Gurjus Singh

10/25/2020

MSDS 460 – Decision Analytics

MSDS 460 HW #3

1.

We can define the Decision Variables as X11…X14 for the cities to LA and X21…X24 from the cities to NY and Y1 and Y2 for Binary Variables Seattle and Baltimore.

X11, X21 – would be decision variables for Atlanta

X12, X22 – would be decision variables for Tulsa

X13, X23 – would be decision variables for Seattle

X14, X24 – would be decision variables for Baltimore

With this information our Optimization function is

MIN 8*X11 + 4*X12 + 5*X13 + 4*X14 + 5*X21 + 7*X22 + 6*X23 + 6*X24

Our constraints are

Y1 + Y2 = 1

X11 + X12 <= 600

X21 + X22 <= 900

X13 + X23 <= 500*Y1

X14 +X24 <= 500*Y2

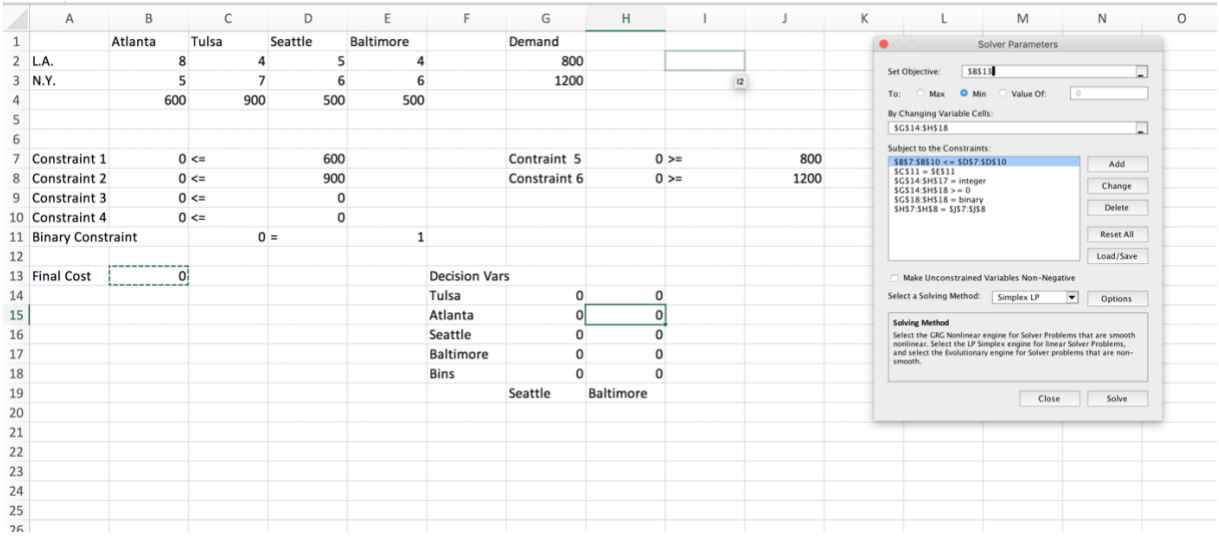
X11 + X12 + X13 + X14 >= 800

X21 + X22 + X23 + X24 >= 1200

All our decision variables need to be integers and above 0.

X11..X24 >= 0; Y1,Y2 >= 0

With this information the problem is set up in excel like this:



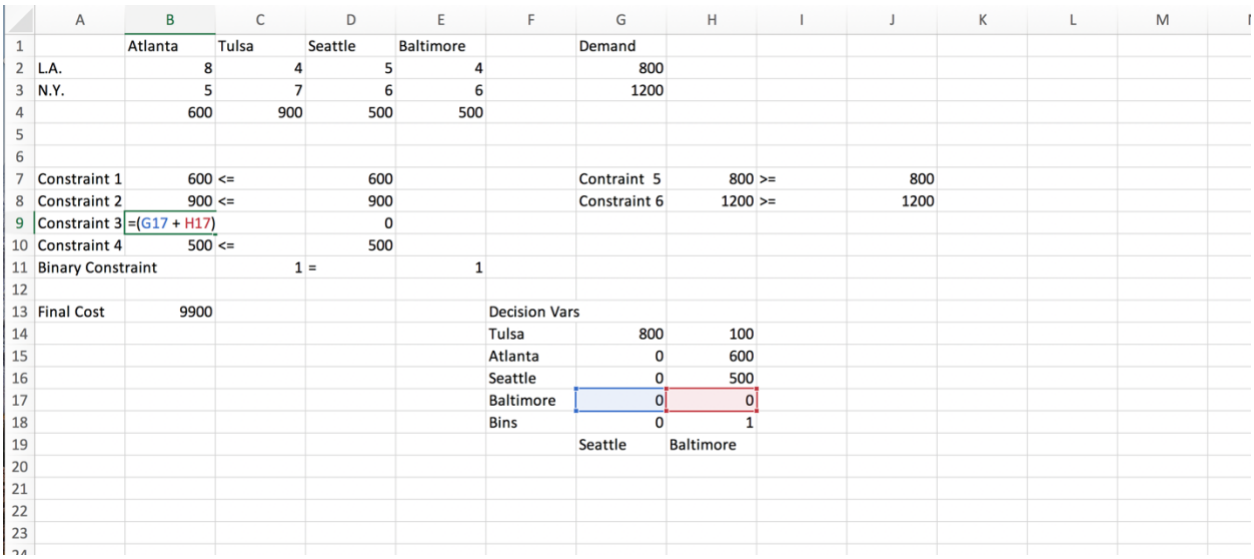
	A	B	C	D	E	F	G	H	I	J
1		Atlanta	Tulsa	Seattle	Baltimore		Demand			
2	L.A.	8	4	5	4		800			
3	N.Y.	5	7	6	6		1200			
4		600	900	500	500					
5										
6										
7	Constraint 1	0	<=	600			Contraint 5	0	>=	800
8	Constraint 2	0	<=	900			Constraint 6	0	>=	1200
9	Constraint 3	0	<=	0						
10	Constraint 4	0	<=	0						
11	Binary Constraint		0	=	1					
12										
13	Final Cost	0				Decision Vars				
14						Tulsa	0	0		
15						Atlanta	0	0		
16						Seattle	0	0		
17						Baltimore	0	0		
18						Bins	0	0		
19							Seattle	Baltimore		
20										
21										
22										
23										
24										
25										

Here are the behind the cell formulas:

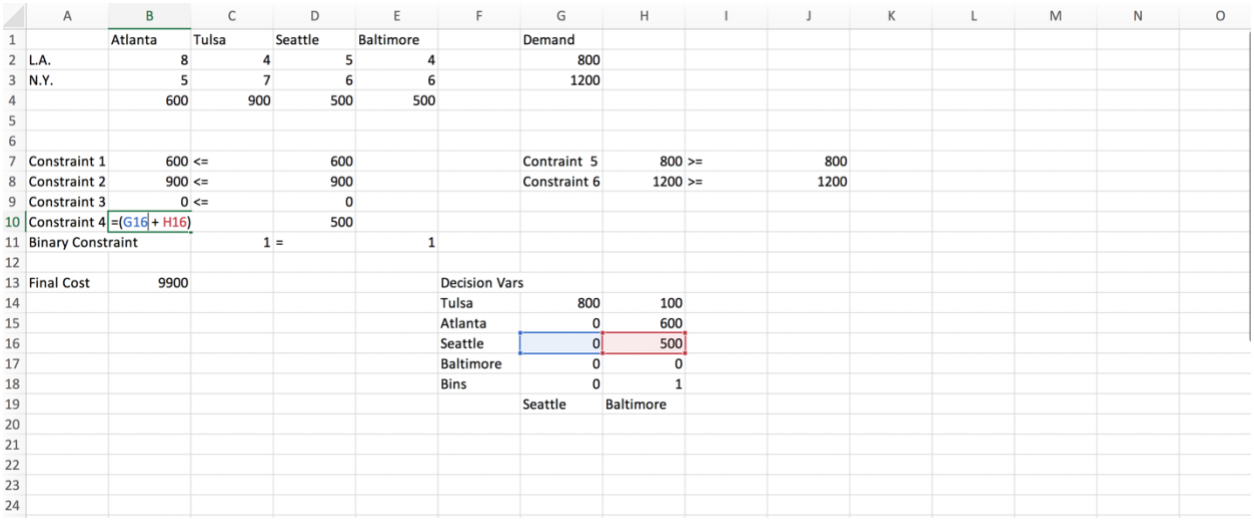
	Atlanta	Tulsa	Seattle	Baltimore		Demand			
L.A.	8	4	5	4		800			
N.Y.	5	7	6	6		1200			
	600	900	500	500					
Constraint 1	600	<=	600			Contraint 5	800	>=	800
Constraint 2	900	<=	900			Constraint 6	1200	>=	1200
Constraint 3	0	<=	0						
Constraint 4	500	<=	500						
Binary Constraint		1	=	1					
Final Cost	=B2*G15 + C2*G14 + (D2*G16)+ (E2*G17) + B3*H15 + C3*H14 + (D3*H16) + (E3*H17)								
					Tulsa	800	100		
					Atlanta	0	600		
					Seattle	0	500		
					Baltimore	0	0		
					Bins	0	1		
						Seattle	Baltimore		

	Atlanta	Tulsa	Seattle	Baltimore		Demand				
L.A.	8	4	5	4		800				
N.Y.	5	7	6	6		1200				
	600	900	500	500						
Constraint 1	=G15 +H15	<=	600			Contraint 5	800	>=	800	
Constraint 2	900	<=	900			Constraint 6	1200	>=	1200	
Constraint 3	0	<=	0							
Constraint 4	500	<=	500							
Binary Constraint		1	=	1						
Final Cost	9900				Decision Vars					
					Tulsa	800	100			
					Atlanta	0	600			
					Seattle	0	500			
					Baltimore	0	0			
					Bins	0	1			
						Seattle	Baltimore			

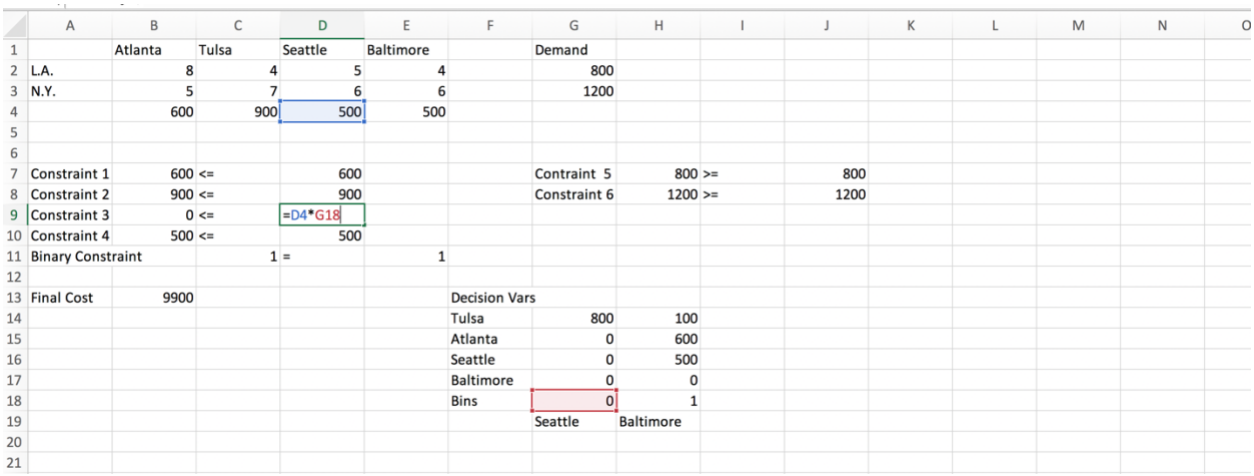
	Atlanta	Tulsa	Seattle	Baltimore		Demand				
L.A.	8	4	5	4		800				
N.Y.	5	7	6	6		1200				
	600	900	500	500						
Constraint 1	600	<=	600			Contraint 5	800	>=	800	
Constraint 2	=G14+H14	<=	900			Constraint 6	1200	>=	1200	
Constraint 3	0	<=	0							
Constraint 4	500	<=	500							
Binary Constraint		1	=	1						
Final Cost	9900				Decision Vars					
					Tulsa	800	100			
					Atlanta	0	600			
					Seattle	0	500			
					Baltimore	0	0			
					Bins	0	1			
						Seattle	Baltimore			



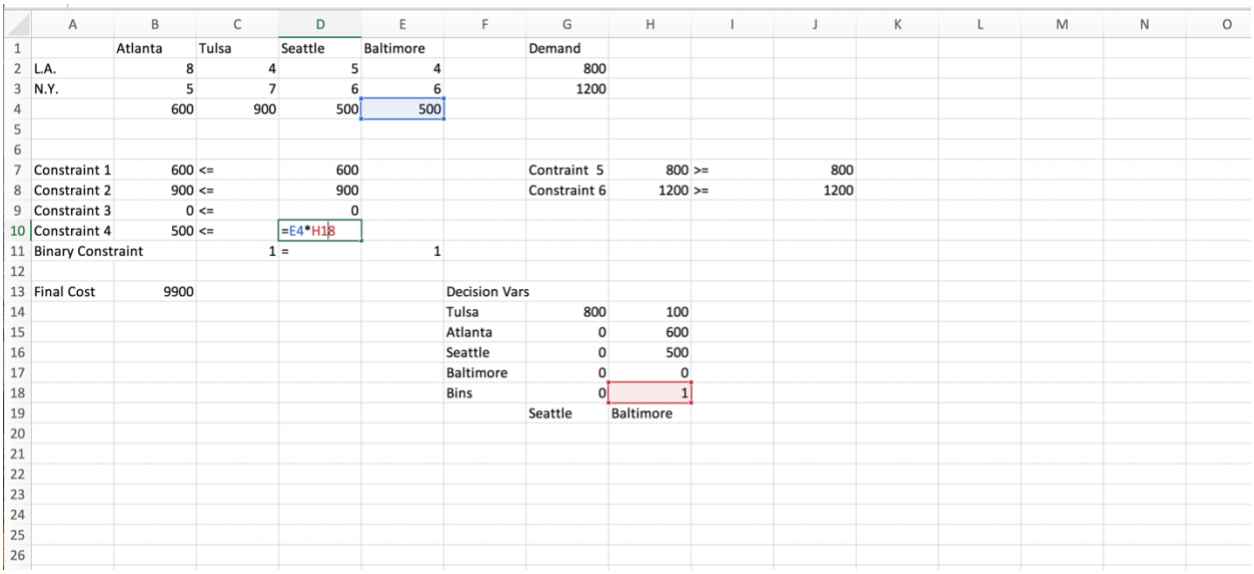
	A	B	C	D	E	F	G	H	I	J
1		Atlanta	Tulsa	Seattle	Baltimore		Demand			
2	L.A.	8	4	5	4		800			
3	N.Y.	5	7	6	6		1200			
4		600	900	500	500					
5										
6										
7	Constraint 1	600	<=	600			Contraint 5	800	>=	800
8	Constraint 2	900	<=	900			Constraint 6	1200	>=	1200
9	Constraint 3	=(G17 + H17)		0						
10	Constraint 4	500	<=	500						
11	Binary Constraint		1	=	1					
12										
13	Final Cost	9900				Decision Vars				
14						Tulsa	800	100		
15						Atlanta	0	600		
16						Seattle	0	500		
17						Baltimore	0	0		
18						Bins	0	1		
19							Seattle	Baltimore		



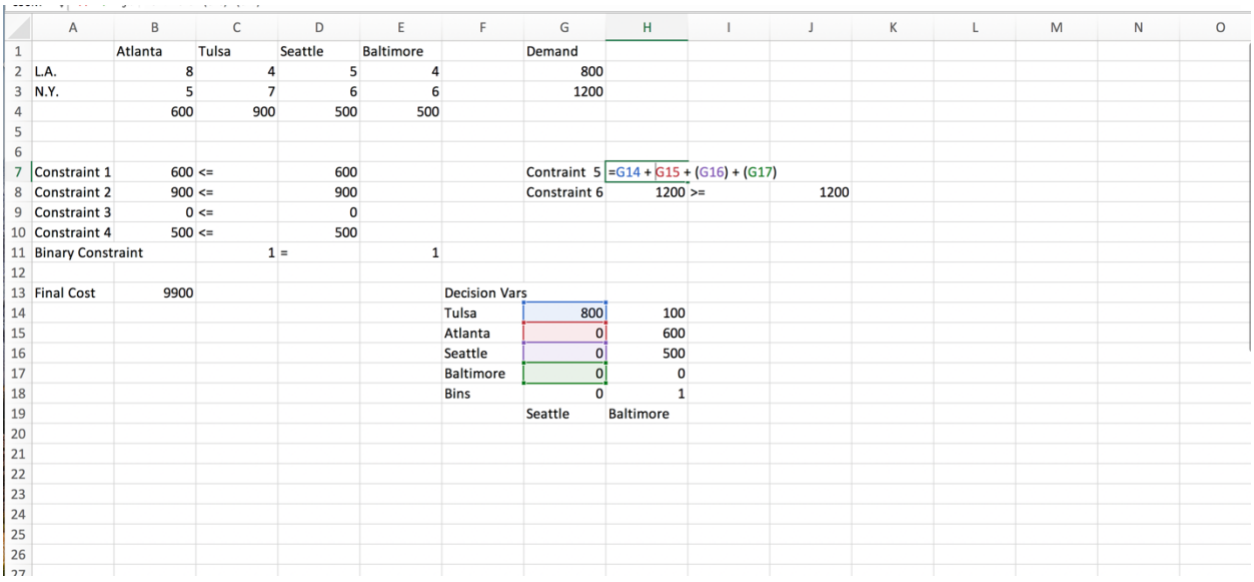
	A	B	C	D	E	F	G	H	I	J
1		Atlanta	Tulsa	Seattle	Baltimore		Demand			
2	L.A.	8	4	5	4		800			
3	N.Y.	5	7	6	6		1200			
4		600	900	500	500					
5										
6										
7	Constraint 1	600	<=	600			Contraint 5	800	>=	800
8	Constraint 2	900	<=	900			Constraint 6	1200	>=	1200
9	Constraint 3	0	<=	0						
10	Constraint 4	=(G16 + H16)		500						
11	Binary Constraint		1	=	1					
12										
13	Final Cost	9900				Decision Vars				
14						Tulsa	800	100		
15						Atlanta	0	600		
16						Seattle	0	500		
17						Baltimore	0	0		
18						Bins	0	1		
19							Seattle	Baltimore		



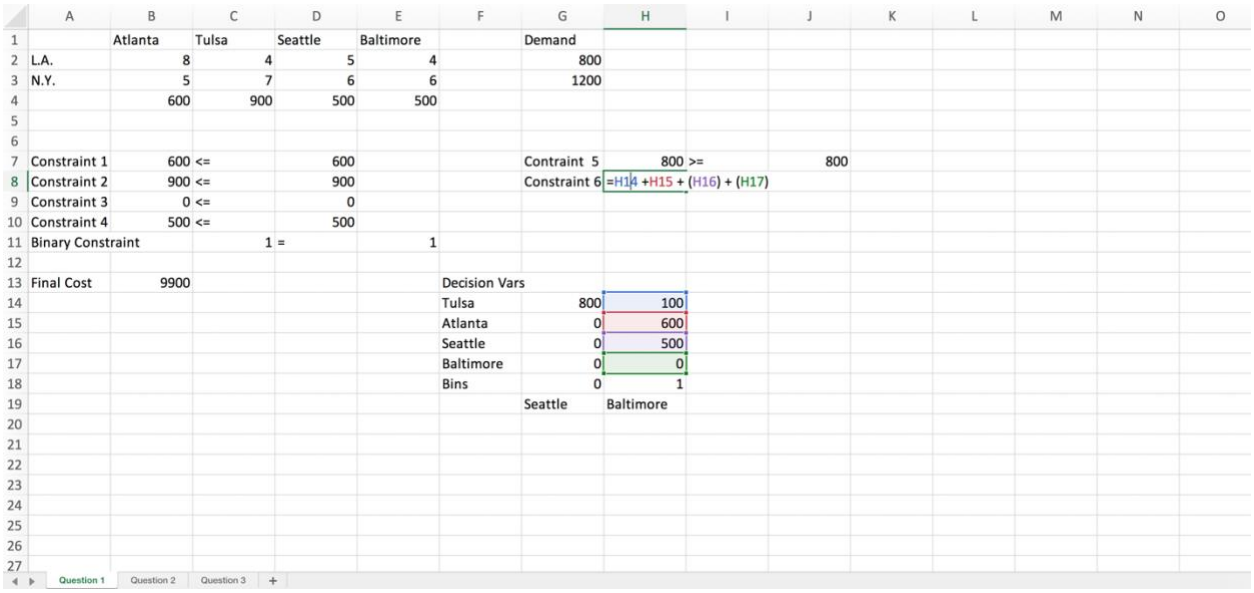
	A	B	C	D	E	F	G	H	I	J
1		Atlanta	Tulsa	Seattle	Baltimore		Demand			
2	L.A.	8	4	5	4		800			
3	N.Y.	5	7	6	6		1200			
4		600	900	500	500					
5										
6										
7	Constraint 1	600	<=	600			Contraint 5	800	>=	800
8	Constraint 2	900	<=	900			Constraint 6	1200	>=	1200
9	Constraint 3	0	<=	=D4*G18						
10	Constraint 4	500	<=	500						
11	Binary Constraint		1	=	1					
12										
13	Final Cost	9900				Decision Vars				
14						Tulsa	800	100		
15						Atlanta	0	600		
16						Seattle	0	500		
17						Baltimore	0	0		
18						Bins	0	1		
19							Seattle	Baltimore		



	A	B	C	D	E	F	G	H	I	J
1		Atlanta	Tulsa	Seattle	Baltimore		Demand			
2	L.A.	8	4	5	4		800			
3	N.Y.	5	7	6	6		1200			
4		600	900	500	500					
5										
6										
7	Constraint 1	600	<=	600			Contraint 5	800	>=	800
8	Constraint 2	900	<=	900			Constraint 6	1200	>=	1200
9	Constraint 3	0	<=	0						
10	Constraint 4	500	<=	=E4*H18						
11	Binary Constraint			1 =	1					
12										
13	Final Cost	9900				Decision Vars				
14						Tulsa	800	100		
15						Atlanta	0	600		
16						Seattle	0	500		
17						Baltimore	0	0		
18						Bins	0	1		
19							Seattle	Baltimore		
20										
21										
22										
23										
24										
25										
26										



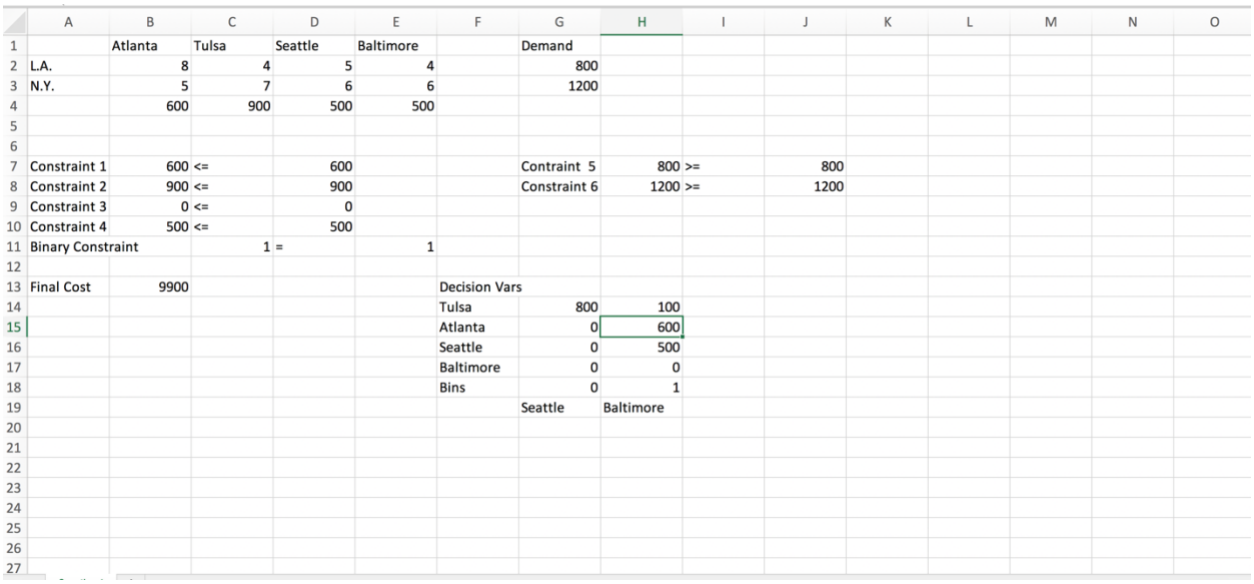
	A	B	C	D	E	F	G	H	I	J
1		Atlanta	Tulsa	Seattle	Baltimore		Demand			
2	L.A.	8	4	5	4		800			
3	N.Y.	5	7	6	6		1200			
4		600	900	500	500					
5										
6										
7	Constraint 1	600	<=	600			Contraint 5	=G14 + G15 + (G16) + (G17)		
8	Constraint 2	900	<=	900			Constraint 6	1200	>=	1200
9	Constraint 3	0	<=	0						
10	Constraint 4	500	<=	500						
11	Binary Constraint			1 =	1					
12										
13	Final Cost	9900				Decision Vars				
14						Tulsa	800	100		
15						Atlanta	0	600		
16						Seattle	0	500		
17						Baltimore	0	0		
18						Bins	0	1		
19							Seattle	Baltimore		
20										
21										
22										
23										
24										
25										
26										



	A	B	C	D	E	F	G	H	I	J
1		Atlanta	Tulsa	Seattle	Baltimore		Demand			
2	L.A.	8	4	5	4		800			
3	N.Y.	5	7	6	6		1200			
4		600	900	500	500					
5										
6										
7	Constraint 1	600	<=	600			Contraint 5	800	>=	800
8	Constraint 2	900	<=	900			Constraint 6	=H14 +H15 + (H16) + (H17)		
9	Constraint 3	0	<=	0						
10	Constraint 4	500	<=	500						
11	Binary Constraint		1	=	1					
12										
13	Final Cost	9900				Decision Vars				
14						Tulsa	800	100		
15						Atlanta	0	600		
16						Seattle	0	500		
17						Baltimore	0	0		
18						Bins	0	1		
19							Seattle	Baltimore		

Question 1 | Question 2 | Question 3 | +

Our solution is then...



	A	B	C	D	E	F	G	H	I	J
1		Atlanta	Tulsa	Seattle	Baltimore		Demand			
2	L.A.	8	4	5	4		800			
3	N.Y.	5	7	6	6		1200			
4		600	900	500	500					
5										
6										
7	Constraint 1	600	<=	600			Contraint 5	800	>=	800
8	Constraint 2	900	<=	900			Constraint 6	1200	>=	1200
9	Constraint 3	0	<=	0						
10	Constraint 4	500	<=	500						
11	Binary Constraint		1	=	1					
12										
13	Final Cost	9900				Decision Vars				
14						Tulsa	800	100		
15						Atlanta	0	600		
16						Seattle	0	500		
17						Baltimore	0	0		
18						Bins	0	1		
19							Seattle	Baltimore		

Optimal cost is 9,900; the Variables are X12 = 800, X22 = 100, X21 = 600, X23 = 500 and Y2 =

1. So we set up a new plant in Baltimore.

I have also set up the same problem in Gurobi shown below:

```
m = Model()
```

```
seabin = m.addVar(vtype=GRB.BINARY, name="seabin")
```

```
baltbin =  m.addVar(vtype=GRB.BINARY, name="baltbin")
seattle = m.addVar(vtype=GRB.INTEGER, name="seattle")
baltimore = m.addVar(vtype=GRB.INTEGER, name="baltimore")
atlanta = m.addVar(vtype=GRB.INTEGER, name="atlanta")
tulsa = m.addVar(vtype=GRB.INTEGER, name="tulsa")
seattle2 = m.addVar(vtype=GRB.INTEGER, name="seattle2")
baltimore2 = m.addVar(vtype=GRB.INTEGER, name="baltimore2")
atlanta2 = m.addVar(vtype=GRB.INTEGER, name="atlanta2")
tulsa2 = m.addVar(vtype=GRB.INTEGER, name="tulsa2")
```

Variable are defined above ^

```
m.addConstr(tulsa + tulsa2 <= 900)
m.addConstr(atlanta + atlanta2 <= 600)
m.addConstr(seattle + seattle2 <= 500*seabin)
m.addConstr(baltimore + baltimore2 <= 500*baltbin)
m.addConstr(atlanta + baltimore + tulsa + seattle >= 800)
m.addConstr(atlanta2 + baltimore2 + tulsa2 + seattle2 >= 1200)
m.addConstr(seabin + baltbin == 1)
m.addConstr(tulsa >= 0)
m.addConstr(tulsa2 >= 0)
m.addConstr(atlanta >= 0)
m.addConstr(atlanta2 >= 0)
m.addConstr(seattle >= 0)
m.addConstr(seattle2 >= 0)
m.addConstr(baltimore >= 0)
m.addConstr(baltimore2 >= 0)
```

Constraints are defined above ^

Here is the objective function

```
cost = 8*atlanta + 4*tulsa + 5*seattle + 4*baltimore + 5*atlanta2 + 7*tulsa2 + 6*seattle2 + 6*baltimore2
m.setObjective(cost,GRB.MINIMIZE)
```

Here is our optimal solution with variables and their values:

```
m.objVal
```

```
9900.0
```

```
m.getVars()
```

```
[<gurobi.Var seabin (value -0.0)>,
 <gurobi.Var baltbin (value 1.0)>,
 <gurobi.Var seattle (value -0.0)>,
 <gurobi.Var baltimore (value -0.0)>,
 <gurobi.Var atlanta (value -0.0)>,
 <gurobi.Var tulsa (value 800.0)>,
 <gurobi.Var seattle2 (value -0.0)>,
 <gurobi.Var baltimore2 (value 500.0)>,
 <gurobi.Var atlanta2 (value 600.0)>,
 <gurobi.Var tulsa2 (value 100.0)>]
```

2. We can define our decision variables for each meat as X1…X4.

So X1 = Beef

X2 = Pork

X3 = Chicken

X4 = Turkey

From this our constraints are

X1 + X2 + X3 + X4 = 0.125 constraint for 2oz, I converted to pounds which is why I show 0.125

640*X1 + 1055*X2 + 740*X3 + 528*X4 <= 100 CAL – constraint for at most 100 calories

32.5*X1 + 54*X2 + 25.6*X3 + 6.4*X4 <= 6 grams of fat – constraint for nothing more than 6 grams of fat

210*X1 + 205*X2 + 220*X3 + 172*X4 <= 27 grams of cholesterol– constraint for nothing more than 27 grams of cholesterol

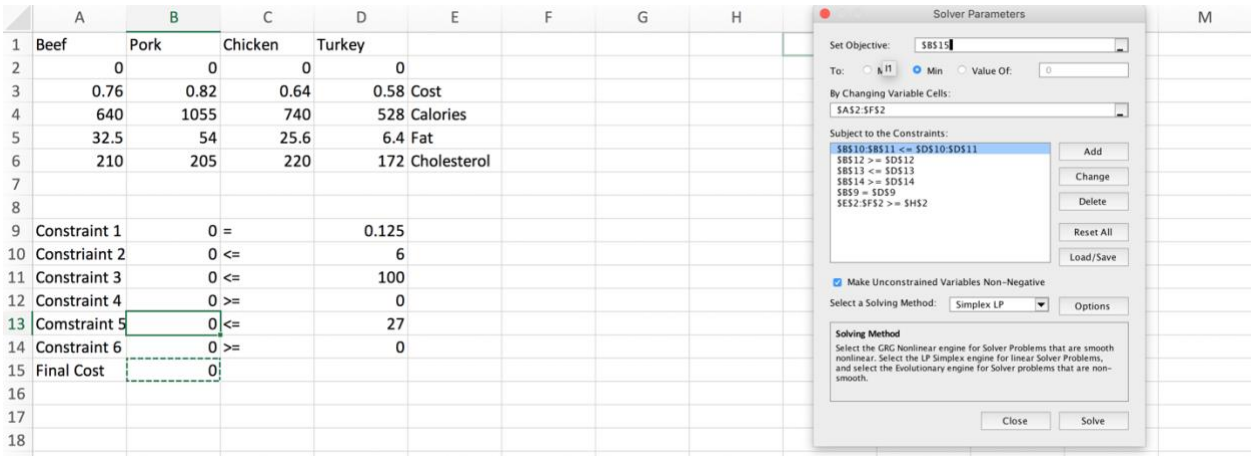
Optimization Function is

MIN 0.76*X1 + 0.82*X2 + 0.64*X3 + 0.58*X4

We need to make sure we satisfy nonnegativity so,

X1…X4 >= 0

Now we can set this up in Excel:



	A	B	C	D	E
1	Beef	Pork	Chicken	Turkey	
2	0	0	0	0	
3	0.76	0.82	0.64	0.58	Cost
4	640	1055	740	528	Calories
5	32.5	54	25.6	6.4	Fat
6	210	205	220	172	Cholesterol
7					
8					
9	Constraint 1	0	=	0.125	
10	Constriaint 2	0	<=	6	
11	Constraint 3	0	<=	100	
12	Constraint 4	0	>=	0	
13	Comstraint 5	0	<=	27	
14	Constraint 6	0	>=	0	
15	Final Cost	0			
16					
17					
18					

Here are the behind the cell formulas:

	A	B	C	D	E	F	G
1	Beef	Pork	Chicken	Turkey			
2	0.03125	0.03125	0	0.0625	0	0	
3	0.76	0.82	0.64	0.58	Cost		
4	640	1055	740	528	Calories		
5	32.5	54	25.6	6.4	Fat		
6	210	205	220	172	Cholesterol		
7							
8							
9	Constraint 1	=A2 + B2+ C2 +D2		0.125			
10	Constriaint 2	3.103125	<=	6			
11	Constraint 3	85.96875	<=	100			
12	Constraint 4	0.03125	>=	0.03125			
13	Comstraint 5	23.71875	<=	27			
14	Constraint 6	0.03125	>=	0.03125			
15	Final Cost	0.085625					
16							
17							

	A	B	C	D	E	F	G
1	Beef	Pork	Chicken	Turkey			
2	0.03125	0.03125	0	0.0625	0	0	
3	0.76	0.82	0.64	0.58	Cost		
4	640	1055	740	528	Calories		
5	32.5	54	25.6	6.4	Fat		
6	210	205	220	172	Cholesterol		
7							
8							
9	Constraint 1	0.125	=	0.125			
10	Constriaint 2	=A2*A5 + B2*B5 + C2*C5 +D2*D5					
11	Constraint 3	85.96875	<=	100			
12	Constraint 4	0.03125	>=	0.03125			
13	Comstraint 5	23.71875	<=	27			
14	Constraint 6	0.03125	>=	0.03125			
15	Final Cost	0.085625					
16							
17							
18							

	A	B	C	D	E	F	G
1	Beef	Pork	Chicken	Turkey			
2	0.03125	0.03125	0	0.0625	0	0	
3	0.76	0.82	0.64	0.58	Cost		
4	640	1055	740	528	Calories		
5	32.5	54	25.6	6.4	Fat		
6	210	205	220	172	Cholesterol		
7							
8							
9	Constraint 1	0.125	=	0.125			
10	Constriaint 2	3.103125	<=	6			
11	Constraint 3	=A2*A4 + B2*B4 + C2*C4 + D2*D4					
12	Constraint 4	0.03125	>=	0.03125			
13	Comstraint 5	23.71875	<=	27			
14	Constraint 6	0.03125	>=	0.03125			
15	Final Cost	0.085625					
16							
17							

	A	B	C	D	E	F	G
1	Beef	Pork	Chicken	Turkey			
2	0.03125	0.03125	0	0.0625	0	0	
3	0.76	0.82	0.64	0.58	Cost		
4	640	1055	740	528	Calories		
5	32.5	54	25.6	6.4	Fat		
6	210	205	220	172	Cholesterol		
7							
8							
9	Constraint 1	0.125	=	0.125			
10	Constriaint 2	3.103125	<=	6			
11	Constraint 3	85.96875	<=	100			
12	Constraint 4	=A2	>=	0.03125			
13	Comstraint 5	23.71875	<=	27			
14	Constraint 6	0.03125	>=	0.03125			
15	Final Cost	0.085625					
16							
17							
18							

	A	B	C	D	E	F	G	H
1	Beef	Pork	Chicken	Turkey				
2	0.03125	0.03125	0	0.0625	0	0		
3	0.76	0.82	0.64	0.58	Cost			
4	640	1055	740	528	Calories			
5	32.5	54	25.6	6.4	Fat			
6	210	205	220	172	Cholesterol			
7								
8								
9	Constraint 1	0.125	=	0.125				
10	Constriaint 2	3.103125	<=	6				
11	Constraint 3	85.96875	<=	100				
12	Constraint 4	0.03125	>=	0.03125				
13	Comstraint 5	=A2*A6 + B2*B6 + C2*C6 + D2*D6						
14	Constraint 6	0.03125	>=	0.03125				
15	Final Cost	0.085625						
16								
17								
18								

	A	B	C	D	E	F	G	H
1	Beef	Pork	Chicken	Turkey				
2	0.03125	0.03125	0	0.0625	0	0		
3	0.76	0.82	0.64	0.58	Cost			
4	640	1055	740	528	Calories			
5	32.5	54	25.6	6.4	Fat			
6	210	205	220	172	Cholesterol			
7								
8								
9	Constraint 1	0.125	=	0.125				
10	Constriaint 2	3.103125	<=	6				
11	Constraint 3	85.96875	<=	100				
12	Constraint 4	0.03125	>=	0.03125				
13	Comstraint 5	23.71875	<=	27				
14	Constraint 6	=B2	>=	0.03125				
15	Final Cost	0.085625						
16								
17								
18								

1	Beef	Pork	Chicken	Turkey			
2	0.03125	0.03125	0	0.0625	0	0	
3	0.76	0.82	0.64	0.58	Cost		
4	640	1055	740	528	Calories		
5	32.5	54	25.6	6.4	Fat		
6	210	205	220	172	Cholesterol		
7							
8							
9	Constraint 1	0.125	=	0.125			
10	Constriaint 2	3.103125	<=	6			
11	Constraint 3	85.96875	<=	100			
12	Constraint 4	0.03125	>=	0.03125			
13	Comstraint 5	23.71875	<=	27			
14	Constraint 6	0.03125	>=	0.03125			
15	Final Cost	=A2*A3 + B2*B3 + C2*C3 + D2*D3					

When we solve the problem, our optimal solution is 0.085625 and our decision values are X1 = 0.03125, X2 = 0.03125, X3 = 0, X4 = 0.0625 shown below:

	A	B	C	D	E	F	G	H	I	J	K
1	Beef	Pork	Chicken	Turkey							
2	0.03125	0.03125	0	0.0625	0	0					
3	0.76	0.82	0.64	0.58	Cost						
4	640	1055	740	528	Calories						
5	32.5	54	25.6	6.4	Fat						
6	210	205	220	172	Cholesterol						
7											
8											
9	Constraint 1	0.125	=	0.125							
10	Constriaint 2	3.103125	<=	6							
11	Constraint 3	85.96875	<=	100							
12	Constraint 4	0.03125	>=	0.03125							
13	Comstraint 5	23.71875	<=	27							
14	Constraint 6	0.03125	>=	0.03125							
15	Final Cost	0.085625									

Now we can solve this in Gurobi, here is the setup and the answer:

```
In [200]: q2 = Model()
          beef = q2.addVar()
          chicken = q2.addVar()
          turkey = q2.addVar()
          pork = q2.addVar()

In [201]: q2.update()

In [205]: q2.addConstr(beef + chicken + turkey + pork == 0.125)
          q2.addConstr(640*beef + 740*chicken + 528*turkey + 1055*pork <= 100)
          q2.addConstr(32.5*beef + 25.6*chicken + 6.4*turkey + 54*pork <= 6)
          q2.addConstr(210*beef + 220*chicken + 172*turkey + 205*pork <= 27)
          q2.addConstr(beef  >= 0.25*(beef + chicken + turkey + pork))
          q2.addConstr(pork >= 0.25*(beef + chicken + turkey + pork))
Out[205]: <gurobi.Constr *Awaiting Model Update*>

In [206]: q2.update()
          costmeat = 0.76*beef + 0.82*pork + 0.64*chicken + 0.58*turkey
          q2.setObjective(costmeat,GRB.MINIMIZE)
```

Decision variables, constraints, optimization function all defined above ^

```
In [208]: q2.optimize()

Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (mac64)
Optimize a model with 7 rows, 4 columns and 28 nonzeros
Model fingerprint: 0x5f28d6f3
Coefficient statistics:
  Matrix range     [2e-01, 1e+03]
  Objective range  [4e-01, 8e-01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e-01, 1e+02]
Presolve removed 1 rows and 0 columns
Presolve time: 0.01s
Presolved: 6 rows, 4 columns, 24 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    7.2500000e-02   3.000000e+00   0.000000e+00      0s
       2    8.5625000e-02   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.01 seconds
Optimal objective  8.562500000e-02

In [209]: q2.getVars()
Out[209]: [<gurobi.Var C0 (value 0.031250000000000001)>,
           <gurobi.Var C1 (value 0.0)>,
           <gurobi.Var C2 (value 0.0625)>,
           <gurobi.Var C3 (value 0.03125)>]

In [211]: q2.printStats()

Statistics for model Unnamed :
  Linear constraint matrix     : 7 Constrs, 4 Vars, 28 NZs
  Matrix coefficient range     : [ 0.25, 1055 ]
  Objective coefficient range : [ 0.58, 0.82 ]
  Variable bound range         : [ 0, 0 ]
  RHS coefficient range        : [ 0.125, 100 ]
```

As you can see we get the same values that we get in Excel, but in scientific notation.

3.

5 X 5 SOLUTION:

For this problem, our goal is to minimize total set up time, by optimization. This is to eliminate mass casualties such as in war.

In order to perform this problem, one needs to duplicate the setup time matrix of 5x5 first. This is below in excel:

0	20	15	8	6	5
15	0	18	9	28	4
24	23	0	13	13	3
15	27	8	0	14	2
8	17	24	15	0	1
1	2	3	4	5	

Then we create a binary decision variable matrix of the same size:

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0

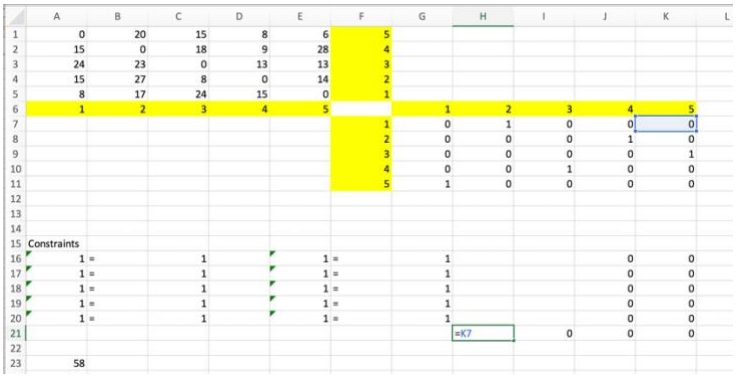
In order to find minimum cost, we take the two matrixes and multiply and add them which is called SUMPRODUCT() in Excel.

We also add constraints so that each row in our binary matrix is equal to 1, and each column is = 1. This is so we only get one selection for each column.

	Constraints						
	0	=	1		0	=	1
	0	=	1		0	=	1
	0	=	1		0	=	1
	0	=	1		0	=	1
	0	=	1		0	=	1

I noticed when I ran the solution the first time that I got a subroute at 1,5,1, so I made a new constraint which is cell 1,5 should be = 0. This got rid of these constraints and I got an optimal solution of 58.

The constraint was added to Excel shown below:



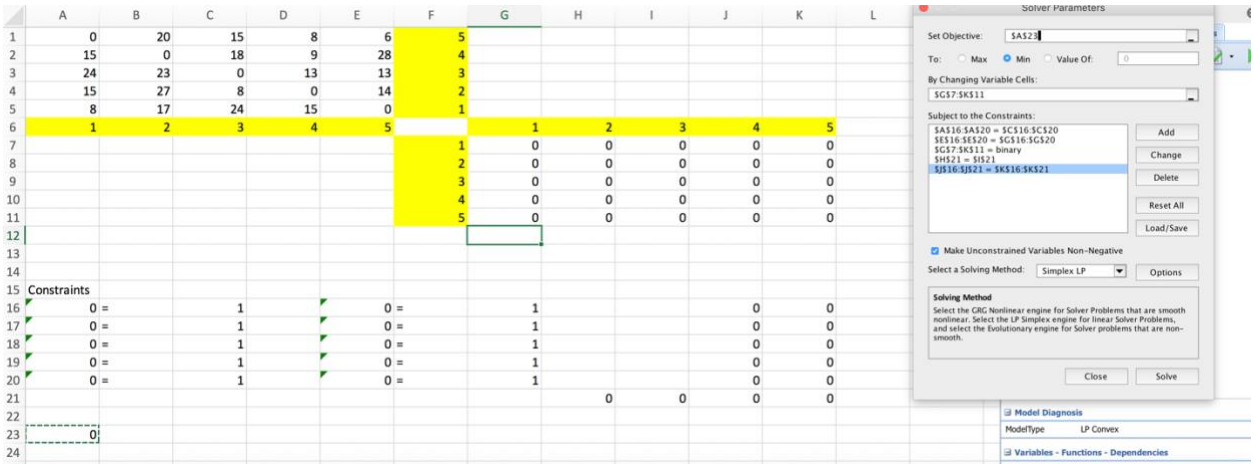
	A	B	C	D	E	F	G	H	I	J	K	L
1	0	20	15	8	6	5						
2	15	0	18	9	28	4						
3	24	23	0	13	13	3						
4	15	27	8	0	14	2						
5	8	17	24	15	0	1						
6	1	2	3	4	5		1	2	3	4	5	
7						1	0	1	0	0	0	
8						2	0	0	0	1	0	
9						3	0	0	0	0	1	
10						4	0	0	1	0	0	
11						5	1	0	0	0	0	
12												
13												
14												
15	Constraints											
16	1	=	1		1	=	1			0	0	
17	1	=	1		1	=	1			0	0	
18	1	=	1		1	=	1			0	0	
19	1	=	1		1	=	1			0	0	
20	1	=	1		1	=	1			0	0	
21								=K7	0	0	0	
22												
23	58											

I made the cell 1, 5 = 0.

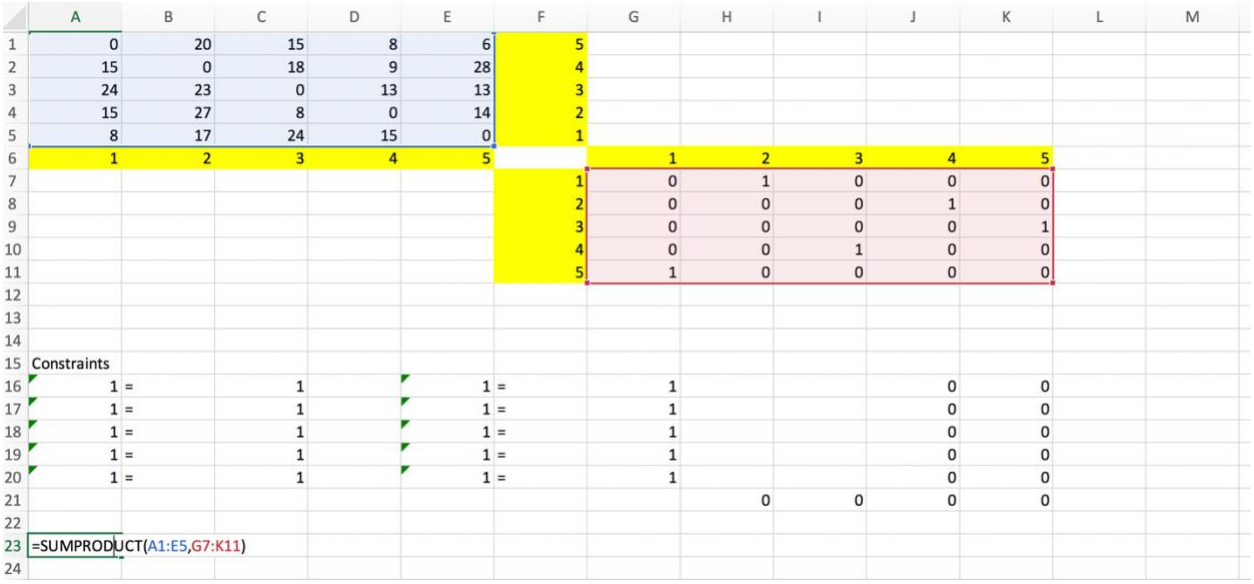
I also made sure the diagonals were all 0s in both matrices:

0	20	15	8	6	5					
15	0	18	9	28	4					
24	23	0	13	13	3					
15	27	8	0	14	2					
8	17	24	15	0	1					
1	2	3	4	5		1	2	3	4	5
					1	0	1	0	0	0
					2	0	0	0	1	0
					3	0	0	0	0	1
					4	0	0	1	0	0
					5	1	0	0	0	0
onstraints										
1	=	1		1	=	1			=G7	0
1	=	1		1	=	1			0	0
1	=	1		1	=	1			0	0
1	=	1		1	=	1			0	0
1	=	1		1	=	1			0	0
							0	0	0	0
58										

Here is the whole setup in Excel after adding the new constraints:

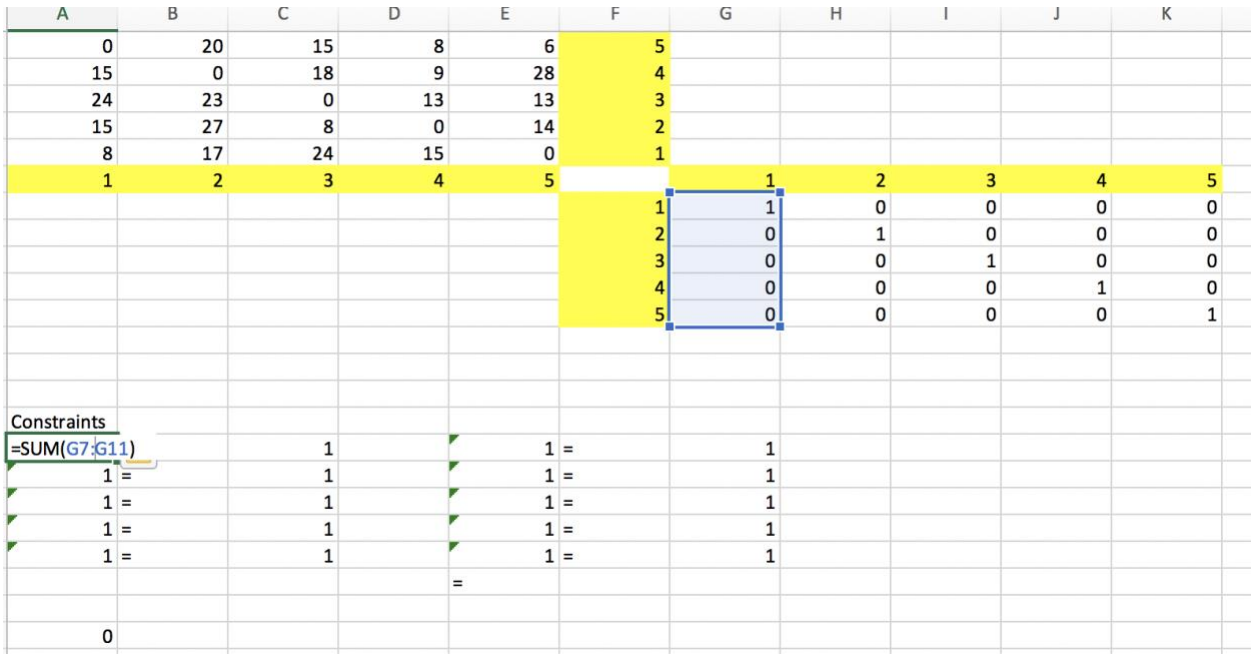


Here are the behind the cell formulas:



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	0	20	15	8	6	5							
2	15	0	18	9	28	4							
3	24	23	0	13	13	3							
4	15	27	8	0	14	2							
5	8	17	24	15	0	1							
6	1	2	3	4	5		1	2	3	4	5		
7						1	0	1	0	0	0		
8						2	0	0	0	1	0		
9						3	0	0	0	0	1		
10						4	0	0	1	0	0		
11						5	1	0	0	0	0		
12													
13													
14													
15	Constraints												
16	1	=	1		1	=	1			0	0		
17	1	=	1		1	=	1			0	0		
18	1	=	1		1	=	1			0	0		
19	1	=	1		1	=	1			0	0		
20	1	=	1		1	=	1			0	0		
21								0	0	0	0		
22													
23	=SUMPRODUCT(A1:E5,G7:K11)												
24													

The Objective Function ^

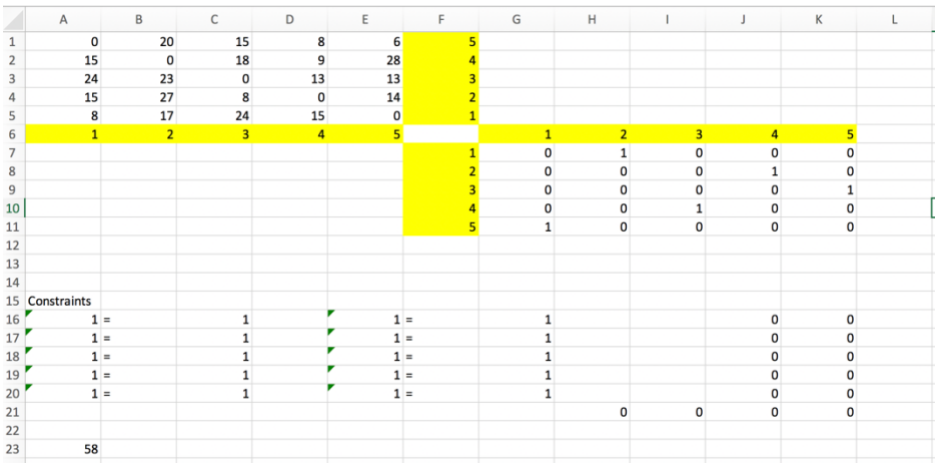


	A	B	C	D	E	F	G	H	I	J	K
	0	20	15	8	6	5					
	15	0	18	9	28	4					
	24	23	0	13	13	3					
	15	27	8	0	14	2					
	8	17	24	15	0	1					
	1	2	3	4	5		1	2	3	4	5
						1	1	0	0	0	0
						2	0	1	0	0	0
						3	0	0	1	0	0
						4	0	0	0	1	0
						5	0	0	0	0	1
	Constraints										
	=SUM(G7:G11)		1		1	=	1				
	1	=	1		1	=	1				
	1	=	1		1	=	1				
	1	=	1		1	=	1				
	1	=	1		1	=	1				
					=						
	0										

Do the same thing for each column and each row

Here is our only constraint which is a subroute between 5 and 1.

For the optimal solution we get: 58, which corresponds to the route **1, 2, 4, 3, 5, 1** and this corresponds to the times 8 + 20 + 8 + 9 + 13= 58.



	A	B	C	D	E	F	G	H	I	J	K	L
1	0	20	15	8	6	5						
2	15	0	18	9	28	4						
3	24	23	0	13	13	3						
4	15	27	8	0	14	2						
5	8	17	24	15	0	1						
6	1	2	3	4	5		1	2	3	4	5	
7						1	0	1	0	0	0	
8						2	0	0	0	1	0	
9						3	0	0	0	0	1	
10						4	0	0	1	0	0	
11						5	1	0	0	0	0	
12												
13												
14												
15	Constraints											
16	1	=	1		1	=	1			0	0	
17	1	=	1		1	=	1			0	0	
18	1	=	1		1	=	1			0	0	
19	1	=	1		1	=	1			0	0	
20	1	=	1		1	=	1			0	0	
21								0	0	0	0	
22												
23	58											

SOLUTION FOR 10 X 10:

Here is the setup for 10 x 10 Matrix with binary matrix. We had to add one constraint to eliminate a subroute at 5,1. After implementing those constraints, I got an optimal solution of 92. I got a path of **1, 2, 6, 3, 5, 4, 10, 7, 8, 9, 1**. The times were 7 + 9 +6 + 7 + 15 + 5 + 6 + 16 + 14 + 7 = 92

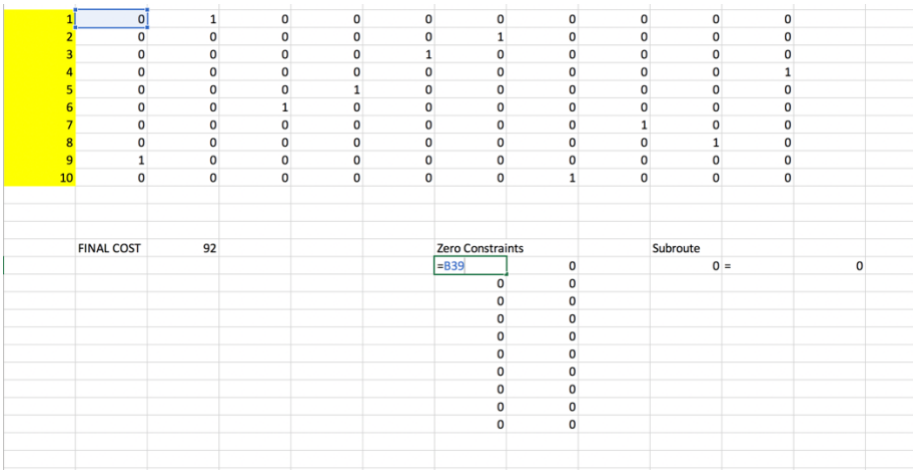
Matrix from Problem:

	1	2	3	4	5	6	7	8	9	10
1	0	9	12	26	11	24	12	13	17	15
2	24	0	28	23	22	5	7	18	9	23
3	19	30	0	30	15	22	25	15	28	15
4	18	10	27	0	28	12	16	19	22	7
5	5	16	11	7	0	25	27	30	23	15
6	7	26	6	17	6	0	28	10	13	28
7	23	26	20	20	24	30	0	16	18	27
8	23	20	22	8	18	10	14	0	14	12
9	7	13	9	19	29	27	18	23	0	30
10	16	10	11	11	28	26	6	11	12	0

Binary Matrix:

1	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	1
5	0	0	0	1	0	0	0	0	0	0
6	0	0	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0
9	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	1	0	0	0

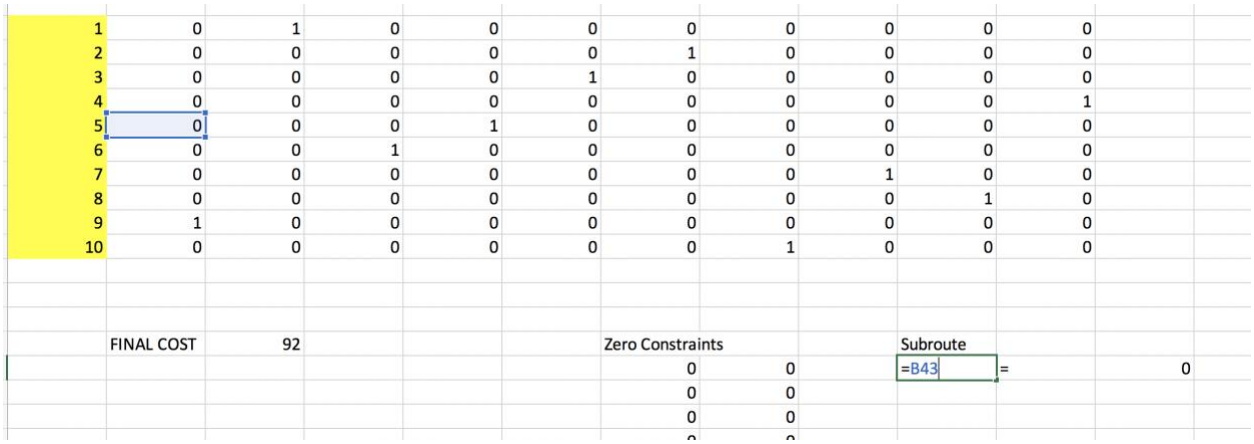
I have to add zero constraints so Binary matrix diagonals stay 0



1	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	1
5	0	0	0	1	0	0	0	0	0	0
6	0	0	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0
9	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	1	0	0	0

FINAL COST	92			Zero Constraints		Subroute			
				=B39	0	0	=	0	
				0	0				
				0	0				
				0	0				
				0	0				
				0	0				
				0	0				
				0	0				
				0	0				
				0	0				

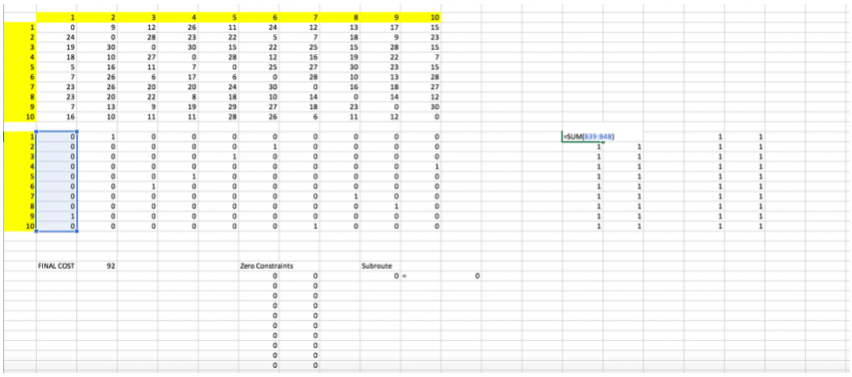
There was a subroute at 5,1 which I set = 0



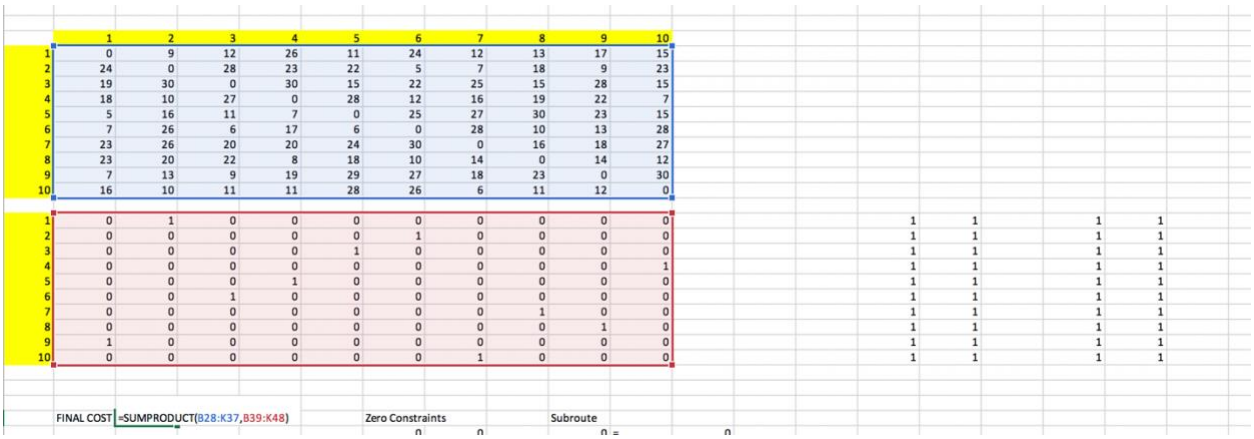
1	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	1
5	0	0	0	1	0	0	0	0	0	0
6	0	0	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0
9	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	1	0	0	0

FINAL COST	92			Zero Constraints		Subroute			
				0	0	=B43	=	0	
				0	0				
				0	0				

I also made constraints, so all rows and columns were = 1 defined here:



My formula below represents optimal minimization function:



	1	2	3	4	5	6	7	8	9	10
1	0	9	12	26	11	24	12	13	17	15
2	24	0	28	23	22	5	7	18	9	23
3	19	30	0	30	15	22	25	15	28	15
4	18	10	27	0	28	12	16	19	22	7
5	5	16	11	7	0	25	27	30	23	15
6	7	26	6	17	6	0	28	10	13	28
7	23	26	20	20	24	30	0	16	18	27
8	23	20	22	8	18	10	14	0	14	12
9	7	13	9	19	29	27	18	23	0	30
10	16	10	11	11	28	26	6	11	12	0

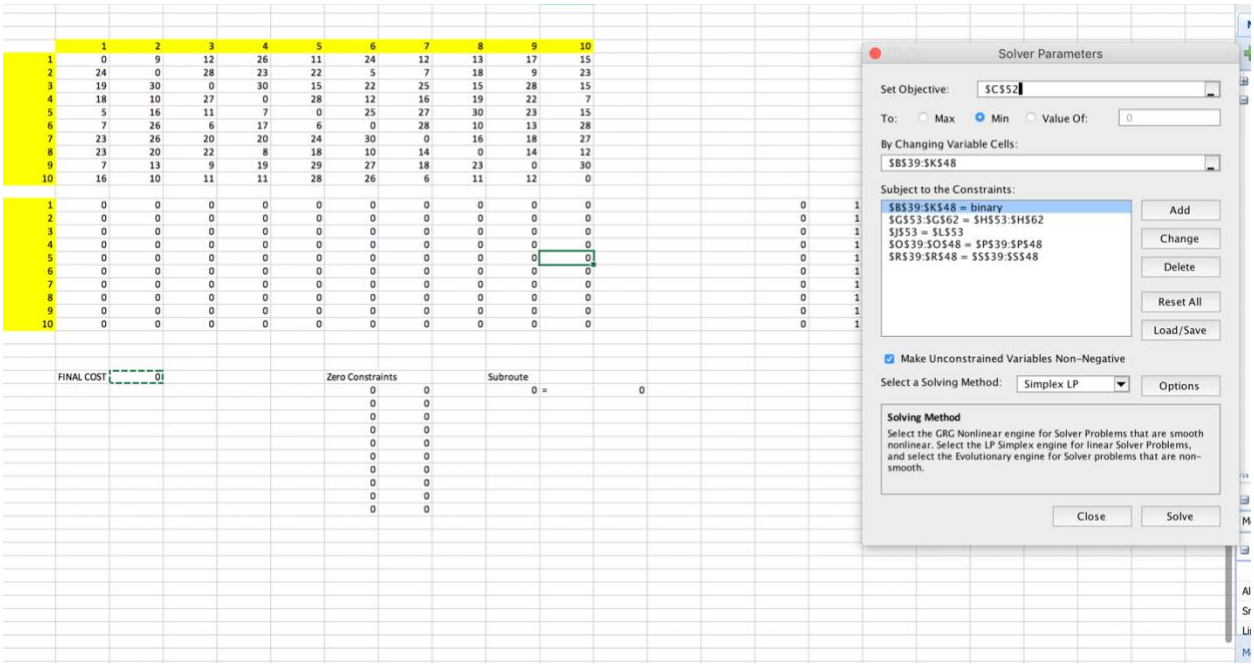
	1	2	3	4	5	6	7	8	9	10
1	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	1
5	0	0	0	1	0	0	0	0	0	0
6	0	0	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0
9	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	1	0	0	0

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

FINAL COST =SUMPRODUCT(B28:K37,B39:K48) Zero Constraints Subroute

0 0 0 = 0

Once I found my answer I put it in solver as below:



	1	2	3	4	5	6	7	8	9	10
1	0	9	12	26	11	24	12	13	17	15
2	24	0	28	23	22	5	7	18	9	23
3	19	30	0	30	15	22	25	15	28	15
4	18	10	27	0	28	12	16	19	22	7
5	5	16	11	7	0	25	27	30	23	15
6	7	26	6	17	6	0	28	10	13	28
7	23	26	20	20	24	30	0	16	18	27
8	23	20	22	8	18	10	14	0	14	12
9	7	13	9	19	29	27	18	23	0	30
10	16	10	11	11	28	26	6	11	12	0

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

0	1	1	1
0	1	1	1
0	1	1	1
0	1	1	1
0	1	1	1
0	1	1	1
0	1	1	1
0	1	1	1
0	1	1	1
0	1	1	1

FINAL COST 0

Zero Constraints Subroute

0 0 0 = 0

0 0

0 0

0 0

0 0

0 0

0 0

0 0

0 0

Solver Parameters

Set Objective: $C$52

To: Max Min Value Of: 0

By Changing Variable Cells:

$B$39:$K$48

Subject to the Constraints:

$B$39:$K$48 = binary
$G$53:$G$62 = $H$53:$H$62
$J$53 = $L$53
$O$39:$O$48 = $P$39:$P$48
$R$39:$R$48 = $S$39:$S$48

Add
Change
Delete
Reset All
Load/Save

Make Unconstrained Variables Non-Negative

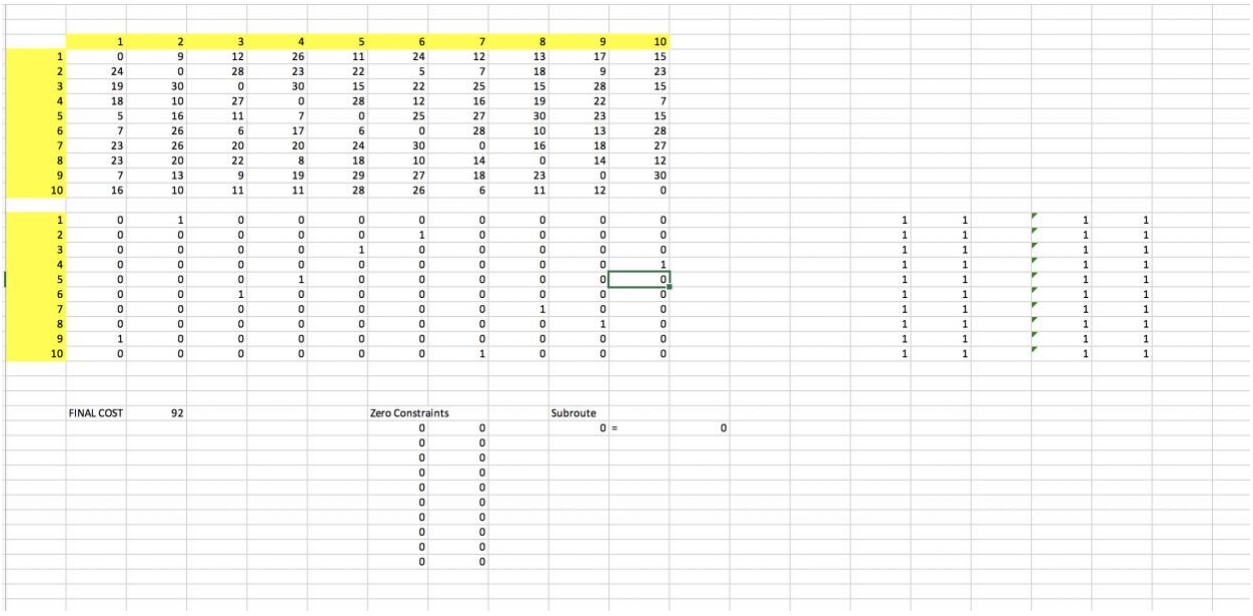
Select a Solving Method: Simplex LP Options

Solving Method

Select the GRG Nonlinear engine for Solver Problems that are smooth nonlinear. Select the LP Simplex engine for linear Solver Problems, and select the Evolutionary engine for Solver problems that are non-smooth.

Close Solve

Final Output/Answer is below as well:



	1	2	3	4	5	6	7	8	9	10
1	0	9	12	26	11	24	12	13	17	15
2	24	0	28	23	22	5	7	18	9	23
3	19	30	0	30	15	22	25	15	28	15
4	18	10	27	0	28	12	16	19	22	7
5	5	16	11	7	0	25	27	30	23	15
6	7	26	6	17	6	0	28	10	13	28
7	23	26	20	20	24	30	0	16	18	27
8	23	20	22	8	18	10	14	0	14	12
9	7	13	9	19	29	27	18	23	0	30
10	16	10	11	11	28	26	6	11	12	0

1	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	1	0	0	0	0
3	0	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	1
5	0	0	0	1	0	0	0	0	0	0
6	0	0	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0
9	1	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	1	0	0	0

	1	1			1	1				
	1	1			1	1				
	1	1			1	1				
	1	1			1	1				
	1	1			1	1				
	1	1			1	1				
	1	1			1	1				
	1	1			1	1				
	1	1			1	1				
	1	1			1	1				

	FINAL COST	92			Zero Constraints		Subroute			
					0	0	0	=	0	
					0	0				
					0	0				
					0	0				
					0	0				
					0	0				
					0	0				
					0	0				
					0	0				
					0	0				

4.

MIN Priority1*(d1$^+$) + Priority2*(2*d_2^+ + d_3^+ + d_4^+) + Priority3*(d_5^- + d_5^+ + d_6^+)

D1 – minimize overutilization as we do not want it to go over 40

D2 – minimize overutilization as we do not want to go over 16,000

D3 - minimize overutilization as we do not want to go over 5,000

D4 - minimize overutilization as we do not want to go over 9,000

D5 & D6 - minimize both deviational variables for budget as mentioned and minimize underutilization for d6 as we to use all the labor hours we have available.

Priority 1

GOAL 1: 2X1 + 2X2 + 1X3+ d_1^- - d_1^+ = 40

Priority 2

GOAL 2: 1.5X1 + 2X2 + 1X3+ d_2^- - d_2^+ = 16,000

GOAL 3: 0.5X1 + 0.5X2 + 1X3 + d_3^- - d_3^+ = 5,000

GOAL 4: 0.3X1 + 0.6X2 + d_4^- - d_4^+ = 9,000

Priority 3

GOAL 5: $7X1 + 5X2 + 4X3 + d_5^- - d_5^+ = 164{,}000$

GOAL 6: $2X1 + 2X2 + 1X3 + d_6^- - d_6^+ = 40$

X1, X2, X3, $d_1 \ldots d_6 >= 0$

5.

We need to minimize the deviational variables $d_{1 \ldots} d_6$.

MIN $Priority1*(d_1^-) + Priority2*(d_2^+) + Priority3*(d_3^+) + Priority4*(d_4^-) + Priority5*(d_5^- + 2*d_6^-)$

D1 – minimize underutilization as we do not want it below 750,000

D2 – minimize overutilization as we do not want to go over 100,000

D3 - minimize overutilization as we do not want to go over 70,000

D4 - minimize underutilization as we do not want to go under 1,000,000

D5 & D6 - minimize underutilization as we do not want go under 250,000 we also know the buying power for D6 which is 25-30 age group is twice as much there for we multiply 2*d6.

Priority 1

GOAL 1: $10{,}000X1 + 7{,}500X2 + d_1^- - d_1^+ = 750{,}000$ - we know that we want exposure to be at least 750,000

Priority 2

GOAL 2: $X1 + X2 + d_2^- - d_2^+ = 100$ = we have to divide 100,000/1,000 to get per 1000 dollars and scale correctly

Priority 3

GOAL 3: $X1 + d_3^- - d_3^+ = 70$ we have to divide 70,000/1,000 to get per 1000 dollars and scale correctly

Priority 4

GOAL 4:10,000X1 + 7,500X2 + d_4^- - d_4^+ = 1,000,000 we want at least 1,000,000 exposures

Priority 5

GOAL 5: 2,500X1 + 3,000X2 + d_5^- - d_5^+ = 250,000

GOAL 6: 3,000X1 + 1,500X2 + d_6^- - d_6^+ = 250,000

These two are for age groups.

X1, X2 $d_1 \ldots d_6$ > = 0

EXTRA CREDIT

A.

Patterns that the lumberyard may use to use a board of 10 feet is below.

To figure out how many patterns they will be we need to use combination formula which tells us there will be 6 patterns. 3!/(3-3)! Where r is that we have 3 options to choose from out of n elements which is also 3.

Where left over materials are less than 3 feet is:

Pattern #1 – 3 feet Boards * 3 = 9 feet used

Pattern #2 – 3 feet board *2 + 1*4 feet board = 10 feet used

Pattern #3 – 3 feet board + 5 feet board = 8 feet used

Pattern #4 – 5 feet boards * 2 = 10 feet used

Pattern #5 – 5 feet board + 4 board = 9 feet used

Pattern #6 = 4*2 = 8 feet used

B.

After figuring out our patterns we can use decision variables for each of our patterns which is equal to 6. X1…X6.

Then we can find our objective function which is to limit how many 10 feet boards we can make which is

Min X1 + X2 + X3 + X4 + X5 +X6

S.T.

We have 90 boards of 3 feet available so:

Constraint 1 is

3*X1 + 2*X2 + X3 == 90

We have 60 boards of 4 feet available so:

X2 + X5+ 2*X6 == 60

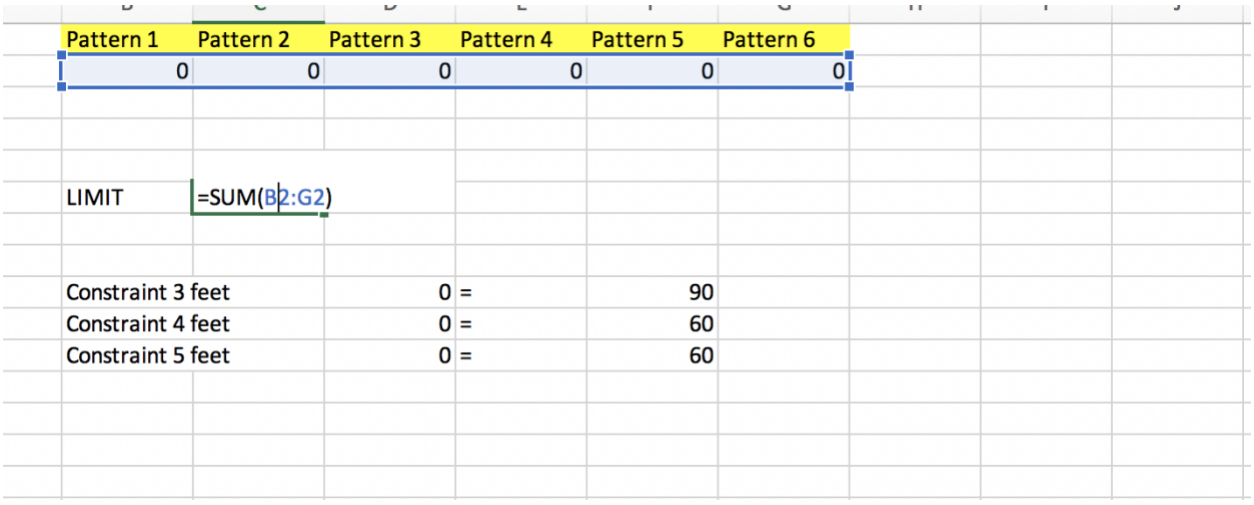
We have 60 boards of 5 feet available so:

X3 + 2*X4+ X5 == 60

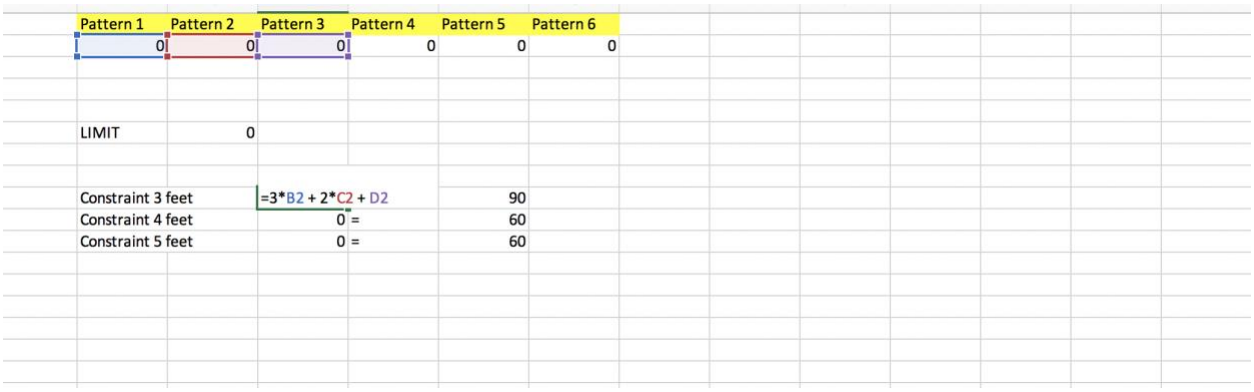
MAKE SURE X1…X6 >= 0

C. Now put above in Excel:

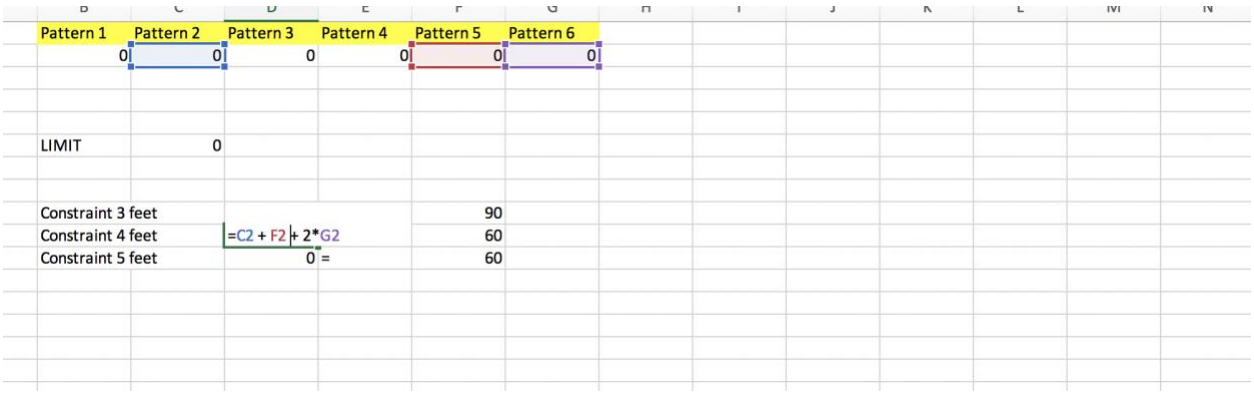
Our formulas in Excel shown before:



The Objective function is above.

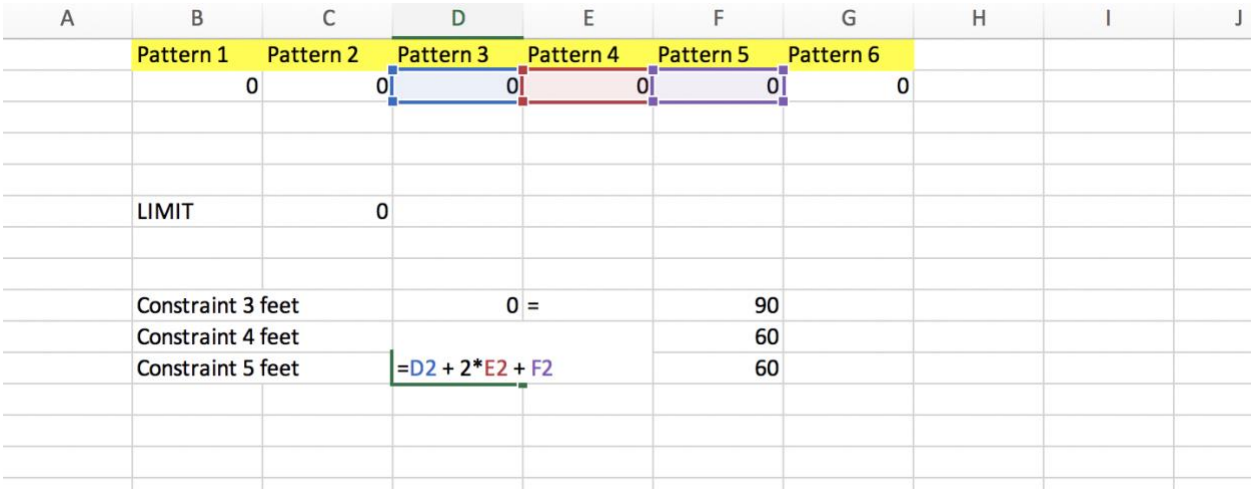


Constraint for 3 feet ^

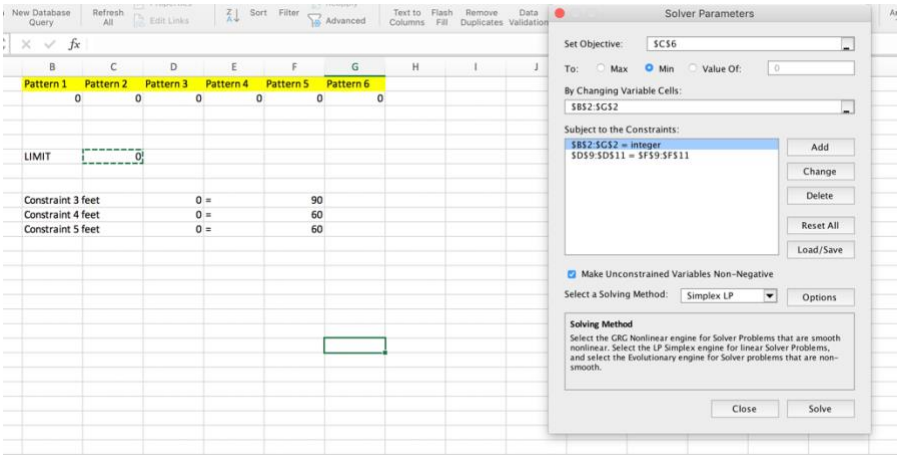


Constraint for 4 feet ^

And Constraint for 5 feet



Here is our setup before solving:



We get an optimal solution of 83 with Pattern 1 having 2 boards, Pattern 2 having 42 boards, Pattern 4 having 30 boards and Pattern 6 having 9 boards.

	Pattern 1	Pattern 2	Pattern 3	Pattern 4	Pattern 5	Pattern 6
	2	42	0	30	0	9
	LIMIT	83				
	Constraint 3 feet		90	=	90	
	Constraint 4 feet		60	=	60	
	Constraint 5 feet		60	=	60	

NOW WE CAN PUT SAME PROBLEM IN GUROBI PYTHON:

In [34]:
```
q3 = Model()
Pattern1 = q3.addVar(vtype=GRB.INTEGER)
Pattern2 = q3.addVar(vtype=GRB.INTEGER)
Pattern3 = q3.addVar(vtype=GRB.INTEGER)
Pattern4 = q3.addVar(vtype=GRB.INTEGER)
Pattern5 = q3.addVar(vtype=GRB.INTEGER)
Pattern6 = q3.addVar(vtype=GRB.INTEGER)

q3.update()

q3.addConstr(3*Pattern1 + 2*Pattern2 + Pattern3 == 90)
q3.addConstr(Pattern2 + Pattern5 + 2*Pattern6 == 60)
q3.addConstr(Pattern3 + 2*Pattern4 + Pattern5 == 60)
q3.update()
LIMIT = Pattern1 + Pattern2 + Pattern3 + Pattern4 + Pattern5  + Pattern6
q3.setObjective(LIMIT,GRB.MINIMIZE)
q3.optimize()
```

```
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (mac64)
Optimize a model with 3 rows, 6 columns and 9 nonzeros
Model fingerprint: 0xab2cd2c7
Variable types: 0 continuous, 6 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 3e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [6e+01, 9e+01]
Presolve time: 0.00s
Presolved: 3 rows, 6 columns, 9 nonzeros
Variable types: 0 continuous, 6 integer (0 binary)
Found heuristic solution: objective 100.0000000
Found heuristic solution: objective 90.0000000

Root relaxation: objective 8.250000e+01, 3 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0   82.50000    0    1   90.00000   82.50000  8.33%     -    0s
H    0     0                       83.0000000   82.50000  0.60%     -    0s
     0     0   82.50000    0    1   83.00000   82.50000  0.60%     -    0s

Explored 1 nodes (3 simplex iterations) in 0.03 seconds
Thread count was 4 (of 4 available processors)

Solution count 3: 83 90 100

Optimal solution found (tolerance 1.00e-04)
Best objective 8.300000000000e+01, best bound 8.300000000000e+01, gap 0.0000%
```

In [32]:
```
q3.getVars()
```

Out[32]:
```
[<gurobi.Var C0 (value -0.0)>,
 <gurobi.Var C1 (value 46.0)>,
 <gurobi.Var C2 (value -0.0)>,
 <gurobi.Var C3 (value 30.0)>,
 <gurobi.Var C4 (value -0.0)>,
 <gurobi.Var C5 (value 7.0)>]
```

In [33]:
```
q3.printStats()
```

```
Statistics for model Unnamed :
  Linear constraint matrix    : 3 Constrs, 6 Vars, 9 NZs
  Variable types              : 0 Continuous, 6 Integer
  Matrix coefficient range    : [ 1, 3 ]
  Objective coefficient range : [ 1, 1 ]
  Variable bound range        : [ 0, 0 ]
  RHS coefficient range       : [ 60, 90 ]
```