Solutions to Final Exam 2023

Q1. Kitchen Remodeling (Concrete Formulation; 11 Points)

Katherine is remodeling her kitchen and would like to decide on the cabinet design, countertop type, and flooring style. Each option differs in terms of looks, price, and lead time. Her contractor only wants to start the project when all the materials have been delivered, which is based on the maximum lead time of all the options she select. The following table summarizes the options she is considering.

Category	Option	Looks-Score	Price	Lead Time
Cabinet Design	A_1	7	10	3
	A_2	5	8	2
	A_3	4	7	1
Countertop Type	B_1	5	4	2
	B_2	4	2	1
	B_3	1	1	1
Flooring Style	C_1	4	8	1
	C_2	3	7	1

As shown above, there are three cabinet designs to choose from, three countertop types, and two flooring styles. Within each category, she must select exactly one option. For example, if she selects cabinet A_1, countertop B_1 and flooring C_1, then the total looks-score would be $7+5+4=16$. The total price would be $10+4+8=22$ (thousands of dollars). The maximum lead time is $\max(3,2,1)=3$ (months). In addition to the above, there are certain combinations of options that look better together, as summarized below:

Combo	Additional Looks-Score
A_1+B_3	3
$A_2+B_2+C_2$	2

In words, if she selects cabinet A_1 and countertop B_3, then she gets an additional boost of 3 to the total looks-score. Similarly, if she selects cabinet A_2, countertop B_2 and flooring C_2, then the total looks-score would be $5+4+3+2=14$, which includes the additional looks-score of 2 given in the second row above.

Formulate a linear optimization model to help her optimally trade off the total looks-score, total price, and maximum lead time. The objective function should correspond to

$$(\text{Total Looks-Score}) - \alpha(\text{Maximum Lead Time}),$$

where α is a known constant (not a decision variable). Beside ensuring that she selects exactly one option from each category, your formulation should sure that (Total Price) $\leq \beta$, where β is her budget and is also a known constant. (You can simply use α and β in your formulation and treat them like numbers, and you don't have to define them.) **You must ensure that the formulation is linear.**

Decision Variables:

- $A_1, A_2, A_3, B_1, B_2, B_3, C_1, C_2$: whether she selects each option. (Binary)
- Y_1, Y_2: auxiliary decision variables corresonding to whether she gets the two boosts. (Binary)
- T: auxiliary decision variable corresponding to the maximum lead time. (Continuous)
- L: auxiliary decision variable corresponding to the total looks score. (Continuous)

Objective:

$$\text{Maximize: } L - \alpha T$$

Constraints:

$$
\begin{aligned}
7A_1 + 5A_2 + 4A_3 + 5B_1 + 4B_2 + B_3 + 4C_1 + 3C_2 + 3Y_1 + 2Y_2 &= L \\
10A_1 + 8A_2 + 7A_3 + 4B_1 + 2B_2 + B_3 + 8C_1 + 7C_2 &\leq \beta \\
A_1 + A_2 + A_3 &= 1 \\
B_1 + B_2 + B_3 &= 1 \\
C_1 + C_2 &= 1 \\
Y_1 &\leq A_1 \\
Y_1 &\leq B_3 \\
Y_2 &\leq A_2 \\
Y_2 &\leq B_2 \\
Y_2 &\leq C_2 \\
3A_1 &\leq T \\
2A_2 &\leq T \\
A_3 &\leq T \\
2B_1 &\leq T \\
B_2 &\leq T \\
C_1 &\leq T \\
C_2 &\leq T
\end{aligned}
$$

The auxiliary variable L is unnecessary, as one can simply substitute in the first constraint into the objective function. The constraints relating Y_1 and Y_2 with the main decision variables are necessary to ensure that she gets the boost associated with Y_1 only when both A_1 and B_3 are selected. Similarly, she gets the boost associated with Y_2 only when A_2, B_2 and C_2 are all selected. The final set of constraints ensure that the objective function penalizes options that have a long lead time. The penalty is at least 3 whenever she selects A_1; the penalty is at least 2 whenever she selects A_2, and so on. Because the numbers are such that the lead time is at least 1 for all options, and either flooring style has a lead time of 1 (month), the last seven constraints can be simplified to the following four:

$$
\begin{aligned}
3A_1 &\leq T \\
2A_2 &\leq T \\
2B_1 &\leq T \\
1 &\leq T
\end{aligned}
$$

Q2. Email Marketing (Abstract Formulation; 10 Points)

A company would like to use optimization to decide which email ads to send to each potential customer. Suppose that there are n customers and m ads. For each customer i and each ad j, the company used machine learning to estimate two parameters: p_{ij} and s_{ij}. Here, p_{ij} is the expected profit from sending ad j to customer i, and s_{ij} is customer i's interest score for ad j. The company would like to maximize its total expected profit subject to the following constraints:

1. Each customer i receives at most k_i ads. Moreover, a customer cannot get the same ad more than once.
2. The average interest score of ads received by each customer is at least a_i. For example, if a customer receives 3 ads, and her interest scores for them are 5, 1 and 3 respectively, then her average interest score is $(5 + 1 + 3)/3 = 3$.
3. Certain pairs of ads are overlapping and cannot be sent to the same customer. For example, suppose that the set of overlapping pairs of ads is $\{(A, B), (A, C), (B, D)\}$, then ads A and B cannot be sent

to the same customer, ads A and C cannot be sent to the same customer, and ads B and D cannot be sent to the same customer.

4. The number of customers who receive each ad is within a factor of 2 of each other. In other words, there does not exist two ads such that the number of customers who receive the first ad is more than double the number of customers who receive the second ad.

Write an abstract formulation of a linear optimization model to implement the above logic. You must ensure that the formulation is linear. For your convenience, some of the data variables are already provided, but you may need to add other data variables.

Data:

- I: the set of customers.
- J: the set of ads.
- p_{ij}: the expected profit of sending ad $j \in J$ to customer $i \in I$.
- s_{ij}: the interest score of ad $j \in J$ for customer $i \in I$.
- k_i: the maximum number of ads that can be sent to customer i.
- a_i: the minimum average interest score of ads sent to customer i.
- S: the set of overlapping pairs of ads. Each element (j_1, j_2) is such that ad j_1 cannot be sent to the same customer as ad j_2.

Decision Variables:

- x_{ij}: whether to send ad j to customer i. (Binary)
- z: an auxiliary decision variable used to implement constraint 4. (Continuous)

Objective:

$$\text{Maximize: } \sum_{i \in I, j \in J} p_{ij} x_{ij}$$

Constraints:

$$\sum_{j \in J} x_{ij} \leq k_i \qquad \text{for each customer } i \in I.$$

$$\sum_{j \in J} s_{ij} x_{ij} \geq a_i \sum_{j \in J} x_{ij} \qquad \text{for each customer } i \in I.$$

$$x_{ij_1} + x_{ij_2} \leq 1 \qquad \text{for each customer } i \in I \text{ and each overlapping pair } (j_1, j_2) \in S.$$

$$z \leq \sum_{i \in I} x_{ij} \leq 2z \qquad \text{for each ad } j \in J.$$

The last constraint above ensures that the number of people who receive each ad is within a certain interval $[z, 2z]$, where the suitable lower bound z is chosen by the computer. An alternative constraint is as follows:

$$\sum_{i \in I} x_{ij_2} \leq 2 \sum_{i \in I} x_{ij_1} \qquad \text{for each pair of ads } j_1, j_2 \in J.$$

This directly ensures that for each pair of ads (j_1, j_2), the number of people who receive the second ad is not more than double the number of people who receive the first ad.

Q3. Diet Optimization (Gurobi Coding; 9 points)

This problem asks you to implement the abstract formulation from Q2 of Sample Final Exam C, which is about optimizing one's diet to minimize cost while satisfying minimal nutritional requirements. The abstract formulation is given below.

Data:

- I: set of foods.
- J: set of nutrients.
- c_i: cost of food i.
- a_{ij}: amount of nutrient j in one serving of food i.
- l_j: lower bound in daily intake of nutrient j.
- u_j: upper bound of nutrient j.
- s_i: minimum number of servings of food i to have if one is to have any such food at all.
- t_i: maximum number of servings of food i to have.

Decision Variables:

- X_i: amount of food i in the diet. (continuous)
- Z_i: whether to include food i at all. (binary)

Objective and Constraints:

$$\begin{aligned} \text{Min.} \quad & \sum_{i \in I} c_i X_i \\ \text{s.t.} \quad & \\ & l_j \leq \sum_{i \in I} a_{ij} X_i \leq u_j \qquad \text{for each nutrient } j \in J. \\ & s_i Z_i \leq X_i \leq t_i Z_i \qquad \text{for each food } i \in I. \\ & X_i \geq 0 \qquad \text{for each food } i \in I. \end{aligned}$$

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

- **inputFile**: the name of an input Excel file. The file format is described below.

The input Excel file has two sheets. The first sheet is called "Nutrition" and looks like this:

	A	B	C	D	E	F	G	H
1	**Foods**	**Calories**	**Protein**	**Fat**	**Sodium**	**Costs**	**Minimum**	**Maximum**
2	1. ice cream	330	8	15	180	1.59	1	2
3	2. chicken	420	32	10	300	2.89	2	3
4	3. pizza	320	15	20	820	1.99	2	3
5	4. fries	380	4	19	270	1.89	1	3
6	5. macaroni	320	12	10	830	2.09	1	3
7	6. milk	100	8	2.5	125	0.89	1	3
8	7. salad	320	31	2	123	2.49	2	4
9								

The first column gives the set of foods I. The last three columns give the cost c_i, the minimum s_i and the maximum t_i for each food $i \in I$. The rest of the table corresponds to a_{ij}. The second sheet is called "Bounds" and looks like this:

	A	B	C	D	E
1		**Calories**	**Protein**	**Fat**	**Sodium**
2	Lower bound	1800	91	0	0
3	Upper bound	2200	9999	65	1779

This gives the lower bound l_j and upper bound u_j for each nutrient $j \in J$.

The function should return a Series which gives the optimal diet. The Series should only include foods that are consumed a positive amount (so foods that not eaten at all should not be included). See the sample run below for an illustration.

[2]:
```
# Sample run
optimize_diet('diet.xlsx')
```

```
2. chicken    1.841060
7. salad      3.208609
dtype: float64
```

[1]:
```
def optimize_diet(inputFile):
    import pandas as pd
    from gurobipy import Model, GRB
    nutrition=pd.read_excel(inputFile,sheet_name='Nutrition',index_col=0)
    bounds=pd.read_excel(inputFile,sheet_name='Bounds',index_col=0)
    mod=Model()
    I=nutrition.index
    c=nutrition['Costs']
    s=nutrition['Minimum']
    t=nutrition['Maximum']
    J=nutrition.columns[:-3]
    l=bounds.loc['Lower bound',:]
    u=bounds.loc['Upper bound',:]
    x=mod.addVars(I)
    z=mod.addVars(I,vtype=GRB.BINARY)
    mod.setObjective(sum(c[i]*x[i] for i in I))
    for j in J:
        amount=sum(nutrition.loc[i,j]*x[i] for i in I)
        mod.addConstr(amount>=l[j])
        mod.addConstr(amount<=u[j])
    for i in I:
        mod.addConstr(s[i]*z[i]<=x[i])
        mod.addConstr(t[i]*z[i]>=x[i])
    mod.setParam('OutputFlag',False)
    mod.optimize()
    return pd.Series({i: x[i].x for i in I if x[i].x>0})
```