Week 10: Using the Gurobi Solver

Session 20: Automating Patterns with Multiple Indices

Example: Numerical Solution for Supply Chain Planning

Recap of Exercise 8.2: The following table provides the shipping cost for a certain item, from three of Amazon's fulfillment centers (FC) to four regions (A, B, C and D).

Region FC	1	2	3
A. Kings County, NY	20	8	25
B. Los Angeles County, CA	18	23	8
C. King County, WA	21	24	8
D. Harris County, TX	8	8	19

The following table summarizes the weekly demand for the item from each region.

Region A	Region B	Region C	Region D
30	50	10	20

Suppose that each FC is able to ship up to 40 units each week in total. Formulate a linear program to determine the minimum transportation cost needed to satisfy all demand while respecting FC capacities, as well as the optimal shipment plan.

Concrete Formulation

Decision variables: Let x_{ij} denote the amount to transport from FC i to region j. (Integer)
Objective:

$$\begin{aligned}\text{Minimize: } & 20x_{1A} + 18x_{1B} + 21x_{1C} + 8x_{1D} + \\ & 8x_{2A} + 23x_{2B} + 24x_{2C} + 8x_{2D} + \\ & 25x_{3A} + 8x_{3B} + 8x_{3C} + 19x_{3D}\end{aligned}$$

Constraints:

$$\begin{aligned}\text{(Capacity 1)} \quad & x_{1A} + x_{1B} + x_{1C} + x_{1D} \leq 40 \\ \text{(Capacity 2)} \quad & x_{2A} + x_{2B} + x_{2C} + x_{2D} \leq 40 \\ \text{(Capacity 3)} \quad & x_{3A} + x_{3B} + x_{3C} + x_{3D} \leq 40 \\ \text{(Demand A)} \quad & x_{1A} + x_{2A} + x_{3A} \geq 30 \\ \text{(Demand B)} \quad & x_{1B} + x_{2B} + x_{3B} \geq 50 \\ \text{(Demand C)} \quad & x_{1C} + x_{2C} + x_{3C} \geq 10 \\ \text{(Demand D)} \quad & x_{1D} + x_{2D} + x_{3D} \geq 20 \\ \text{(Non-negativity)} \quad & x_{ij} \geq 0 \qquad \text{for all } i \text{ and } j.\end{aligned}$$

The following code implements the above, while avoiding hard-coding in the numbers by obtaining them from appropriate data structures.

Input Data

```
[1]: import pandas as pd
     cost=pd.DataFrame([[20,18,21,8],[8,23,24,8],[25,8,8,19]],\
                       index=[1,2,3],columns=['A','B','C','D'])
     cost
```

```
    A   B   C   D
1  20  18  21   8
2   8  23  24   8
3  25   8   8  19
```

```
[2]: demand=pd.Series([30,50,10,20],index=['A','B','C','D'])
     demand
```

```
A    30
B    50
C    10
D    20
dtype: int64
```

```
[3]: capacity=pd.Series([40]*3, index=[1,2,3])
     capacity
```

```
1    40
2    40
3    40
dtype: int64
```

Implementing the Formulation Incrementally

```
[4]: # Creating variables
     from gurobipy import Model,GRB
     mod=Model()
     FCs=cost.index
     regions=cost.columns
     x=mod.addVars(FCs,regions,name='x')
     mod.update()
     x
```

```
{(1, 'A'): <gurobi.Var x[1,A]>,
 (1, 'B'): <gurobi.Var x[1,B]>,
 (1, 'C'): <gurobi.Var x[1,C]>,
 (1, 'D'): <gurobi.Var x[1,D]>,
 (2, 'A'): <gurobi.Var x[2,A]>,
 (2, 'B'): <gurobi.Var x[2,B]>,
 (2, 'C'): <gurobi.Var x[2,C]>,
 (2, 'D'): <gurobi.Var x[2,D]>,
 (3, 'A'): <gurobi.Var x[3,A]>,
 (3, 'B'): <gurobi.Var x[3,B]>,
 (3, 'C'): <gurobi.Var x[3,C]>,
 (3, 'D'): <gurobi.Var x[3,D]>}
```

```
[5]: # Objective
     sum(cost.loc[f,r]*x[f,r] for f in FCs for r in regions)
```

```
<gurobi.LinExpr: 20.0 x[1,A] + 18.0 x[1,B] + 21.0 x[1,C] + 8.0 x[1,D] + 8.0 x[2,A] + 23.0 x[2
```

```
[6]: # Capacity 1 Constraint
     sum(x[1,r] for r in regions)<=capacity[1]
```

```
<gurobi.TempConstr: <gurobi.LinExpr: x[1,A] + x[1,B] + x[1,C] + x[1,D]> <= 40>
```

```
[7]: # Demand 1 Constraint
     sum(x[f,'A'] for f in FCs)>=demand['A']
```

```
<gurobi.TempConstr: <gurobi.LinExpr: x[1,A] + x[2,A] + x[3,A]> >= 30>
```

```
[8]: # Full model
     mod=Model()
     x=mod.addVars(FCs,regions,name='x')
     mod.setObjective(sum(cost.loc[f,r]*x[f,r] for f in FCs for r in regions))
     for f in FCs:
         mod.addConstr(sum(x[f,r] for r in regions)<=capacity[f],name=f'Capacity_{f}')
     for r in regions:
         mod.addConstr(sum(x[f,r] for f in FCs)>=demand[r],name=f'Demand_{r}')
     mod.write('10-supplyChain.lp')
     %cat 10-supplyChain.lp
```

```
\ LP format - for model browsing. Use MPS format to capture full model detail.
Minimize
  20 x[1,A] + 18 x[1,B] + 21 x[1,C] + 8 x[1,D] + 8 x[2,A] + 23 x[2,B]
   + 24 x[2,C] + 8 x[2,D] + 25 x[3,A] + 8 x[3,B] + 8 x[3,C] + 19 x[3,D]
Subject To
 Capacity_1: x[1,A] + x[1,B] + x[1,C] + x[1,D] <= 40
 Capacity_2: x[2,A] + x[2,B] + x[2,C] + x[2,D] <= 40
 Capacity_3: x[3,A] + x[3,B] + x[3,C] + x[3,D] <= 40
 Demand_A: x[1,A] + x[2,A] + x[3,A] >= 30
 Demand_B: x[1,B] + x[2,B] + x[3,B] >= 50
 Demand_C: x[1,C] + x[2,C] + x[3,C] >= 10
 Demand_D: x[1,D] + x[2,D] + x[3,D] >= 20
Bounds
End
```

```
[9]: # Numerical Solution
     mod.setParam('OutputFlag',False)
     mod.optimize()
     print('Minimum cost:',mod.objval)
```

```
Minimum cost: 1080.0
```

```
[10]: # Creating the output table
      shipment=pd.DataFrame(index=FCs, columns=regions)
      shipment
```

```
     A    B    C    D
1  NaN  NaN  NaN  NaN
2  NaN  NaN  NaN  NaN
3  NaN  NaN  NaN  NaN
```

```
[11]: # Filling in the output table
      for f in FCs:
          for r in regions:
              shipment.loc[f,r]=x[f,r].x
      shipment
```

```
    A   B   C   D
1   0  20   0  20
2  30   0   0   0
3   0  30  10   0
```

```
[12]: # Final code
      from gurobipy import Model,GRB
      mod=Model()
      FCs=cost.index
      regions=cost.columns
      x=mod.addVars(FCs,regions)
      mod.setObjective(sum(cost.loc[f,r]*x[f,r] for f in FCs for r in regions))
      for f in FCs:
          mod.addConstr(sum(x[f,r] for r in regions)<=capacity[f])
      for r in regions:
          mod.addConstr(sum(x[f,r] for f in FCs)>=demand[r])
      mod.setParam('OutputFlag',False)
      mod.optimize()
      print('Minimum cost:',mod.objval)
      shipment=pd.DataFrame(index=FCs, columns=regions)
      for f in FCs:
          for r in regions:
              shipment.loc[f,r]=x[f,r].x
      shipment
```

```
Minimum cost: 1080.0
```

```
    A   B   C   D
1   0  20   0  20
2  30   0   0   0
3   0  30  10   0
```

Exercise 10.3 Numerical Solution for Assignment of Consultants to Projects

Download the Jupyter notebook attached to the Blackboard link for this exercise and submit it there after completing it. The notebook asks you to incrementally create Gurobi code to implement Q3 of Problem Set 6, following the example given in the lecture.

Recap of Q3 of Problem Set 6: There are two projects and four consultants: Alice, Bob, Charles, and Daphne. Each consultant can be assigned to at most one project, and each project requires at least two consultants. As a manager, you evaluated the relative fitness of the four consultants for each project on a scale of 1 to 5, with 5 being the best fit and 1 being the worst.

	Project 1	Project 2
Alice	5	2
Bob	3	2
Charles	4	5
Daphne	3	1

Furthermore, Alice, Bob and Daphne are senior consultants and each project requires at least one senior on the team.

Formulate a linear optimization problem to maximize the total fitness of the consultants to their assigned project, subject to all the business constraints.

Concrete Formulation

Decision variables: Let x_{ij} denote whether to assign consultant i to project j. (Binary)
Objective:

$$\text{Maximize: } 5x_{A1} + 2x_{A2} + 3x_{B1} + 2x_{B2} + 4x_{C1} + 5x_{C2} + 3x_{D1} + x_{D2}$$

Constraints:

$$\begin{aligned}
\text{(Alice)} \quad & x_{A1} + x_{A2} \leq 1 \\
\text{(Bob)} \quad & x_{B1} + x_{B2} \leq 1 \\
\text{(Charles)} \quad & x_{C1} + x_{C2} \leq 1 \\
\text{(Daphne)} \quad & x_{D1} + x_{D2} \leq 1 \\
\text{(Project 1 Total)} \quad & x_{A1} + x_{B1} + x_{C1} + x_{D1} \geq 2 \\
\text{(Project 2 Total)} \quad & x_{A2} + x_{B2} + x_{C2} + x_{D2} \geq 2 \\
\text{(Project 1 Senior)} \quad & x_{A1} + x_{B1} + x_{D1} \geq 1 \\
\text{(Project 2 Senior)} \quad & x_{A2} + x_{B2} + x_{D2} \geq 1
\end{aligned}$$

Implement the above in Gurobi while obtaining all numbers from the below data structures

Follow the example given in the lecture. See the desired intermediate outputs for every step below. You should write your code in such a way such that if the input data is changed, such as if additional consultants are added, projects are added, or numerical values changed, the code will still work.

Input Data

```
[13]: import pandas as pd
      consultants=['Alice', 'Bob', 'Charles', 'Daphne']
      projects=[1,2]
      fitness=pd.DataFrame([[5,2],[3,2],[4,5],[3,1]],index=consultants,columns=projects)
      senior=['Alice','Bob','Daphne']
      capacity=pd.Series([1,1,1,1],index=consultants)
      demand=pd.Series([2,2],index=projects)
      seniorDemand=pd.Series([1,1],index=projects)

[14]: # Creating variables

{('Alice', 1): <gurobi.Var x[Alice,1]>,
 ('Alice', 2): <gurobi.Var x[Alice,2]>,
 ('Bob', 1): <gurobi.Var x[Bob,1]>,
 ('Bob', 2): <gurobi.Var x[Bob,2]>,
 ('Charles', 1): <gurobi.Var x[Charles,1]>,
 ('Charles', 2): <gurobi.Var x[Charles,2]>,
 ('Daphne', 1): <gurobi.Var x[Daphne,1]>,
 ('Daphne', 2): <gurobi.Var x[Daphne,2]>}
```

```
[15]: # Objective
```

```
<gurobi.LinExpr: 5.0 x[Alice,1] + 2.0 x[Alice,2] + 3.0 x[Bob,1] + 2.0 x[Bob,2] + 4.0 x[Charle
```

```
[16]: # (Alice) constraint
```

```
<gurobi.TempConstr: <gurobi.LinExpr: x[Alice,1] + x[Alice,2]> <= 1>
```

```
[17]: # (Project 1 Total) constraint
```

```
<gurobi.TempConstr: <gurobi.LinExpr: x[Alice,1] + x[Bob,1] + x[Charles,1] + x[Daphne,1]> >= 2
```

```
[18]: # (Project 1 Senior) constraint
```

```
<gurobi.TempConstr: <gurobi.LinExpr: x[Alice,1] + x[Bob,1] + x[Daphne,1]> >= 1>
```

```
[19]: # Full model
```

```
\ LP format - for model browsing. Use MPS format to capture full model detail.
Maximize
  5 x[Alice,1] + 2 x[Alice,2] + 3 x[Bob,1] + 2 x[Bob,2] + 4 x[Charles,1]
   + 5 x[Charles,2] + 3 x[Daphne,1] + x[Daphne,2]
Subject To
 Alice: x[Alice,1] + x[Alice,2] <= 1
 Bob: x[Bob,1] + x[Bob,2] <= 1
 Charles: x[Charles,1] + x[Charles,2] <= 1
 Daphne: x[Daphne,1] + x[Daphne,2] <= 1
 Project_1: x[Alice,1] + x[Bob,1] + x[Charles,1] + x[Daphne,1] >= 2
 Project_2: x[Alice,2] + x[Bob,2] + x[Charles,2] + x[Daphne,2] >= 2
 Project_1_Senior: x[Alice,1] + x[Bob,1] + x[Daphne,1] >= 1
 Project_2_Senior: x[Alice,2] + x[Bob,2] + x[Daphne,2] >= 1
Bounds
Binaries
 x[Alice,1] x[Alice,2] x[Bob,1] x[Bob,2] x[Charles,1] x[Charles,2]
 x[Daphne,1] x[Daphne,2]
End
```

```
[20]: # Numerical Solution
```

```
Maximum total fitness: 15.0
```

```
[21]: # Creating the output table
```

```
           1    2
Alice    NaN  NaN
Bob      NaN  NaN
Charles  NaN  NaN
Daphne   NaN  NaN
```

```
[22]: # Filling in the output table
```

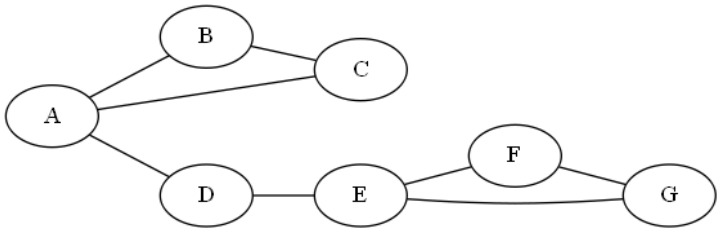
```
         1  2
Alice    1  0
Bob      0  1
Charles  0  1
Daphne   1  0
```

```
[23]: # Final code

Maximum total fitness: 15.0

         1  2
Alice    1  0
Bob      0  1
Charles  0  1
Daphne   1  0
```

Exercise 10.4: Numerical Solution for Project Selection

Download the Jupyter notebook attached to the Blackboard link for this exercise and submit it there after completing it. The notebook asks you to implement the formulation for the Project Selection problem, which is from session 16 and is reproduced below.

Recap of the problem: Ebony is an ambitious master's student who would like to maximize the number of extra-curricular business analytics projects she takes part of this year. However, projects may conflict with one another. The following graph summarizes the conflicts. (For example, project A conflicts with B, C and D, but projects B and D can be done together.)



Beside the conflict above,

- Project A is a prerequisite to project F (meaning that pursuing F requires also pursuing A.)
- Project B is a prerequisite to project G.

Formulate a linear optimization problem to help her decide which projects to pursue.

Concrete Formulation

Decision Variables:

X_i: whether to pursue project i. (Binary)

Objective and Constraints:

$$
\begin{aligned}
\text{Maximize} \quad & X_A + X_B + \cdots + X_G \\
\text{s.t.} \quad & \\
& X_A + X_B \leq 1 \\
& X_B + X_C \leq 1 \\
& X_A + X_C \leq 1 \\
& X_A + X_D \leq 1 \\
& X_D + X_E \leq 1 \\
& X_E + X_F \leq 1 \\
& X_F + X_G \leq 1 \\
& X_E + X_G \leq 1 \\
& X_A \geq X_F \\
& X_B \geq X_G
\end{aligned}
$$

Input data

```
[24]: projects=['A','B','C','D','E','F','G']
      conflicts=[['A','B'],['B','C'],['A','C'],['A','D'],\
                 ['D','E'],['E','F'],['F','G'],['E','G']]
      prereqs=[['A','F'],['B','G']]

[25]: # Examples of looping through the above data structures
      # Standard way
      for pair in conflicts:
          print(f'{pair[0]} and {pair[1]} are in conflict.')
      # Shortcut
      for p1,p2 in prereqs:
          print(f'{p1} is a pre-requisite to {p2}')
```

```
A and B are in conflict.
B and C are in conflict.
A and C are in conflict.
A and D are in conflict.
D and E are in conflict.
E and F are in conflict.
F and G are in conflict.
E and G are in conflict.
A is a pre-requisite to F
B is a pre-requisite to G
```

Write Python code to implement the above using Gurobi. Your code must obtain all data from the above input data structures, such that if new projects are added or the list of conflicts and pre-reqs change, the code will continue to work.

```
[26]: # Final code
```

```
Optimal objective: 3.0
Optimal projects to pursue: B D G
```