Session 17: Introduction to Linear Programming (LP) (Solutions)

Example: Production Planning

(DMD Ex. 7.2) The Gemstone Tool Company (GTC) produces wrenches and pliers. Each product is made of steel, and requires using a Molding Machine and an Assembly Machine. The daily availability of each resource, as well as the resources required to produce one units of each tool, are shown below.

	Wrench (1 unit)	Plier (1 unit)	Daily Availability
Steel	1.5 lbs	1.0 lbs	27,000 lbs
Molding Machine	1.0 hours	1.0 hours	21,000 hours
Assembly Machine	0.3 hours	0.5 hours	9,000 hours

There is demand for 16,000 wrenches and 15,000 pliers per day. Each wrench earns a profit of .10 dollars and each plier earns a profit of .13 dollars.

a) How much of each product should GTC produce each day and what is the maximum possible profit?

b) How much additional profit can the company obtain if it had one additional unit of each of the three resources?

Step 1. Identify the decision, objective, and constraints in English

Decision: How many units of wrench and plier to produce each day?

Objective: Maximize profit.

Constraints:

- not using more steel, molding, or assembly than what is available.
- not producing more units than the maximum daily demand.

Step 2. Formulate the optimization as linear expressions of decision variables

Decision Variables:

- W: the number of wrenches to produce.
- P: the number of pliers to produce.

Objective:

$$\text{Maximize: } .1W + .13P$$

Constraints:

$$\begin{aligned}
&\text{Subject to:} \\
\text{(Steel)} \quad & 1.5W + P \leq 27000 \\
\text{(Molding)} \quad & W + P \leq 21000 \\
\text{(Assembly)} \quad & .3W + .5P \leq 9000 \\
\text{(Demand W)} \quad & W \leq 16000 \\
\text{(Demand P)} \quad & P \leq 15000 \\
\text{(Non-negativity)} \quad & W, P \geq 0
\end{aligned}$$

Step 3. Numerically solve using Gurobi

```
[1]: import gurobipy as grb
     mod=grb.Model()
     W=mod.addVar()
     P=mod.addVar()
     mod.setObjective(.1*W+.13*P,sense=grb.GRB.MAXIMIZE)
     steel=mod.addConstr(1.5*W+P <= 27000)
     molding=mod.addConstr(W+P <=21000)
     assembly=mod.addConstr(.3*W+.5*P<=9000)
     mod.addConstr(W<=16000)
     mod.addConstr(P<=15000)
     mod.optimize()
```

```
Academic license - for non-commercial use only
Optimize a model with 5 rows, 2 columns and 8 nonzeros
Coefficient statistics:
  Matrix range     [3e-01, 2e+00]
  Objective range  [1e-01, 1e-01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [9e+03, 3e+04]
Presolve removed 2 rows and 0 columns
Presolve time: 0.01s
Presolved: 3 rows, 2 columns, 6 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    3.5100000e+03   3.375000e+03   0.000000e+00      0s
       3    2.5050000e+03   0.000000e+00   0.000000e+00      0s

Solved in 3 iterations and 0.01 seconds
Optimal objective  2.505000000e+03
```

```
[2]: print('Optimal profit:',mod.objval)
     print('W:',W.x)
     print('P:',P.x)
     print('\nShadow prices:')
     print(f'Steel {steel.pi} \t valid RHS: {steel.sarhslow} to {steel.sarhsup}')
     print(f'Molding {molding.pi:.3f} \t valid RHS: {molding.sarhslow} to {molding.sarhsup}')
     print(f'Assembly {assembly.pi:.3f} \t valid RHS: {assembly.sarhslow} to {assembly.sarhsu
```

```
Optimal profit: 2505.0
W: 7500.0
P: 13500.0

Shadow prices:
Steel 0.0          valid RHS: 24750.0 to 1e+100
Molding 0.055          valid RHS: 20000.0 to 22000.0
Assembly 0.150          valid RHS: 8100.0 to 9300.0
```

Debugging by Outputing Formulation

```
[3]: mod.write('GTC.lp')
     !cat GTC.lp
```

```
\ LP format - for model browsing. Use MPS format to capture full model detail.
Maximize
  0.1 C0 + 0.13 C1
Subject To
 R0: 1.5 C0 + C1 <= 27000
 R1: C0 + C1 <= 21000
 R2: 0.3 C0 + 0.5 C1 <= 9000
 R3: C0 <= 16000
 R4: C1 <= 15000
Bounds
End
```

```
[4]: # Naming variables and constraints
     import gurobipy as grb
     mod=grb.Model()
     W=mod.addVar(name='W')
     P=mod.addVar(name='P')
     mod.setObjective(.1*W+.13*P,sense=grb.GRB.MAXIMIZE)
     steel=mod.addConstr(1.5*W+P <= 27000,name='Steel')
     molding=mod.addConstr(W+P <=21000,name='Molding')
     assembly=mod.addConstr(.3*W+.5*P<=9000,name='Assembly')
     mod.addConstr(W<=16000,name='Demand-W')
     mod.addConstr(P<=15000,name='Demand-P')
     mod.write('GTC.lp')
     !cat GTC.lp
```

```
\ LP format - for model browsing. Use MPS format to capture full model detail.
Maximize
  0.1 W + 0.13 P
Subject To
 Steel: 1.5 W + P <= 27000
 Molding: W + P <= 21000
 Assembly: 0.3 W + 0.5 P <= 9000
 Demand-W: W <= 16000
 Demand-P: P <= 15000
Bounds
End
```

Exercise: Transportation Planning

There are 2 production plants, A and B, with capacities 20 and 15 respectively. There are 3 demand centers, 1, 2, 3, with demand of 10 each. The cost of transporting each unit of good from each plant to each demand center is shown below.

	1	2	3
A	3	7	5
B	5	3	3

a) What is the minimum transportation cost needed to satisfy all demand while respecting plant capacities, and how would you achieve this cost?
b) How would increasing one unit of capacity at each plant affect the optimal cost?
c) How would increasing one unit of demand at each center affect the optimal cost?

Step 1. Identify the decision, objective, and constraints in English

Decision: How many units to transport from each plant to each demand center.
Objective: Minimize transportation cost.
Constraints:

- Not transporting more out of a plant than its capacity.
- Transporting enough to each center to meet its demand.

Step 2. Formulate the optimization as linear expressions of decision variables

Decision variables: Let x_{ij} denote the amount to transport from plant i to demand center j.
Objective and Constraints:

$$\begin{aligned}
\text{Minimize:} \quad & 3x_{A1} + 7x_{A2} + 5x_{A3} + 5x_{B1} + 3x_{B2} + 3x_{B3} \\
\text{Subject to:} \quad & \\
\text{(Capacity A)} \quad & x_{A1} + x_{A2} + x_{A3} \leq 20 \\
\text{(Capacity B)} \quad & x_{B1} + x_{B2} + x_{B3} \leq 15 \\
\text{(Demand 1)} \quad & x_{A1} + x_{B1} \geq 10 \\
\text{(Demand 2)} \quad & x_{A2} + x_{B2} \geq 10 \\
\text{(Demand 3)} \quad & x_{A3} + x_{B3} \geq 10 \\
\text{(Non-negativity)} \quad & x_{ij} \geq 0 \quad \text{for all } i \in \{A, B\}, j \in \{1, 2, 3\}
\end{aligned}$$

Step 3. Numerically solve using Gurobi

[5]:
```
from gurobipy import Model, GRB
mod=Model()
A1=mod.addVar()
A2=mod.addVar()
A3=mod.addVar()
B1=mod.addVar()
B2=mod.addVar()
B3=mod.addVar()
mod.setObjective(3*A1+7*A2+5*A3+5*B1+3*B2+3*B3)
capA=mod.addConstr(A1+A2+A3<=20)
capB=mod.addConstr(B1+B2+B3<=15)
dem1=mod.addConstr(A1+B1>=10)
dem2=mod.addConstr(A2+B2>=10)
dem3=mod.addConstr(A3+B3>=10)
mod.optimize()
```

```
Optimize a model with 5 rows, 6 columns and 12 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [3e+00, 7e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+01, 2e+01]
Presolve time: 0.01s
Presolved: 5 rows, 6 columns, 12 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   3.000000e+01   0.000000e+00      0s
       4    1.0000000e+02   0.000000e+00   0.000000e+00      0s

Solved in 4 iterations and 0.01 seconds
Optimal objective  1.000000000e+02
```

```
[6]: print('A) Minimal cost:',mod.objval)
     print('Optimal transportation plan:')
     print('\tA1:',A1.x)
     print('\tA2:',A2.x)
     print('\tA3:',A3.x)
     print('\tB1:',B1.x)
     print('\tB2:',B2.x)
     print('\tB3:',B3.x)
```

```
A) Minimal cost: 100.0
Optimal transportation plan:
        A1: 10.0
        A2: 0.0
        A3: 5.0
        B1: 0.0
        B2: 10.0
        B3: 5.0
```

```
[7]: print('B) Effect of adding 1 unit of plant capacity')
     print('\t Plant A:',capA.pi,'\tValid RHS:',capA.sarhslow,'to',capA.sarhsup)
     print('\t Plant B:',capB.pi,'\tValid RHS:',capB.sarhslow,'to',capB.sarhsup)
```

```
B) Effect of adding 1 unit of plant capacity
         Plant A: 0.0         Valid RHS: 15.0 to 1e+100
         Plant B: -2.0         Valid RHS: 10.0 to 20.0
```

```
[8]: print('C) Effect of demand increase by 1 unit')
     print('\t Center 1:',dem1.pi,'\tValid RHS:',dem1.sarhslow,'to',dem1.sarhsup)
     print('\t Center 2:',dem2.pi,'\tValid RHS:',dem2.sarhslow,'to',dem2.sarhsup)
     print('\t Center 3:',dem3.pi,'\tValid RHS:',dem3.sarhslow,'to',dem3.sarhsup)
```

```
C) Effect of demand increase by 1 unit
         Center 1: 3.0         Valid RHS: -0.0 to 15.0
         Center 2: 5.0         Valid RHS: 5.0 to 15.0
```

```
        Center 3: 5.0         Valid RHS: 5.0 to 15.0
```

```
[9]: # Alternative using for loops and list comprehension
     from gurobipy import Model, GRB
     import pandas as pd

     # Input data
     plants=['A','B']
     centers=[1,2,3]
     q=pd.Series([20,15],index=plants)
     d=pd.Series([10,10,10],index=centers)
     c=pd.DataFrame([[3,7,5],[5,3,3]],index=plants,columns=centers)

     # Build model
     mod=Model()
     x=mod.addVars(plants,centers)
     cap={}
     for i in plants:
         cap[i]=mod.addConstr(sum(x[i,j] for j in centers)<=q[i])
     dem={}
     for j in centers:
         dem[j]=mod.addConstr(sum(x[i,j] for i in plants)>=d[j])

     mod.setObjective(sum(c.loc[i,j]*x[i,j] for i in plants for j in centers))

     # Solve and print output
     mod.setParam('outputflag',False)
     mod.optimize()

     print('A) Minimal cost:',mod.objval)
     print('Optimal transportation plan:')
     for i in plants:
         for j in centers:
             print(f'\t{i}{j}: {x[i,j].x}')

     print('B) Effect of adding 1 unit of plant capacity')
     for i in plants:
         print (f'\t Plant {i}: {cap[i].pi}')

     print('C) Effect of demand increase by 1 unit')
     for j in centers:
         print (f'\t Center {j}: {dem[j].pi}')
```

```
A) Minimal cost: 100.0
Optimal transportation plan:
        A1: 10.0
        A2: 0.0
        A3: 5.0
        B1: 0.0
        B2: 10.0
        B3: 5.0
```

```
B) Effect of adding 1 unit of plant capacity
        Plant A: 0.0
        Plant B: -2.0
C) Effect of demand increase by 1 unit
        Center 1: 3.0
        Center 2: 5.0
        Center 3: 5.0
```