Open in Colab

Math 210

Aaron Graybill

Problem Set 2

3/4/21

Problem 1.

a.

The objective function remains the same: $g(x_1, x_2) = 30x_1 + 12x_2$.

We have the slack equations:

$$10x_1 + 5x_2 - 150 = -t_1 \tag{1}$$
$$4x_1 + 3x_2 - 80 = -t_2 \tag{2}$$

With the same non-negativity constraints, $x_1, x_2 \geq 0$ and well as $t_1, t_2 \geq 0$.

b.

Solving (1) for x_1 gives:

$$x_1 = -\frac{1}{10}t_1 - \frac{1}{2}x_2 + 15 \tag{3}$$

Plugging this into g gives:

$$\begin{aligned} g(-\frac{1}{10}t_1 - \frac{1}{2}x_2 + 15, x_2) &= 30\left(-\frac{1}{10}t_1 - \frac{1}{2}x_2 + 15\right) + 12x_2 \\ &= -3t_1 - 3x_2 + 450 \end{aligned}$$

We could also rewrite the constraints as:

$$\frac{1}{10}t_1 + \frac{1}{2}x_2 - 15 = -x_1 \tag{1}$$
$$4\left(-\frac{1}{10}t_1 - \frac{1}{2}x_2 + 15\right) + 3x_2 - 80 = -t_2 \implies \tag{2}$$
$$-\frac{2}{5}t_1 + x_2 - 20 = -t_2 \tag{3}$$

c.

Look at the rewritten objective function. Note that by our new non-negativity constraints both x_2 and t_1 must be greater than or equal to zero. This allows to easily see that the theoretical maximum of this this function is 450 (because it decreasing in x_2, t_1). We have then to show that we can actually attain 450. In order for this to happen both $x_2 = t_1 = 0$. Plugging into our constraints gives:

$$\frac{1}{10}0 + \frac{1}{2}0 - 15 = -x_1 \iff x_1 = 15$$
$$-\frac{2}{5}0 + 0 - 20 = -t_2 \iff t_2 = 20$$

Neither of those are outside the constraint set. Therefore our full answer must be:
$x_1^* = 15, x_2^* = 0, t_1^* = 0, t_2^* = 15, g(x_1^*, x_2^*) = 450$

d.

We can interpret the t_is in the following way. In both cases the t_i gives the amount of remaining slack in constraint i. But in context:

> $t_1^* = 0$: For the solution that maximizes sales, the bakery has 0 units of flour remaining.
>
> $t_2^* = 20$: For the solution that maximizes sales, the baker has 20 units of sugar remaining.

Problem 2.

The tableau as given is:

d	c	-1	
10	5	150	$=-t_1$
4	3	80	$=-t_2$
30	20	0	$=obj$

The red cell is the pivot point.

We should first empty the tableau and rename appropriately:

t_2	c	-1	
			$=-t_1$
			$=-d$
			$=obj$

Let's add $\frac{1}{p}$ giving:

t_2	c	-1

t_2	c	-1	
			$= -t_1$
$\frac{1}{4}$			$= -d$
			$= obj$

Then let's add the $\frac{q}{p}$ in the same row as p giving:

t_2	c	-1	
			$= -t_1$
$\frac{1}{4}$	$\frac{3}{4}$	20	$= -d$
			$= obj$

Now let's do the $-\frac{q}{p}$ in the same column giving:

t_2	c	-1	
$-\frac{5}{2}$			$= -t_1$
$\frac{1}{4}$	$\frac{3}{4}$	20	$= -d$
$-\frac{15}{2}$			$= obj$

Now applying the $\frac{ps-qr}{p}$ to the other entries gives:

t_2	c	-1	
$-\frac{5}{2}$	$-\frac{5}{2}$	-50	$= -t_1$
$\frac{1}{4}$	$\frac{3}{4}$	20	$= -d$
$-\frac{15}{2}$	$-\frac{5}{2}$	-60	$= obj$

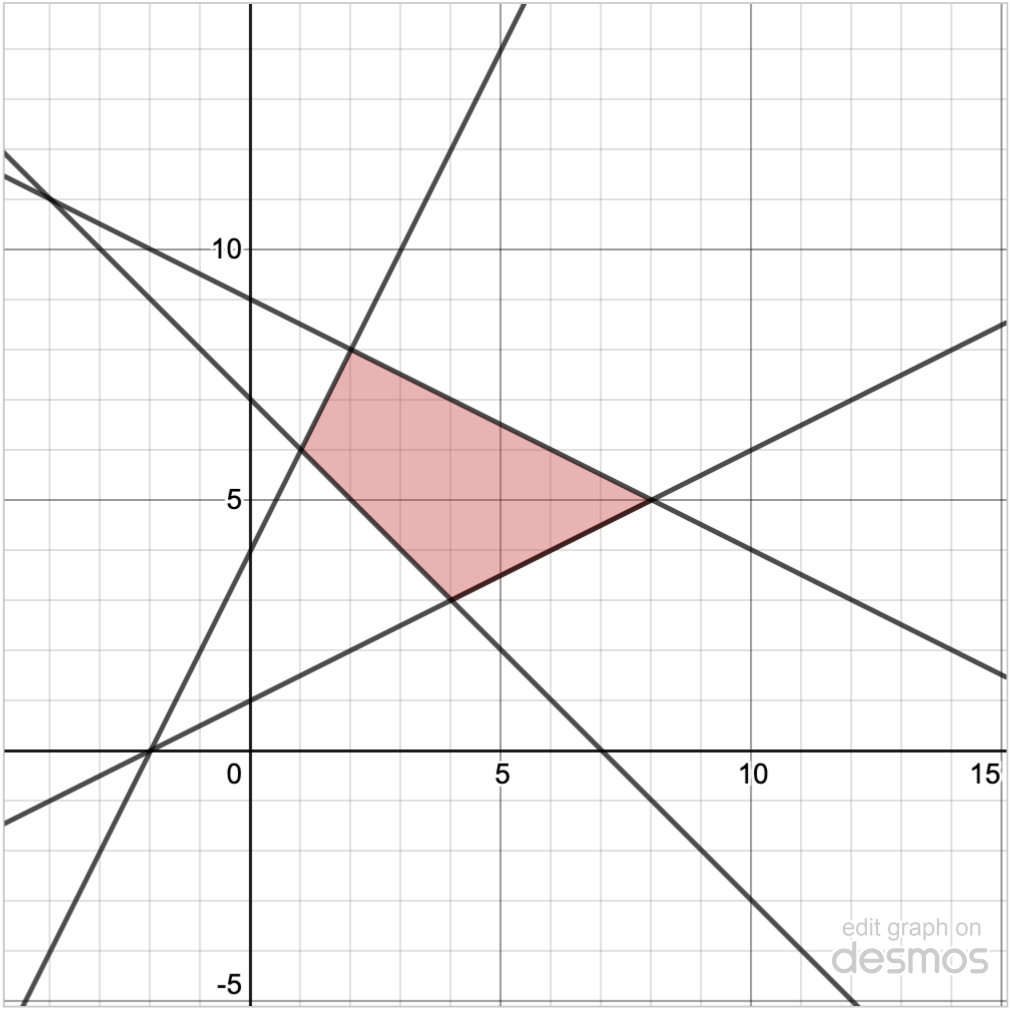
And then making sure that works:

Problem 3

1.

The graph of the region is as follows:

In [9]:
```
%%html
<iframe src="https://www.desmos.com/calculator/6ubr2jksrz?embed" width="500px" h
```



2.

This only needs some minor manipulations to put it into `CanonicalMax` form. It is as follows:

$$\begin{aligned} x - 2y &\leq -2 \\ -x - y &\leq -7 \\ -2x + y &\leq 4 \\ x + 2y &\leq 18 \\ x, y &\geq 0 \end{aligned}$$

3.

Getting this to a Tucker tableau would then be:

$$\begin{array}{ccc|l} x & y & -1 & \\ \hline 1 & -2 & -2 & = -t_1 \\ -1 & -1 & -7 & = -t_2 \\ -2 & 1 & 4 & = -t_3 \\ 1 & 2 & 18 & = -t_4 \\ 2 & 3 & 0 & = obj \end{array}$$

4.

In the below hidden code block I pull in the functions Rob created to pivot tableaus.

In [10]:
```
#@title
import numpy as np

def print_tableau(a,indep_names,dep_names):
#
# Given matrix "a" and lists of variables names "indep_names" and "dep_names",
# this function prints the matrix and labels in standard tableau format
# (including adding the -1, the minus signs in the last column, and labeling the
#
# First, check the inputs: indep_names should be one shorter than the number of
#                          dep_names should be one shorter than the number of ro
#
    nrows = a.shape[0]    # use the shape function to determine number of rows a
    ncols = a.shape[1]
    nindep = len(indep_names)
    ndep = len(dep_names)
    if nindep != ncols-1:
        print("WARNING: # of indep vbles should be one fewer than # columns of m
    if ndep != nrows-1:
        print("WARNING: # of dep vbles should be one fewer than # rows of matrix
# Now do the printing (uses a variety of formatting techniques in Python)
    for j in range(ncols-1):                     # Print the independent variable
        print(indep_names[j].rjust(10),end="")   # rjust(10) makes fields 10 wide
                                                 #    the end command prevents ne
    print("        -1")                          # Tack on the -1 at the end of t
    for i in range(nrows-1):
        for j in range(ncols):                   # Print all but the last row of
            print("%10.3f" % a[i][j],end="") # The syntax prints in a field 10 w
        lab = "= -" + dep_names[i]
        print(lab.rjust(10))
    for j in range(ncols):
        print("%10.3f" % a[nrows-1][j],end="")   # Print the last row of the matr
    lab = "= obj"
    print(lab.rjust(10))
    print(" ")    # Put blank line at bottom


def pivot(a,pivrow,pivcol,indep_names,dep_names) :
#
# Given matrix "a", a row number "pivrow" and column number "pivcol",
#  and lists of variable names "indep_names" and "dep_names", this
#  function does three things:
#    (1) outputs the new version of the matrix after a pivot,
#    (2) updates the lists of variable names post-pivot
#    (3) prints the new matrix, including labels showing the variable names
#
# First, check the inputs: indep_names should be one shorter than the number of
#                          dep_names should be one shorter than the number of ro
#                          you should not be pivoting on the last row or last co
#
    a = a.astype(float)   # make sure entries are treated as floating point numb
    nrows = a.shape[0]    # use the shape function to determine number of rows a
    ncols = a.shape[1]
    nindep = len(indep_names)
    ndep = len(dep_names)
```

```
    if nindep != ncols-1:
        print("WARNING: # of indep vbles should be one fewer than # columns of m
    if ndep != nrows-1:
        print("WARNING: # of dep vbles should be one fewer than # rows of matrix
    if pivrow > nrows-1 or pivcol > ncols-1:
        print("WARNING: should not pivot on last row or column")
    newa = a.copy()       # make a copy of A, to be filled in below with result
    p = a[pivrow-1][pivcol-1]   # identify pivot element
    newa[pivrow-1][pivcol-1] = 1/p   # set new value of pivot element
    # Set entries in p's row
    for j in range(ncols):
        if j != pivcol-1:
            newa[pivrow-1][j]=a[pivrow-1][j]/p;
    # Set entries in p's column
    for i in range(nrows):
        if i != pivrow-1:
            newa[i][pivcol-1]=-a[i][pivcol-1]/p;
    # Set all other entries
    for i in range(nrows):
        for j in range(ncols):
            if i != pivrow-1 and j != pivcol-1:
                r = a[i][pivcol-1]
                q = a[pivrow-1][j]
                s = a[i][j]
                newa[i][j]=(p*s-q*r)/p
    # Now swap the variable names
    temp = indep_names[pivcol-1]
    indep_names[pivcol-1]=dep_names[pivrow-1]
    dep_names[pivrow-1]=temp
    print_tableau(newa,indep_names,dep_names) # Print the matrix with updated la
    return newa;
```

With those imported, and pivoting as desired gives:

In [13]:

```
#Define Problem Parameters
indep_names=["x","y"]
dep_names=["t1","t2","t3","t4"]
b_1=np.array([[1,-2,-2],[-1,-1,-7],[-2,1,4],[1,2,18],[2,3,0]])
print_tableau(b_1,indep_names,dep_names)
```

```
         x          y         -1
     1.000     -2.000     -2.000      = -t1
    -1.000     -1.000     -7.000      = -t2
    -2.000      1.000      4.000      = -t3
     1.000      2.000     18.000      = -t4
     2.000      3.000      0.000      = obj
```

In [14]:

```
#Do First Pivot:
b_2=pivot(b_1,1,1,indep_names,dep_names)
```

```
        t1          y         -1
     1.000     -2.000     -2.000       = -x
     1.000     -3.000     -9.000      = -t2
     2.000     -3.000      0.000      = -t3
    -1.000      4.000     20.000      = -t4
    -2.000      7.000      4.000      = obj
```

Reading off of this tableau, $x = -2$ and $y = 0$, however by inspection of the above graph this is not in the constraint set.

5.

Pivoting as requested gives the following.

In [15]:
```
#Do second pivot:
b_3=pivot(b_2,4,2,indep_names,dep_names)
```

```
       t1      t4       -1
    0.500   0.500    8.000      = -x
    0.250   0.750    6.000     = -t2
    1.250   0.750   15.000     = -t3
   -0.250   0.250    5.000      = -y
   -0.250  -1.750  -31.000     = obj
```

reading off the tableau, $x = 8, y = 5$ and the objective function is 31.

4.

In the first tableau, the ratios are: $4, \frac{5}{2}, \frac{3}{5}, 1$, so we select the $\frac{3}{5}$ entry to pivot on, (3,1), 5

In the second tableau, the ratios are: $4, \frac{5}{2}, \frac{3}{5}, 0$, so we select the 0 entry to pivot on, (4,1), 3

In the third tableau, the ratios are: $4, \frac{5}{2}, -\frac{3}{5}, -1$, so we select the -1 entry to pivot on, (2,1), 2 because we have to pick a positive.

5.

In [17]:
```
#Define Problem Parameters
indep_names=["cookie","cake","brownie"]
dep_names=["t1","t2","t3"]
c_1=np.array([[2,4,3,250],[1,1.5,2,150],[1,.25,.5,50],[15,20,12,0]])
print_tableau(c_1,indep_names,dep_names)
```

```
   cookie    cake  brownie        -1
    2.000   4.000    3.000   250.000      = -t1
    1.000   1.500    2.000   150.000      = -t2
    1.000   0.250    0.500    50.000      = -t3
   15.000  20.000   12.000     0.000      = obj
```

Selecting cake as the pivot column (because it has the highest coefficient) means we have the ratios: $62.5, 100, 200$. We then select the 62.5 entry, $(1, 2) = 4$. Applying that pivot yields

In [18]:
```
#Do first pivot
c_2=pivot(c_1,1,2,indep_names,dep_names)
```

```
   cookie      t1  brownie        -1
    0.500   0.250    0.750    62.500    = -cake
    0.250  -0.375    0.875    56.250      = -t2
    0.875  -0.062    0.312    34.375      = -t3
    5.000  -5.000   -3.000 -1250.000      = obj
```

Nice, we then only have one positive coefficient in the objective function, so let's pivot on the first column, `cookie` . We compute the ratios yielding: $125, 225, 39.2857...$, we pivot on the 39 and change, $(3, 1)$ giving:

In [19]:
```
#Do second pivot:
c_3=pivot(c_2,3,1,indep_names,dep_names)
```

```
        t3        t1   brownie        -1
    -0.571     0.286     0.571    42.857    = -cake
    -0.286    -0.357     0.786    46.429      = -t2
     1.143    -0.071     0.357    39.286 = -cookie
    -5.714    -4.643    -4.786 -1446.429      = obj
```

We have solved the question and see that the maximmum revenue would be $\$1446.429$ which is attained with 0 brownies and zero slack in the first and third constraints meaning $cake = 42.857$ and $cookie = 39.286$.

5.

a.

Let's first right down all of the constraints in their most basic form. Denote the weight of mixture i as m_i. The constraints would then be:

$$\begin{aligned} m_3 &\geq 2m_2 \\ m_2 &\geq 2m_1 \\ m_1 + .8m_2 + .6m_3 &\leq 500 \\ 0m_1 + .15m_2 + .3m_3 &\leq 250 \\ 0m_1 + .05m_2 + .1m_3 &\leq 100 \\ m_1, m_2, m_3 &\geq 0 \end{aligned}$$

Converting this to canonical max gives $\max_{m_1,m_2,m_3} 2m_1 + 1.5m_2 + m_3$ such that:

$$\begin{aligned} 0m_1 + 2m_2 - m_3 &\leq 0 \\ 2m_1 - m_2 + 0m_3 &\leq 0 \\ m_1 + .8m_2 + .6m_3 &\leq 500 \\ 0m_1 + .15m_2 + .3m_3 &\leq 250 \\ 0m_1 + .05m_2 + .1m_3 &\leq 100 \\ m_1, m_2, m_3 &\geq 0 \end{aligned}$$

And then as a tableau:

In [41]:
```
#Define Problem Parameters
indep_names=["m_1","m_2","m_3"]
dep_names=["t1","t2","t3","t4","t5"]
d_1=np.array([[0,2,-1,0],[2,-1,0,0],[1,.8,.6,500],[0,.15,.3,250],[0,.05,.1,100],
print_tableau(d_1,indep_names,dep_names)
```

```
       m_1       m_2       m_3        -1
     0.000     2.000    -1.000     0.000      = -t1
     2.000    -1.000     0.000     0.000      = -t2
     1.000     0.800     0.600   500.000      = -t3
     0.000     0.150     0.300   250.000      = -t4
     0.000     0.050     0.100   100.000      = -t5
     2.000     1.500     1.000     0.000      = obj
```

b.

First, I select to pivot on the m_1 column. As such the ratios necessary are: $\frac{0}{0}, \frac{0}{2}, 500, \frac{250}{0}, \frac{250}{0}$. Well that's a little funky. Let's pivot on the $(2,1) = 0$ entry giving:

In [42]:
```
#first pivot
d_2=pivot(d_1,2,1,indep_names,dep_names)
```

```
         t2        m_2        m_3         -1
     -0.000      2.000     -1.000      0.000      = -t1
      0.500     -0.500      0.000      0.000     = -m_1
     -0.500      1.300      0.600    500.000      = -t3
     -0.000      0.150      0.300    250.000      = -t4
     -0.000      0.050      0.100    100.000      = -t5
     -1.000      2.500      1.000      0.000      = obj
```

I choose next to target $c_2 = 2.5$ because it is the largest. The ratios in that column are: $\frac{0}{2}, \frac{0}{-.5}, \frac{500}{1.3}, \frac{250}{.3}, 1000$. The smallest non-negative ratio is at $(1,2) = 2$. Pivoting there yields:

In [43]:
```
#Pivot Number Two
d_3=pivot(d_2,1,2,indep_names,dep_names)
```

```
         t2         t1        m_3         -1
     -0.000      0.500     -0.500      0.000     = -m_2
      0.500      0.250     -0.250      0.000     = -m_1
     -0.500     -0.650      1.250    500.000      = -t3
      0.000     -0.075      0.375    250.000      = -t4
      0.000     -0.025      0.125    100.000      = -t5
     -1.000     -1.250      2.250      0.000      = obj
```

Targeting $c_3 = 2.25$, the non-negative ratios are: $400, 666\frac{2}{3}, 800$. Choosing the smallest, we must pivot on the $(3,3) = 1.25$. Doing that gives:

In [44]:
```
#Third and final pivot:
d_4=pivot(d_3,3,3,indep_names,dep_names)
```

```
         t2         t1         t3         -1
     -0.200      0.240      0.400    200.000     = -m_2
      0.400      0.120      0.200    100.000     = -m_1
     -0.400     -0.520      0.800    400.000     = -m_3
      0.150      0.120     -0.300    100.000      = -t4
      0.050      0.040     -0.100     50.000      = -t5
     -0.100     -0.080     -1.800   -900.000      = obj
```

Okay nice, all of the coefficients in the objective function are now negative giving us the three constraints will we will acheive. That is, we set t_1, t_2, t_3 to zero and as such set $m_1 = 100, m_2 = 200, m_3 = 300$ with 100 units of slack in the 4th constraint and 50 units of slack in the 5th constraint. That is, we will retain 100 pounds of cashews and 50 pounds of pecans, with zero pounds of peanuts remaining. Maximal profit it $\$900$.