Homework 6 Solution

Learning Objectives Assessed:

- Formulating linear programs (LP) to model an optimization problem. (Model)
- Solving LPs of two variables manually by graphing. (Analyze)
- Solving LPs using Gurobi in Python. (Code)
- Interpreting shadow price information. (Analyze)
- Communicate LP formulations using LaTex. (Communicate)

For this homework, you must add your response to this Jupyter notebook and submit it.

Instructions for typesetting LP formulations in Markdown cells

In questions 1,3,4,5, you are asked to "formulate a LP." This means to define your decision variables, then write the objective function and constraints. An example of a LP formulation is given below (this is the LP from the class on Thursday 3/1).

Let decision variables X and Y denote the amount of product X and product Y to use respective. The linear program is

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

Notice that the variables X and Y are in a special font denoting mathematical variables. Moreover, notice that the linear program above is centered and aligned, both at the colons and at the signs ($\leq$ and $\geq$). It is useful to learn to do such formating in Markdown in order to better communicate your LP formulations to others.

Double click this Markdown cell to see the code that created the math. Then execute the cell again to render it.

As you can see, to render a mathematical expression nicely, we enclose it using dollar signs. So the expression X>0, becomes $X > 0$. A single dollar sign is for mathematical expressions inline, and double dollar signs are for a standalone mathematical expressions in its own line (See code by double clicking this cell).

$$X >= 0$$

To make the linear program aligned, we not only use the double dollar signs, but also use the `\begin{aligned} \end{aligned}` commands. (Double click the linear program above to see the code.) Within the alignment block:

- `\text{ }` is for displaying the enclosed string as plain text, without the mathematical rendering.
- `&` is for alignment. The convention is right align before the first `&`, then left align between the first and second `&` of each line, then right align again between the second and third `&` and so on. Hence, to make it right aligned both before and after the colon, we use a double `&&` after the `\text{}`. If this is confusing, you can simply copy this convention for this homework (`&&` after the explanation of constraint, and `&` before the sign).
- `\\` is for creating a new line. Notice that we end the line early using `\\` for "subject to" and "maximize".

- `\le` (less than or equal to) is for $\le$, and `\ge` (greater than or equal to) is for $\ge$. This looks better than $<=$ and $>=$.

This notation is called LaTex, and is the standard for people in technical fields to display math. Make sure that you close any braces you begin and put the alignment characters `&` and newlines `\\` at the right places, or the math may not render. You can see another example in the prompt to Q2 (double click the prompt of Q2 to see the LaTex code).

Q1: (Solving a LP manually by graphing)

Solve DMD Exercise 7.2. The question begins with "The Magnetron Company manufactures and markets microwave ovens." For part a), you should type your formulation into a Markdown cell following the typesetting instructions above. For parts b) and c), you should plot by hand on a scrap piece of paper, and type the answer in a Markdown cell. **You do not have to submit your hand drawing**, but you need to learn to do this for a quiz or an exam.

Solution:

a) Decision variables: Let F and C be the number full-size and compact-size microwave ovens to produce respectively **(in hundreds)**. (Expressing in hundreds is not necessary, but is done here to illustrate the point that you can decide the units in your formulation.) The linear program is:

$$\begin{aligned}
\text{maximize} \quad & 12000F + 13000C \\
\text{subject to:} \quad & \\
\text{General assembly:} \quad & 2F + C \le 5 \\
\text{Electric assembly:} \quad & 2F + 3C \le 8 \\
\text{Full-size demand:} \quad & F \le 2.2 \\
\text{Compact-size demand:} \quad & C \le 1.8 \\
\text{Non-negativity:} \quad & F, C \ge 0
\end{aligned}$$

b) The plot is below. The direction of the objective function is the vector $(F, C) = (12000, 13000)$, which is toward the north east in the graph below (slightly more north than east because $13000 > 12000$). Hence, the optimal solution is the intersection of the general assembly constraint and the electric assembly constraint. (These two are the binding constraints at the optimal solution.) Solving the system of equations:

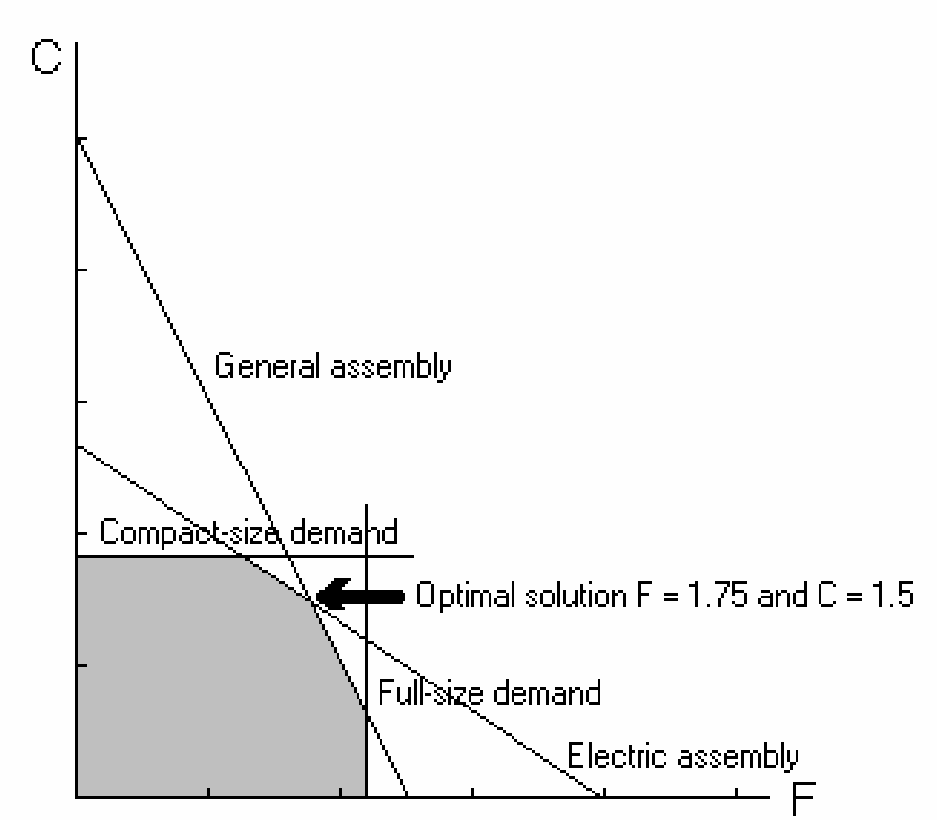
$$\begin{aligned}
2F + C &= 5 \\
2F + 3C &= 8
\end{aligned}$$

We get $(F, C) = (1.75, 1.5)$. The value of the objective function is 40500 dollars.

c) When the number 500 is changed to 510, the new system of equations is (recall that F and C are expressed in hundreds)

$$\begin{aligned}
2F + C &= 5.1 \\
2F + 3C &= 8
\end{aligned}$$

The solutions is $(F, C) = (1.825, 1.45)$. The new value of the objective is $12000 \times 1.825 + 13000 \times 1.45 = 40750$, which is 250 more than the previous objective. Hence, the marginal value of general assembly labor is $250/(510 - 500) = 25$ dollars per hour.



Graphical Solution of LP for Q1

Q2: (Solving a LP using Gurobi)

a) Solve the LP for the NBS supply problem (DMD section 7.1) using Gurobi. Print the optimal solution, as well as the shadow price of each constraint. For your convenience, the LP formulation is reproduced below.

Let decision variables A, B, C, D, E, F, G, and H correspond to the amount of coal to be contracted from Ashley, Bedford, Consol, Dunby, Earlam, Florence, Gaston and Hopt respectively (in mtons).

$$
\begin{aligned}
\text{minimize} \quad & 49.5A + 50B + 61C + 63.5D + 66.5E + 71F + 72.5G + 80H \\
\text{subject to:} \quad & \\
\text{Supply:} \quad & A + B + C + D + E + F + G + H = 1225 \\
\text{Union:} \quad & A + B - C + D - E + F - G - H \geq 0 \\
\text{Truck:} \quad & B + D + E + F \leq 720 \\
\text{Rail:} \quad & A + C + G + H \leq 650 \\
\text{Volatility:} \quad & -4A - 3B - C + D + 2E + 3F + 4G + 6H \geq 0 \\
\text{Acap:} \quad & A \leq 300 \\
\text{Bcap:} \quad & B \leq 600 \\
\text{Ccap:} \quad & C \leq 510 \\
\text{Dcap:} \quad & D \leq 655 \\
\text{Ecap:} \quad & E \leq 575 \\
\text{Fcap:} \quad & F \leq 680 \\
\text{Gcap:} \quad & G \leq 450 \\
\text{Hcap:} \quad & H \leq 490 \\
\text{Non-negativity:} \quad & A, B, C, D, E, F, G, H \geq 0
\end{aligned}
$$

[61]:
```
import gurobipy as grb
m=grb.Model()

# add variables
A=m.addVar(lb=0,name='A')
B=m.addVar(lb=0,name='B')
C=m.addVar(lb=0,name='C')
D=m.addVar(lb=0,name='D')
E=m.addVar(lb=0,name='E')
F=m.addVar(lb=0,name='F')
G=m.addVar(lb=0,name='G')
H=m.addVar(lb=0,name='H')

# add objective
m.setObjective(49.5*A+50*B+61*C+63.5*D+66.5*E+71*F+72.5*G+80*H,sense=grb.GRB.MINIMIZE)

# add constraints
m.addConstr(A+B+C+D+E+F+G+H==1225,name='Supply constraint')
m.addConstr(A+B-C+D-E+F-G-H >= 0, name='Union constraint')
m.addConstr(B+D+E+F <=720, name='Truck constraint')
m.addConstr(A+C+G+H<=650, name='Rail constraint')
m.addConstr(-4*A-3*B-C+D+2*E+3*F+4*G+6*H >=0, name='Volatility constraint')
m.addConstr(A<=300, name='Acap')
m.addConstr(B<=600, name='Bcap')
m.addConstr(C<=510, name='Ccap')
m.addConstr(D<=655, name='Dcap')
m.addConstr(E<=575, name='Ecap')
m.addConstr(F<=680, name='Fcap')
m.addConstr(G<=450, name='Gcap')
m.addConstr(H<=490, name='Hcap')
```

```
        # Solve the model
        m.setParam('OutputFlag',False)
        m.optimize()

        print('Optimal solution')
        for var in m.getVars():
            print('\t{0}={1}'.format(var.VarName,var.x))

        # Print the shadow prices
        print('\nShadow prices')
        for constr in m.getConstrs():
            print('\t{0}: {1}'.format(constr.ConstrName,constr.PI))
```

```
Optimal solution
        A=55.0
        B=600.0
        C=0.0
        D=20.0
        E=100.0
        F=0.0
        G=450.0
        H=0.0

Shadow prices
        Supply constraint: 61.5
        Union constraint: 0.0
        Truck constraint: -1.0
        Rail constraint: 0.0
        Volatility constraint: 3.0
        Acap: 0.0
        Bcap: -1.5
        Ccap: 0.0
        Dcap: 0.0
        Ecap: 0.0
        Fcap: 0.0
        Gcap: -1.0
        Hcap: 0.0
```

b) For each of the constraints except for the volatility constraint, write 1-2 sentence describing what insights can be obtained from the shadow price. You may check your answers using DMD section 7.5.4, but please write the sentence in your own words. For the capacity constraints with zero shadow cost (Acap, Ccap, Dcap, Ecap, Fcap, Hcap), you only need to describe for one constraint, and say that the others are like it.

Example sentence: The shadow price of the union constraint is zero, which means that the union is not a bottleneck in this scenario. (In other words, NBS Corporation would not be able to benefit from the union relaxing its requirement on coal contracts, by allowing for slightly more coal to be contracted from non-union suppliers than from union suppliers.)

Solution: The shadow price of the supply constraint is 61.5, which means that increasing the supply requirement by one mton would increase NBS Corporation's costs by 61.5 thousand dollars.

The shadow price of the union constraint is zero, which means that the union is not a bottleneck for NBS.

The shadow price of the truck constraint is -1, which means that if NBS can ship 1 mton more by truck, then it can decrease its cost by 1 thousand dollars.

The shadow price of the rail constraint is zero, which means that the capacity for rail is not a bottleneck.

The shadow price of the capacitiy constraints for A, C, D, E, F, and H are zero, meaning that these are not bottlenecks for HBS.

The shadow price of the capacity constraint for B is -1.5, meaning that having 1 mton additional capacity from Bedford would decrease the overall cost by 1.5 thousand dollars.

The shadow price of the capacity constraint for G is -1, meaning that having 1 mton additional capacity from Gaston would decrease the overall cost by 1 thousand dollars.

Q3 (Production Planning)

Solve DMD Exercise 7.8. The question begins with "Nature's best Frozen Foods company produces four different mixes of frozen ready-to-eat vegetables.

For part a), you should formulate the linear program following the instructions at the top of this homework (define the decision variables then write the LP nicely in a Markdown cell using LaTex).

a) Let S be the number of bags of Stir Fry mixes to produce, B the number of bags of Barbeque, H the number of bags of Hearty Mushrooms, and V the number of bags of Veggie Crunch.

$$
\begin{aligned}
\text{maximize:} \quad & 0.22S + 0.2B + 0.18H + 0.18V \\
\text{subject to:} \quad & \\
\text{Carrots:} \quad & 2.5S + 2.0B + 2.5V \leq 150000 \\
\text{Mushrooms:} \quad & 3S + 4H \leq 80000 \\
\text{Green peppers:} \quad & 2.5S + 2B + 3H + 2.5V \leq 135000 \\
\text{Broccoli:} \quad & 2S + 3B + 3H + 2.5V \leq 140000 \\
\text{Corn:} \quad & 3B + 2.5V \leq 150000 \\
\text{Non-negativity:} \quad & S, B, H, V \geq 0
\end{aligned}
$$

b) See the code below.

```
[35]: import gurobipy as grb
      m=grb.Model()
      # Define the decision variables
      S=m.addVar(lb=0)
      B=m.addVar(lb=0)
      H=m.addVar(lb=0)
      V=m.addVar(lb=0)

      # Set the objective
      m.setObjective(0.22*S+0.2*B+0.18*H+0.18*V,sense=grb.GRB.MAXIMIZE)

      # Add the constraints
      m.addConstr(2.5*S+2*B+2.5*V<=150000)
      m.addConstr(3*S+4*H <=80000)
```

```
green=m.addConstr(2.5*S+2*B+3*H+2.5*V <= 135000)
m.addConstr(2*S+3*B+3*H+2.5*V <= 140000)
m.addConstr(3*B + 2.5*V <= 150000)

# Optimize
m.setParam( 'OutputFlag', False ) # Not print Gurobi outputs while solving
m.optimize()

# Optimal solution
print('Optimal objective value is',m.ObjVal)
print('Optimal solution is S={0}, B={1}, H={2}, V={3}'.format(S.x,B.x,H.x,V.x))
print('Shadow price of green peppers constraint is',green.PI)
```

```
Optimal objective value is 11813.333333333334
Optimal solution is S=26666.666666666668, B=18333.333333333343, H=0.0, V=12666.666666666655
Shadow price of green peppers constraint is 0.016
```

c) Based on the shadow price, the value of an extra ounce of green peppers is 0.016 dollars.

Q4 (Formulating Constraints)

Solve DMD Exercise 7.12. Before solving using Gurobi, you should formulate the linear program by defining the decision variables and write the objective and constraints nicely using LaTex.

a) Let A, B, C, D, E be the amounts invested in each of the five funds respectively. The linear program is

$$\begin{aligned}
\text{maximize} \quad & 0.045A + 0.0562B + 0.068C + 0.1015D + 0.206E \\
\text{subject to:} \quad & \\
\text{Total investment:} \quad & A + B + C + D + E = 10000 \\
\text{Average risk:} \quad & A + 2B + 2C + 3D + 5E \leq 2.5(A + B + C + D + E) \\
\text{Money market:} \quad & A + B \geq 0.3(A + B + C + D + E) \\
\text{Aggressive growth:} \quad & E \leq 2000 \\
\text{Non-negativity:} \quad & A, B, C, D, E \geq 0
\end{aligned}$$

The solution is as follows.

[62]:
```
import gurobipy as grb
mod=grb.Model()
A=mod.addVar(lb=0,name='Adams')
B=mod.addVar(lb=0,name='Barney')
C=mod.addVar(lb=0,name='Chilton')
D=mod.addVar(lb=0,name='Dunster')
E=mod.addVar(lb=0,name='Excelsior')
mod.setObjective(0.045*A+0.0562*B+0.068*C+0.1015*D+0.206*E,sense=grb.GRB.MAXIMIZE)
mod.addConstr(A+B+C+D+E==10000, 'Total investment')
mod.addConstr(A+2*B+2*C+3*D+5*E <= 2.5*(A+B+C+D+E),'Average risk')
mod.addConstr(A+B >= 0.3*(A+B+C+D+E), 'Money market')
```

```
mod.addConstr(E <= 2000, 'Aggressive growth')

mod.setParam('OutputFlag',False)
mod.optimize()

print('Optimal objective',mod.ObjVal)
print('Optimal solution:')
for var in mod.getVars():
    print('\t',var.VarName,'=',var.x)
```

```
Optimal objective 969.75
Optimal solution:
        Adams = 4500.0
        Barney = 0.0
        Chilton = 0.0
        Dunster = 3500.0
        Excelsior = 2000.0
```

Q5 (Production and Transportation)

Solve DMD Exercise 7.9. The question begins with "Johnson Electric produces small electric motors for four appliance manufacturers in each of the three plants).

When formulating the linear program in part a), **instead of manually defining each variable as in the hint in the book, you should practice using indexing notation**, which also saves you effort on manual input and makes your formulation easily generalizable to larger data sets. You should use the following notation in your formulation:

- $I = \{A, B, C\}$ represents the set of plants;
- $J = \{O, T, H, D\}$ represent the set of manufacturers.
- Let your decision variables be x_{ij}, which is the amount shipped from plant $i \in I$ to manufacturer $j \in J$. (For example x_{AO} is the amount shipped from Arlington to Onyx. $\sum_{j \in J} x_{Aj}$ is the total amount produced from Arlington. $\sum_{i \in I} x_{iO}$ is the total amount shipped to Onyx.)
- For each plant $i \in I$, the production cost is p_i and the monthly capacity is q_i. (For example, $p_A = 17$ and $q_A = 800$.)
- For each manufactuer $j \in J$, the demand of motors per month is given by d_j. (For example, $d_O = 300$, $d_T = 500$.)
- For each plant $i \in I$ and each manufacturer $j \in J$, the transportation cost from plant i to manufactuer j can be indexed by c_{ij}. (For example using the data from the table associated with the question, $c_{AO} = 3$, $c_{BT} = 4$.)

To write a indexed variable in Latex, you can use `c_{ij}` for c_{ij} (make sure to enclose it either in single dollar sign for in-line math or double-dollar sign for stand-alone math). To write summation, you can use `\sum_{i \in I, j \in J} c_{ij}x_{ij}` for

$$\sum_{i \in I, j \in J} c_{ij} x_{ij}.$$

For example you can write the capacity constraint at all three plants using the equation

$$\text{Respecting capacity: } \sum_{j \in J} x_{ij} \leq q_i \text{ for every plant } i \in I.$$

a) Let $I = \{A, B, C\}$ be the set of plants and $J = \{O, T, H, D\}$ be the set of manufacturers. For each $i \in I$ and $j \in J$, let c_{ij} be the unit cost to ship from plant i to manufacturer j. The decision variables x_{ij} represents the amount to ship from plant i to manufactuer j.

$$\begin{aligned}
\text{maximize} \quad & \sum_{i \in I, j \in J} (p_i + c_{ij}) x_{ij} \\
\text{subject to:} & \\
\text{Respecting capacity:} \quad & \sum_{j \in J} x_{ij} \leq q_i \text{ for every plant } i \in I \\
\text{Satisfying demand:} \quad & \sum_{i \in I} x_{ij} \geq d_j \text{ for every manufacturer } j \in J \\
\text{Non-negativity:} \quad & x_{ij} \geq 0 \text{ for every } i \in I, j \in J.
\end{aligned}$$

b) See code below.

```
[53]: import gurobipy as grb
      import pandas as pd
      plants=['Arlington','Binghamton','Canton']
      manufac=['Onyx','Treble','Hilton','Dean']
      prodCosts=pd.Series([17,20,24],index=plants)
      capacities=pd.Series([800,600,700],index=plants)
      demands=pd.Series([300,500,400,600],index=manufac)
      tranCosts=pd.DataFrame([[3,2,5,7],[6,4,8,3],[9,1,5,4]],index=plants,columns=manufac)

      mod=grb.Model()
      x={}
      for i in plants:
          for j in manufac:
              x[i,j]=mod.addVar(lb=0,name='x[{0},{1}]'.format(i,j))
      mod.setObjective(sum((tranCosts.loc[i,j]+prodCosts.loc[i])*x[i,j] for i in plants for j in manufac))
      demConstr={}
      for j in manufac:
          demConstr[j]=mod.addConstr(sum(x[i,j] for i in plants) >= demands.loc[j],name='Demand for '+j)
      capConstr={}
      for i in plants:
          capConstr[i]=mod.addConstr(sum(x[i,j] for j in manufac) <= capacities.loc[i],name='Capacity for '+i)

      mod.setParam( 'OutputFlag', False ) # Not print Gurobi outputs while solving
      mod.optimize()
      mod.write('7.8.lp')
      print ('Optimal objective value is',mod.ObjVal)

      print('Optimal solution')
      for var in mod.getVars():
          print('\t{0}={1}'.format(var.VarName,var.x))

      print('Shadow costs')
      for constr in mod.getConstrs():
          print('\t{0}: {1}'.format(constr.ConstrName,constr.PI))
```

```
Optimal objective value is 40500.0
Optimal solution
        x[Arlington,Onyx]=300.0
        x[Arlington,Treble]=100.0
        x[Arlington,Hilton]=400.0
        x[Arlington,Dean]=0.0
        x[Binghamton,Onyx]=0.0
        x[Binghamton,Treble]=0.0
```

```
        x[Binghamton,Hilton]=0.0
        x[Binghamton,Dean]=600.0
        x[Canton,Onyx]=0.0
        x[Canton,Treble]=400.0
        x[Canton,Hilton]=0.0
        x[Canton,Dean]=0.0
Shadow costs
        Demand for Onyx: 26.0
        Demand for Treble: 25.0
        Demand for Hilton: 28.0
        Demand for Dean: 24.0
        Capacity for Arlington: -6.0
        Capacity for Binghamton: -1.0
        Capacity for Canton: 0.0
```

As can be seen from the solution, the optimal solution is not using all the capacity for Canton (where the production cost is the highest), and this plant is only serving Treble, for which it has a good transportation cost. Furthermore, all of the demand from Dean is being served from Binghamton, which serves no other manufacturer. Arlington, the plant where the production cost is the lowest, is used to satisfy demand from a variety of manufacturers, and is the most valuable resource. Increasing capacity at Arlington by 1 motor/month would reduce total costs by 6 dollars, where as increasing capacity at Binghamton by 1 motor/month would reduce total cost only by 1 dollar. Increasing capacity at Canton is not helpful at this point as we are not even using all of the current capacity.