Mixed Integer Programming (3/22)

Learning Objectives:

- Read and write Excel files using `pandas`.
- Define Mixed Integer Programs (MIP) and describe their differences with LPs.
- Model complex constraints using binary variables.

Textbook Readings: DMD 9:1-5
In class Handouts:

- Handout on DataFrames and Excel.
- In class exercises.

DataFrames and Excel

Here are the solutions to the in-class exercise I.

Exercise I

1. Read in the "Fulfilment Centers" sheet from "small_data.xlsx" as a DataFrame and call it `fcs`.
2. Create a dictionary called `cap` that maps each "FC_name" in `fcs` to two times its capacity. (i.e., `caps['A']` should be 2000.) You should use a loop rather than doing this manually.
3. Read in the "Distances" sheet from "small_data.xlsx" as a DataFrame and obtain a Series called `s` corresponding to the column sums. (If you are not getting the right sheet, try to replace `sheet_name` with `sheetname`.)
4. Create a table (as a list of lists) in which the first column corresponds to each "FC_name" (index set of `fcs`), the second column contains the values of the dictionary `cap` for the FC, and the third value corresponds to the value of the Series `s` for the FC. Using this table, create a new DataFrame called `df` with this data, and the column names "FC_name","Twice Capacity", "Sum of distances".
5. Output an Excel file called "exampleOutput.xlsx", in which the first sheet is named "FCs" and is the same as the `fcs` DataFrame, and the second sheet is named "Results" and is the same as the `df` DataFrame from step 4. Inspect the Excel file to make sure the formatting is the way you want it.
6. Run the entire Jupyter notebook again after changing the input file to "data.xlsx"

[1]:
```
import pandas as pd
inputFile='small_data.xlsx'
fcs=pd.read_excel(inputFile,sheet_name='Fulfilment Centers',index_col=0)
fcs
```

FC_name	city	zip	capacity	lat	long
A	Schertz, Texas	78154	1000	29.547068	-98.262642
B	WestColumbia, SouthCarolina	29172	1000	33.912417	-81.076978

[2]:
```
cap={}
for center in fcs.index:
    cap[center]=2*fcs.loc[center,'capacity']
cap
```

```
{'A': 2000, 'B': 2000}
```

```
[3]: distances=pd.read_excel(inputFile,sheet_name='Distances',index_col=0)
     distances
```

region_ID	A	B
0	1.55160	0.62956
1	1.43880	2.25933
2	1.02175	0.43547

```
[4]: s=distances.sum()
     s
```

```
A    4.01215
B    3.32436
dtype: float64
```

Note: If you were not able to read the above sheet, change `sheet_name` to `sheetname`.

```
[5]: table=[]
     for center in fcs.index:
         table.append([center,cap[center],s.loc[center]])
     table
```

```
[['A', 2000, 4.0121500000000001], ['B', 2000, 3.32436]]
```

```
[6]: df=pd.DataFrame(table,columns=['FC_name','Twice Capacity','Sum of distances'])
     df
```

	FC_name	Twice Capacity	Sum of distances
0	A	2000	4.01215
1	B	2000	3.32436

```
[8]: writer=pd.ExcelWriter('exampleOutput.xlsx')
     fcs.to_excel(writer,sheet_name='FCs')
     df.to_excel(writer,sheet_name='Results',index=False)
     writer.save()
```

Introduction to Mixed Integer Programming (MIP)

The only difference between a mixed integer program (MIP) and a linear program is that **in a MIP, some of the variables are constrained to be integers.** In other words, a MIP must have a linear objective function, linear constraints, and variables are either continuous or integers. Because of the allowance of integer constraints, MIPs is a generalization of LP and is thus more flexible. (Having a new type of constraint gives us more tools to play with, we can choose to use it or not, but it's a new option.)

Recall the production planning example from the course notes to session "16-LP Duality" (with two products X and Y, and three resources: material 1, material 2 and labor). Suppose

that using any material 2 at all requires a set up cost of 90. If we pay this cost, then we have 48 units of material 2 at our disposal, otherwise we have no material 2. The profit maximizing production planning problem can be formulated as the following MIP.

$$\begin{aligned}
\text{maximize} \quad & 20X + 10Y - 90M_2 \\
\text{subject to:} \quad & \\
\text{(Material 1)} \quad & 4X + Y \leq 60 \\
\text{(Material 2)} \quad & 2Y \leq 48M_2 \\
\text{(Labor)} \quad & X + Y \leq 30 \\
\text{(Non-negativity)} \quad & X, Y \geq 0 \\
\text{(Binary)} \quad & M_2 \in \{0, 1\}
\end{aligned}$$

In the above, the binary variable M_2 corresponds whether we use material 2 or not. If we don't use it ($M_2 = 0$), then we don't pay the setup cost, but the Material 2 constraint becomes $2Y \leq 0$. If we use it ($M_2 = 1$), then we must pay 90 in the objective, but the Material 2 constraint becomes $2Y \leq 48$.

Note that binary variables are allowed within our definition of MIP because $M_2 \in \{0, 1\}$ can be represented by linear constraint $0 \leq M_2 \leq 1$ and the constraint that M_2 is an integer.

Note that with MIP, we can model situations in which we either don't do something at all, or go all the way. Many business decisions are such:

- Capital investment requires paying the whole cost in order to reap any gain (cannot pay 5% and get 5% results).
- Scheduling requires scheduling a person to a particular shift or not to the shift. (We cannot split a person and assign half a person to a shift.
- When production quantities are low, it may not make sense to approximate the quantity produced as a continuous variable, as we either make 2 units or 3 units, but not 2.5.
- Business strategies are often requir focusing of resources: we either do plan A or plan B, but cannot do half of each.

Comparing MIPS and LPs

With LPs, we cannot model the dichotomous situations above, as all variables in LPs are continuous. Therefore, MIPS are more **more flexible** than LPs, because of the possibility of using more complicated constraints.

However, LPs have a special geometry (see Lecture for 15-Introduction to LP, or DMD 7.3), which MIPs do not have. The difference in geometry makes MIPS **less tractable** than LPs, meaning that it is more difficult to solve. A LP with a million variable and constraints can be solved routinely with Gurobi in an hour, but a MIP of a few hundred or a thousand variables may require days to solve, depending on the particular structure of the problem.

As a result of the difference in geometry, the concept of **shadow prices is not well defined in a MIP**, in the same way as it is for LPs. While one can still compute the change in the optimal objective as the right hand side (RHS) of a constraint changes, one does not get this information for free when solving the MIP. Hence, do not try to obtain the shadow price of a constraint in Gurobi when you have a MIP.

Implementing in Gurobi

The only difference between implementing a MIP in Gurobi and implementing a LP is in the step defining variable.

For a continuous variable $x \geq 0$, we define it as

```
x=mod.addVar(lb=0)
```

For a binary variable $x \in \{0,1\}$, we must change this to

```
x=mod.addVar(vtype=grb.GRB.BINARY)
```

where `grb` is from `import gurobipy as grb`.
For a general integer variable x (x is any integer), we must change this to

```
x=mod.addVar(vtype=grb.GRB.INTEGER)
```

To denote integers, we can use the notation, $x \in \mathbb{Z}$. Here, $\mathbb{Z}$ is the mathematical notation for the set of integers: $\mathbb{Z} = \{\cdots, -5, -4, -3, -2, -1, 0, 1, 2, 3, \cdots\}$.
The following is an implementation of the MIP above.

[11]:
```
# Explicitly constructing a simple production planning LP with fixed cost of getting ma
import gurobipy as grb
mod=grb.Model()

X=mod.addVar(lb=0)
Y=mod.addVar(lb=0)
M2=mod.addVar(lb=0,vtype=grb.GRB.BINARY)

mod.setObjective(20*X+10*Y-90*M2,sense=grb.GRB.MAXIMIZE)

mat1=mod.addConstr(4*X+Y <=60)
mat2=mod.addConstr(2*Y<=48*M2)
labor=mod.addConstr(X+Y<=30)

mod.optimize()

print('\n')
print('Optimal objective: {0:.2f}'.format(mod.ObjVal))
print('Optimal solution:')
print('\tX= {0:.2f}'.format(X.x))
print('\tY= {0:.2f}'.format(Y.x))
print('\tM2= {0:.0f}'.format(M2.x))
```

```
Optimize a model with 3 rows, 3 columns and 6 nonzeros
Variable types: 2 continuous, 1 integer (1 binary)
Coefficient statistics:
  Matrix range     [1e+00, 5e+01]
  Objective range  [1e+01, 9e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [3e+01, 6e+01]
Found heuristic solution: objective 300.0000000
Presolve time: 0.00s
Presolved: 3 rows, 3 columns, 7 nonzeros
Variable types: 2 continuous, 1 integer (1 binary)

Root relaxation: objective 3.100000e+02, 2 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
```

```
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0     310.0000000  310.00000  0.00%     -    0s

Explored 0 nodes (2 simplex iterations) in 0.05 seconds
Thread count was 4 (of 4 available processors)

Solution count 2: 310 300

Optimal solution found (tolerance 1.00e-04)
Best objective 3.100000000000e+02, best bound 3.100000000000e+02, gap 0.0000%


Optimal objective: 310.00
Optimal solution:
        X= 10.00
        Y= 20.00
        M2= 1
```

The first lines (before "Optimal objective: 310.00") are print outs from the Gurobi MIP solver listing the steps it took to solve the MIP. You can read more about how MIP solvers work in DMD section 9.4. You can suppress this output using `mod.setParam('OutputFlag',False)` as with LPs.

Modeling with Binary Variables

Binary variables can be used to model complex constraints. For example, let X, Y, and Z be binary variables, representing whether we want to invest in project X, Y or Z respectively.

- If X then Y (Project Y is a pre-requisite for X): $X \leq Y$.
- X and Y are mutually exclusive (If X then not Y): $X + Y \leq 1$.
- Need at least two projects: $X + Y + Z \geq 2$.
- Budget constraint (The projects cost 3,2, and 5 million respectively and we have 5 million): $3X + 2Y + 5Z \leq 5$.
- Turning on/off a constraint: Suppose we have a labor constraint

  $$X + Y \leq 30$$

  as in the production planning example. However, we want to compare having this constraint, or out-sourcing our labor altogether, and pay a fee of 50. In this case, we can have a binary variable Z and do

  $$X + Y \leq 30 + MZ$$

where M is a huge constant, like 1000000. We would subtract $50Z$ from the objective. If $Z = 0$, then we don't out-source, so we have the original labor constraint. If $Z = 1$, then we pay 50 in the objective, but the above labor constraint becomes $X + Y \leq 30 +$ Huge constant, which effectively removes the labor constraint altogether.

Here is the in-class modeling exercise and its solution:

Exercise II

Alice, Bob and Charlie are three professors who each teach 2 sections, but must share the same special room. For simplicity, there are 6 time slots at the room, indexed from 0 to 5. (Slot 0

represents Monday morning, 1 represents Monday afternoon, 2 represents Monday evening, 3 represents Tuesday morning, 4 Tuesday afternoon, and 5 Tuesday evening.)

Decision Variables: Let a_0 be a binary variable denoting whether Alice teaches in slot 0, Similarly define $a_1, \cdots, a_5$ and the same for Bob and Charlie. Let m_a be a binary variable denoting whether Alice teaches on Mondays. Similarly define m_b, m_c, as well as t_a, t_b, t_c for Tuesdays.

Model each of the following using a linear expression or a series of linear constraints:

- Linear expression: The total number days Alice teaches, plus the total number of days Bob teaches, plus the total number of days Charlie teaches. (Think of this as the objective to minimize, as professors want to get all teaching done in the same day.)

$$m_a + m_b + m_c + t_a + t_b + t_c$$

- Constraint: slot 0 can have at most one person assigned. (Similarly for slots 1 through 5.)

$$a_0 + b_0 + c_0 \leq 1$$

- Constraint: Alice must be assigned to exactly 2 slots. (Similarly for Bob and Charlie.)

$$a_0 + a_1 + \cdots + a_5 = 2$$

- Constraints: Bob cannot teach in the mornings. Moreover, Charlie works at another job on Mondays and cannot teach Monday mornings and afternoons.

$$b_0 = 0 \qquad b_3 = 0 \qquad c_0 = 0 \qquad c_1 = 0$$

- Constraints: Alice and Bob cannot teach on the same day.

$$m_a + m_b \leq 1 \qquad t_a + t_b \leq 1$$

- Constraints: If Alice teaches two slots on a given day, they must be consecutive slots.

The key here is to rephrase the constraint as she cannot teach the morning and evening slots of any given day at the same time.

$$a_0 + a_2 \leq 1 \qquad a_3 + a_5 \leq 1$$

- Logical Constraints: If Alice teaches Monday mornings (a_0), then she must also necessarily teach on Mondays (m_a). Similarly, if she teaches Monday afternoons (a_1), then she must be teaching on Mondays (m_a). Write the 6 logical constraints connecting $a_0, a_1, \cdots, a_5$ to m_a and t_a.

$$a_0 \leq m_a \quad a_1 \leq m_a \quad a_2 \leq m_a$$
$$a_3 \leq t_a \quad a_4 \leq t_a \quad a_5 \leq t_a$$

Note: in a problem with more people, you may want to define decision variable x_{a0} for whether Alice teaches in slot 0, rather than a_0. In other words, x_{ij} would be a binary variable denoting whether person i teaches in slot j.