving on to prepare another dish.

For example, if Raj starts at 11am, then he would like to finish everything by noon, at which time he would travel to the potluck. If he first makes the baked salmon, then he would be preparing it until 11:10am and begin cooking it in the oven then. At 11:10am he can decide to start any of the other dishes. If he chooses to do the chicken curry next, then he would be preparing that dish until 11:30am and start cooking it on the stove top at 11:30am, and he would just have enough time to finish that dish by noon. At 11:30am, he can only prepare the salad, because if he started preparing the lamb vindaloo at 11:30am, he would only be able to finish cooking it by 12:30pm, which is too late for the noon deadline.

Write a concrete formulation of a linear optimization model to help Raj plan his order for

preparing dishes, so that he can maximize the total interest score of the dishes that he can bring to the potluck. The objective and constraints must be linear. Note that because all of the preparation and cooking times are multiples of 10 minutes, the only start times that make sense for preparing a new dish are 11am, 11:10am, 11:20am, 11:30am, 11:40am, and 11:50am.

Solution 1: formulation with symmetrical decision variables

Decision Variables:

- $B_0, B_1, B_2, B_3, B_4, B_5$: whether to start the Baked salmon dish at times 11am, 11:10am, 11:20am, 11:30am, 11:40am, and 11:50am respectively (assuming start time of 11am). (Binary)
- $C_0, C_1, C_2, C_3, C_4, C_5$: analogous decision variables for Chicken curry. (Binary)
- $L_0, L_1, L_2, L_3, L_4, L_5$: analogous decision variables for Lamb vindaloo. (Binary)
- $S_0, S_1, S_2, S_3, S_4, S_5$: analogous decision variables for Sald. (Binary)
- B, C, L, S: whether to bring each of the four dishes to the potluck. (Binary)

Objective and Constraints:

$$
\begin{aligned}
\text{Maximize} \quad & 6B + 10C + 8L + 4S \\
\text{s.t.} \quad & B_0 + B_1 + B_2 + B_3 + B_4 + B_5 = B \\
& C_0 + C_1 + C_2 + C_3 + C_4 + C_5 = C \\
& L_0 + L_1 + L_2 + L_3 + L_4 + L_5 = L \\
& S_0 + S_1 + S_2 + S_3 + S_4 + S_5 = S \\
& B_0 + C_0 + L_0 + S_0 \leq 1 \\
& B_1 + C_1 + L_1 + S_1 \leq 1 \\
& B_2 + C_2 + L_2 + S_2 \leq 1 \\
& B_3 + C_3 + L_3 + S_3 \leq 1 \\
& B_4 + C_4 + L_4 + S_4 \leq 1 \\
& B_5 + C_5 + L_5 + S_5 \leq 1 \\
& B_5 = 0 \\
& C_2, C_3, C_4, C_5 = 0 \\
& L_1, L_2, L_3, L_4, L_5 = 0 \\
& C_0 + B_1 \leq 1 \\
& C_0 + L_1 \leq 1 \\
& C_0 + S_1 \leq 1 \\
& C_1 + B_2 \leq 1 \\
& C_1 + L_2 \leq 1 \\
& C_1 + S_2 \leq 1 \\
& L_0 + B_1 \leq 1 \\
& L_0 + C_1 \leq 1 \\
& L_0 + S_1 \leq 1
\end{aligned}
$$

Solution 2: eliminating impossible decision variables up front

Decision Variables:

- $S_0, S_1, S_2, S_3, S_4, S_5$: whether to start the Salad at times 11am, 11:10am, 11:20am, 11:30am, 11:40am, and 11:50am respectively (assuming start time of 11am). (Binary)
- B_0, B_1, B_2, B_3, B_4: analogous decision variables for the Baked salmon. (Binary) Note that the Salmon cannot start at 11:50am for the cooking to finish, so we don't need to consider B_5.

- C_0, C_1: analogous decision variables for Chicken curry. (Binary) Note that the only possible start times for the chicken are 11am and 11:10am in order to meet the noon deadline.
- L_0: analogous decision variables for Lamb vindaloo. (Binary)
- B, C, L, S: whether to bring each of the four dishes to the potluck. (Binary)

Objective and Constraints:

$$\begin{aligned}
\text{Maximize} \quad & 6B + 10C + 8L + 4S \\
\text{s.t.} \quad & B_0 + B_1 + B_2 + B_3 + B_4 = B \\
& C_0 + C_1 = C \\
& L_0 = L \\
& S_0 + S_1 + S_2 + S_3 + S_4 + S_5 = S \\
& B_0 + C_0 + L_0 + S_0 \leq 1 \\
& B_1 + C_1 + S_1 \leq 1 \\
& B_2 + S_2 \leq 1 \\
& B_3 + S_3 \leq 1 \\
& B_4 + S_4 \leq 1 \\
& C_0 + B_1 \leq 1 \\
& C_0 + S_1 \leq 1 \\
& C_1 + B_2 \leq 1 \\
& C_1 + S_2 \leq 1 \\
& L_0 + B_1 \leq 1 \\
& L_0 + C_1 \leq 1 \\
& L_0 + S_1 \leq 1
\end{aligned}$$

Q2. Warehouse Sourcing (Abstract Formulation; 10 points)

A warehouse needs to make sure it has enough inventory in each week to fulfill all customer demand based on its forecast. However, the warehouse can control when it receives shipments of inventory from suppliers and would like to optimize the timing of shipments so as to minimize the combination of the following two types of costs:

1. **fixed cost of receiving a single shipment from the supplier**: each time the warehouse receives a shipment, there is a certain fixed cost, which does not depend on the amount of inventory received. Hence, it is preferable to receive shipments in large batches so as to need fewer shipments.
2. **holding cost**: carrying extra inventory from the end of one week to the next requires storage space, so that having low inventory at the end of each week is desired.

For example, suppose that the fixed cost of receiving each shipment is 100 dollars and the cost of carrying each unit of inventory from the end of one week to the next is 2 dollars. Suppose that the warehouse forecasts the following demand in the next eight weeks.

Week:	1	2	3	4	5	6	7	8
Demand:	20	10	0	40	50	10	0	30

The beginning inventory at the end of Week 0 is zero, and shipments from the supplier arrive at the beginning of a week before any demand needs to be fulfilled. Suppose that the warehouse wants to receive only one shipment, then it needs to receive 160 units at the beginning of Week 1 in order to have enough to fulfill all

demand in the 8 weeks. Call this plan A. Under plan A, the amount of leftover inventory at the end of each week is as follows:

Week:	1	2	3	4	5	6	7	8
Leftover inventory under plan A:	140	130	130	90	40	30	30	0

Hence, the total holding cost is $2 \times (140 + 130 + 130 + 90 + 40 + 30 + 30 + 0) = 1180$. Since only one shipment needs to be received, the total cost of plan A is $1180 + 100 \times 1 = 1280$.

An alternative plan, which we refer to as plan B, is to receive a separate shipment at the beginning of each week with positive demand, and to receive just enough to fulfil the demand for that week. The total holding cost under this plan is 0, since all received inventory will be used up by the end of a week, and the total fixed cost is $100 \times 6 = 600$ (since there are 6 weeks with non-zero demand in this example). The total cost of plan B is $0 + 600 = 600$, which is lower than plan A.

The **optimal plan** in this example is to receive 4 shipments from the supplier: receive 30 units at the beginning of Week 1, 40 units at the beginning of Week 4, 60 units at the beginning of Week 5, and 30 units at the beginning of Week 8. Under this plan, the leftover inventory at the end of each week is as follows:

Week:	1	2	3	4	5	6	7	8
Leftover inventory under plan A:	10	0	0	0	10	0	0	0

Hence, the total cost is $2 \times 20 + 100 \times 4 = 440$, which is lower than both plans A and B.

Write an abstract formulation of a linear optimization model to solve for the optimal plan. Your formulation must be able to handle arbitrary number of weeks, arbitrary unit holding cost, arbitrary fixed cost of receiving a shipment, as well as arbitrary demand in each week. You can assume that the inventory at the end of Week 0 is zero, and that accurate demand forecasts are given. You can define whatever data variables you need, as long as they can be straighforwardly obtained from the given data. Your objective and constraints must all be linear.

Data:

- n: the number of weeks to plan.
- d_t: the demand in week $t \in T$.
- h: unit holding cost.
- f: fixed cost of receiving each shipment.
- $T = \{1, 2, \cdots, n\}$.
- $M = \sum_{t \in T} d_t$.
- $y_0 = 0$.

Decision Variables:

- x_t: the amount of inventory to receive at the beginning of week $t \in T$. (Continuous)
- y_t: the amount of leftover inventory at the end of week $t \in T$. (Continuous)
- z_t: whether to receive a shipment at the beginning of week $t \in T$. (Binary)

Objective and Constraints:

$$
\begin{aligned}
\text{Minimize} \quad & f \sum_{t \in T} z_t + h \sum_{t \in T} y_t & & \\
\text{s.t.} \quad & y_t = y_{t-1} + x_t - d_t & & \text{for each week } t \in T. \\
& x_t \le M z_t & & \text{for each week } t \in T. \\
& x_t, y_t \ge 0 & & \text{for each } t \in T.
\end{aligned}
$$

Q3. Network Flow (Gurobi Optimization; 9 points)

This question asks you to implement an abstract formulation for the generalized version of the Sample Problem from Session 18. A company manufactures a type of heavy machinery in city 0 and would like to determine the fastest rate at which it can deliver machines to customers in city n. (Rate, or throughput, is measured in the average number of machines delivered per day.) The bottleneck is that the company must use a special type of truck to ship the machine, and a limited number of these trucks travel each day from one city to an adjacent one. Each truck can carry only one machine at a time. The daily number of trucks going between each pair of city is given and cannot be controlled by the company. The following bstract formulation can be used to solve this problem.

Data Variables:

- n: the number of cities excluding the source, City 0.
- C: the set of cities. $C = \{0, 1, \cdots, n\}$.
- $q_{i,j}$: the number of trucks traveling from city $i \in C$ to city $j \in C$ each day.

Decision Variables:

- $x_{i,j}$: the rate of shipping from city $i \in C$ to city $j \in C$. (Continuous)

Objective and Constraints:

$$\text{Maximize:} \quad \sum_{i \in C} x_{i,n}$$

s.t.

$$\text{(In-flow = Out-flow)} \quad \sum_{j \in C} x_{i,j} = \sum_{j \in C} x_{j,i} \quad \text{For each intermediate city } i \in \{1, 2, \cdots, n-1\}.$$

$$\text{(No out-flow from destination)} \quad x_{n,i} = 0 \quad \text{For each city } i \in C.$$

$$\text{(Truck capacity)} \quad x_{i,j} \leq q_{i,j} \quad \text{For all } i, j \in C.$$

$$\text{(Non-negativity)} \quad x_{i,j} \geq 0 \quad \text{For all } i, j \in C.$$

Write a function called "optimize" with two input parameters:

- **inputFile**: filename of a CSV file providing the data on the truck capacities. The format is described below.
- **outputFile**: the name of an Excel file to which you will write the results of the optimization.

The input CSV file has the following format:

	A	B	C
1	Origin	Destination	Capacity
2	0	1	8
3	1	0	8
4	0	2	5
5	2	0	5
6	1	2	2
7	2	1	2
8	1	3	4
9	3	1	4
10	2	3	10
11	3	2	10

The table lists the $q_{i,j}$ for each pair of origin and destination cities with positive truck capacity. (Capacity $q_{i,j}$ is defined as the number of trucks traveling each day from i to j, which the company can use if needed.) For

any pair (i, j) that does not exist in this table, you should assume that the capacity $q_{i,j} = 0$. You can infer the value of n from the maximum value of the column "Destination," which is 3 in the above sample input. You can assume that all values in the columns "Origin" and "Destination" are among the set $\{0, 1, \cdots, n\}$.

Your function should write two sheets to the outputFile. The first sheet should be called "Objective" and should give the optimal objective value. The second sheet should be called "Shipment" and should give optimal values of $x_{i,j}$'s (only positive values should be listed in the output). For the above sample input, the first sheet "Objective" should look like this:

	A
1	**Maximum Rate**
2	11

The second sheet "Shipment" should look like this:

	A	B	C
1	**Origin**	**Destination**	**Shipment**
2	0	1	6
3	0	2	5
4	1	2	2
5	1	3	4
6	2	3	7

[2]:
```
import pandas as pd
from gurobipy import Model, GRB
def optimize(inputFile,outputFile):
    data=pd.read_csv(inputFile)
    n=data['Destination'].max()
    q={}
    for i in range(n+1):
        for j in range(n+1):
            q[i,j]=0
    for ind in data.index:
        i=data.loc[ind,'Origin']
        j=data.loc[ind,'Destination']
        q[i,j]=data.loc[ind,'Capacity']
    mod=Model()
    C=range(n+1)
    x=mod.addVars(C,C)
    mod.setObjective(sum(x[i,n] for i in C),sense=GRB.MAXIMIZE)
    for i in range(1,n):
        mod.addConstr(sum(x[i,j] for j in C)==sum(x[j,i] for j in C))
    for i in C:
        mod.addConstr(x[n,i]==0)
    for i in C:
        for j in C:
            mod.addConstr(x[i,j]<=q[i,j])
    mod.setParam('OutputFlag',False)
    mod.optimize()
    writer=pd.ExcelWriter(outputFile)
    pd.DataFrame([mod.objVal],columns=['Maximum Rate']).
↪to_excel(writer,sheet_name='Objective',index=False)
```

```
        table=[]
        for i in C:
            for j in C:
                if x[i,j].x>0:
                    table.append([i,j,x[i,j].x])
        pd.DataFrame(table,columns=['Origin','Destination','Shipment']).
↪to_excel(writer,sheet_name='Shipment',index=False)
        writer.close()
```

[3]:
```
# Test Code
optimize('sample-final-D-input.csv','sample-final-D-output.xlsx')
```