Open in Colab

Math 210

Aaron Graybill

Problem Set 6

4/8/21

Working with Gabe S. and Eli M.

I don't do a ton of explaining but I'm pressed for time and I have worked through all of the stuff independently so i do know what's happening if that's worth anything.

In [1]:
```
import numpy as np
from scipy.optimize import linprog
```

Problem 1.

1.a

In [22]:
```
a1=-np.concatenate((np.identity(4),np.identity(4),np.identity(4)),axis=1)
a2=np.array([[1,1,1,1,0,0,0,0,0,0,0,0],[0,0,0,0,1,1,1,1,0,0,0,0],[0,0,0,0,0,0,0,0,1,1,1,1]])
a=np.append(a1,a2,axis=0)
print(a)

b=np.array([-14,-12, -19, -11,20,15,25])


c=np.array([4,2,3,5,6,4,2,2,2,4,5,4])


linprog(c,A_ub=a,b_ub=b,method="simplex").x.reshape((3,4))
```

```
[[-1. -0. -0. -0. -1. -0. -0. -0. -1. -0. -0. -0.]
 [-0. -1. -0. -0. -0. -1. -0. -0. -0. -1. -0. -0.]
 [-0. -0. -1. -0. -0. -0. -1. -0. -0. -0. -1. -0.]
 [-0. -0. -0. -1. -0. -0. -0. -1. -0. -0. -0. -1.]
 [ 1.  1.  1.  1.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.  1.  1.  1.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  1.  1.  1.  1.]]
```

Out[22]:
```
array([[ 0., 12.,  8.,  0.],
       [ 0.,  0., 11.,  4.],
       [14.,  0.,  0.,  7.]])
```

So factory one sends 12 to customer 3, and the remaining to factory 3 and so on.

1.b

In [23]:
```
a1=np.concatenate((np.identity(5),np.identity(5),np.identity(5)),axis=1)
a2=np.array([
             [1,1,1,1,1,0,0,0,0,0,0,0,0,0,0],
             [0,0,0,0,0,1,1,1,1,1,0,0,0,0,0],
             [0,0,0,0,0,0,0,0,0,0,1,1,1,1,1],
             ])
a_eq=np.append(a1,a2,axis=0)
print(a_eq)

b_eq=np.array([14,12,19,11,4,20,15,25])
c=np.array([4,2,3,5,0,6,4,2,2,0,2,4,5,4,0])

linprog(c,A_eq=a_eq,b_eq=b_eq,method="simplex").x.reshape((3,5))
```

```
[[1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1.]]
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:13: OptimizeWarning: A_eq does not appear to be of
```

```
full row rank. To improve performance, check the problem formulation for redundant equality constraints.
  del sys.path[0]
```

Out[23]:
```
array([[ 0., 12.,  8.,  0.,  0.],
       [ 0.,  0., 11.,  4.,  0.],
       [14.,  0.,  0.,  7.,  4.]])
```

This is nice because it produces the same output while telling us directly that it is factory three that is leaving 4 unsent.

Problem 2

In [4]:
```
def assignment_matrix(n_workers,n_tasks):
  #create output matrix with first identity
  a=np.identity(n_tasks)
  # for the remaining number of workers, create an idenitity and add to right
  for i in range(n_workers-1):
    a=np.concatenate((a,np.identity(n_tasks)),axis=1)


  # for the number of workers make a vector of 1s and zeros as desired
  for i in range(n_workers):
    lower_index=n_tasks*i #first index that should be 1
    upper_index=n_tasks*(i+1) #last index that should be 1
    new_vec=np.zeros((n_workers*n_tasks)) #vector of all zeros
    new_vec[lower_index:upper_index]=1 #add ones where required
    new_vec=np.array([new_vec]) # make dimensions conformable with a
    a=np.concatenate((a,new_vec),axis=0) #add new row
  return a

assignment_matrix(6,6).shape
```

Out[4]:
```
(12, 36)
```

In [25]:
```
a_eq=assignment_matrix(6,6)
b_eq=np.ones(12)
c=np.array([6, 8, 5, 9, 6, 7,3, 5, 7, 4, 8, 7,4, 8, 6, 8, 9, 7,7, 5, 5, 6, 4, 3,9, 7, 3, 3, 7, 5,8, 5, 7, 5, 7,

out=linprog(c,A_eq=a_eq,b_eq=b_eq,method="simplex")

import pandas as pd
row_names=["E1","E2","E3","E4","E5","E6"]
column_names=["task1","task2","task3","task4","task5","task6"]
pd.DataFrame(out.x.reshape(6,6), columns=column_names, index=row_names)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: OptimizeWarning: A_eq does not appear to be of f
ull row rank. To improve performance, check the problem formulation for redundant equality constraints.
  """
```

Out[25]:

	task1	task2	task3	task4	task5	task6
E1	0.0	0.0	0.0	0.0	1.0	0.0
E2	0.0	0.0	0.0	1.0	0.0	0.0
E3	1.0	0.0	0.0	0.0	0.0	0.0
E4	0.0	0.0	0.0	0.0	0.0	1.0
E5	0.0	0.0	1.0	0.0	0.0	0.0
E6	0.0	1.0	0.0	0.0	0.0	0.0

Problem 3.

In [26]:
```
print("Horizontal Blocks")
horz=np.abs(np.subtract.outer([4,1,3,6],[3,7,2,7,1]))
print(horz)

print("Vertical Blocks")
vert=np.abs(np.subtract.outer([42,64,41,88],[50,72,90,30,33]))
print(vert)

print("taxicab distance, tenths of miles")
taxi_distance=np.transpose(2*horz+1*vert)
print(taxi_distance)
```

```
Horizontal Blocks
[[1 3 2 3 3]
 [2 6 1 6 0]
 [0 4 1 4 2]
 [3 1 4 1 5]]
Vertical Blocks
```

```
[[ 8 30 48 12  9]
 [14  8 26 34 31]
 [ 9 31 49 11  8]
 [38 16  2 58 55]]
taxicab distance, tenths of miles
[[10 18  9 44]
 [36 20 39 18]
 [52 28 51 10]
 [18 46 19 60]
 [15 31 12 65]]
```

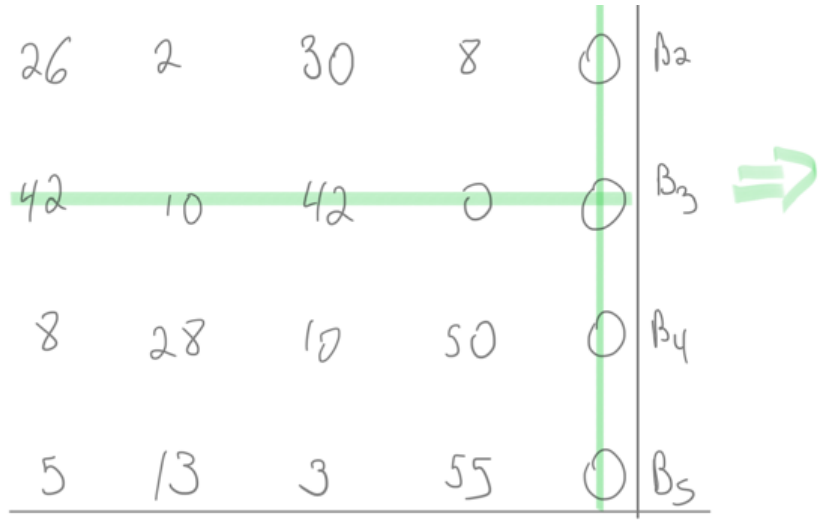
R_1	R_2	R_3	R_4	R_F	
10	18	9	44	0	B_1
36	20	39	18	0	B_2
52	28	51	10	0	B_3
18	46	19	60	0	B_4
15	31	12	65	0	B_5

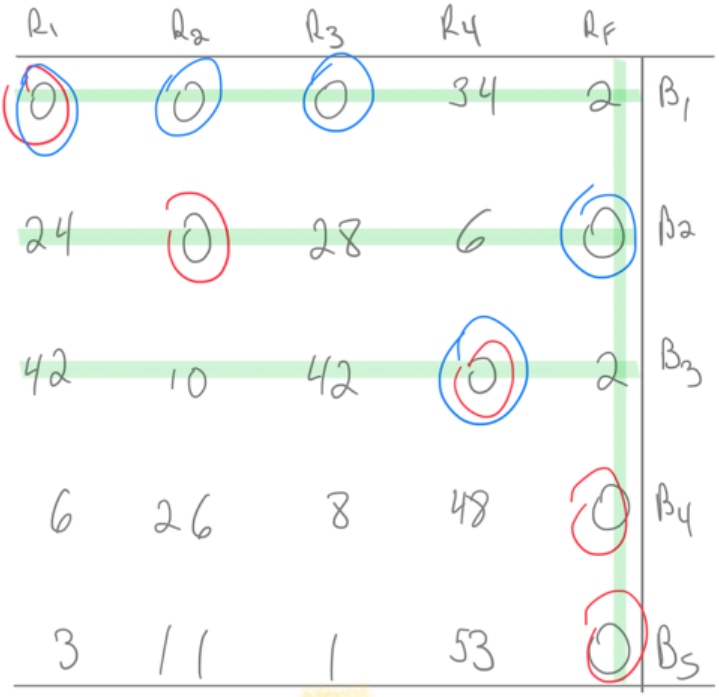
⇒

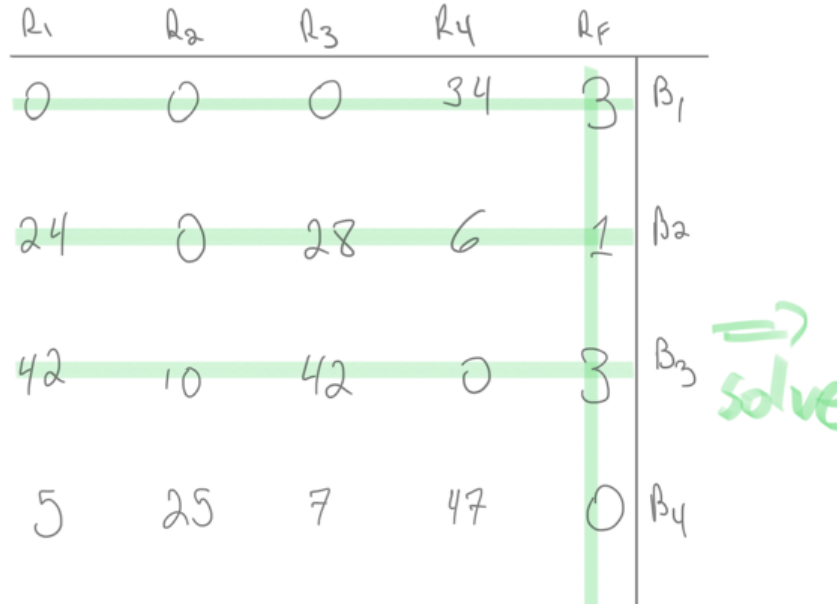
R_1	R_2	R_3	R_4	R_F	
10	18	9	44	0	B_1
36	20	39	18	0	B_2
52	28	51	10	0	B_3
18	46	19	60	0	B_4
15	31	12	65	0	B_5

⇒

R_1	R_2	R_3	R_4	R_F	
0	0	0	34	0	B_1


26 2 30 8 0 B2
42 10 42 0 0 B3
8 28 17 50 0 B4
5 13 3 55 0 B5


R1 R2 R3 R4 RF
0 0 0 34 2 B1
24 0 28 6 0 B2
42 10 42 0 2 B3
6 26 8 48 0 B4
3 11 1 53 0 B5


R1 R2 R3 R4 RF
0 0 0 34 3 B1
24 0 28 6 1 B2
42 10 42 0 3 B3
5 25 7 47 0 B4

So biker one goes to restaurant one and biker two goes to restaurant 2 and so on, with biker 4 not doing anything.

Problem 4.

In [17]:
```
assignment_matrix(8,64)


student_ranks=np.array(
    [[1,2,4,3],
     [2,4,1,3],
     [4,3,1,2],
     [3,2,1,4],
     [2,3,4,1],
     [3,1,2,4],
     [3,1,4,2],
     [3,4,2,1]])

faculty_ranks=np.array(
          [[1, 3 ,8, 5, 7, 6, 4, 2],
          [3, 4, 7, 6, 2, 8, 1, 5],
          [3, 2, 4, 1, 5, 8, 7, 6],
          [7, 1, 3, 4, 5, 8, 2, 6]])


full_rating=np.concatenate((student_ranks+np.transpose(faculty_ranks),student_ranks+np.transpose(faculty_ranks)))
print(full_rating)


c=np.array(full_rating.flatten())
b=np.ones(16).flatten()
#print(b.shape)

out=linprog(c,A_eq=assignment_matrix(8,8),b_eq=b,method="simplex")

#print(out.x.reshape(8,8))

import pandas as pd
column_names=["Dumbledore1", "Snape1", "Sprout1", "Flitwick1","Dumbledore2", "Snape2", "Sprout2", "Flitwick2"]
row_names=["Harry", "Ron", "Hermione", "Parvati", "Neville", "Ginny", "Cho", "Pansy"]
pd.DataFrame(out.x.reshape(8,8), columns=column_names, index=row_names)
```

```
[[ 2  5  7 10  2  5  7 10]
 [ 5  8  3  4  5  8  3  4]
 [12 10  5  5 12 10  5  5]
 [ 8  8  2  8  8  8  2  8]
 [ 9  5  9  6  9  5  9  6]
 [ 9  9 10 12  9  9 10 12]
 [ 7  2 11  4  7  2 11  4]
 [ 5  9  8  7  5  9  8  7]]
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:29: OptimizeWarning: A_eq does not appear to be of
full row rank. To improve performance, check the problem formulation for redundant equality constraints.
```

Out[17]:

	Dumbledore1	Snape1	Sprout1	Flitwick1	Dumbledore2	Snape2	Sprout2	Flitwick2
Harry	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
Ron	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
Hermione	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
Parvati	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
Neville	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
Ginny	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
Cho	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
Pansy	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

The assignments are as detailed above

Problem 5

In [43]:

```
block=np.array([[-1, 0, 0, -1, -1, -1, -1],
                [-1, -1, 0, 0, -1, -1 ,-1],
                [-1, -1, -1, 0 ,0, -1, -1],
                [-1, -1, -1 ,-1, 0, 0 ,-1],
                [-1, -1, -1, -1, -1, 0, 0],
                [0, -1, -1, -1, -1, -1, 0],
                [0, 0, -1, -1, -1, -1, -1]])
zeros=np.zeros((7,7))

#print(block)
#print(zeros)

A=np.block([[block,zeros,zeros],
            [block,block,zeros],
            [zeros,block,block],
            [zeros,zeros,block]])

week_b = lambda week_need,weekend_need : np.concatenate((np.repeat(week_need,5),
                                                         np.repeat(weekend_need,2)
                                                         ))

b=-np.concatenate((
    week_b(144,72),
    week_b(108,144),
    week_b(96,120),
    week_b(72,96)
))

c=np.concatenate((np.repeat(13,7),np.repeat(11,7),np.repeat(12,7)))

print(b)

out=linprog(c,A_ub=A,b_ub=b,method="simplex")

column_names=["S1", "S2", "S3", "S4","S5", "S6", "S7"]
row_names=["5am-1pm","9am-5pm","1pm-9pm"]
pd.DataFrame(out.x.round().reshape((3,7)), columns=column_names, index=row_names)
```

```
[-144 -144 -144 -144 -144  -72  -72 -108 -108 -108 -108 -108 -144 -144
  -96  -96  -96  -96  -96 -120 -120  -72  -72  -72  -72  -72  -96  -96]
```

Out[43]:

	S1	S2	S3	S4	S5	S6	S7
5am-1pm	48.0	24.0	24.0	24.0	24.0	24.0	24.0
9am-5pm	0.0	8.0	8.0	0.0	8.0	0.0	8.0
1pm-9pm	0.0	16.0	16.0	24.0	16.0	24.0	16.0

This output indicates we get 24 of shift one schedule two etc.