Name: Manan Lohia

1a.i We have parameters

p_j : selling price of j th product $(j = 1, 2 \ldots n)$
a_{ij} : no of units of i th raw material required for j th product $(i = 1, 2 \ldots m)$
c_i = cost per unit of i th raw material
B = Total budget to procure raw materials

Decision variables

x_j = production quantity of j th product
y_i = procurement quantity of i th raw material

∴ Revenue from selling the products

$$R = \sum_{j=1}^{n} p_j x_j$$

Cost of procurement

$$C = \sum_{i=1}^{m} c_i y_i$$

∴ The objective to maximize profit is given by

$$\max \sum_{j=1}^{n} p_j x_j - \sum_{i=1}^{m} c_i y_i$$

Since B is the budget for procurement, we have

$$\sum_{i=1}^{m} c_i y_i \leq B$$

Furthermore, x_j units of product j requires a_{ij} units of ith raw material

$$\therefore \; y_i = \sum_{j=1}^{n} a_{ij} x_j$$

$\therefore$ The linear programming model is given by

$$\max \sum_{j=1}^{n} p_j x_j - \sum_{i=1}^{m} c_i y_i$$

$$\text{s.t} \quad \sum_{i=1}^{m} c_i y_i \leq B$$

$$y_i = \sum_{j=1}^{n} a_{ij} x_j, \quad i = 1, 2 \ldots m$$

$$x_j \geq 0 \;\; \forall \; j = 1, 2 \ldots n$$

$$y_i \geq 0 \;\; \forall \; i = 1, 2 \ldots m$$

ii In matrix form,

$$\max \; p' x - c' y$$

$$\text{s.t} \quad c' y \leq B$$

$$A x = y$$

$$x, y \geq 0$$

In the standard form, we include slack variable s_1

$\therefore$ we get

$$-\min \sum_{i=1}^{m} c_i y_i - \sum_{j=1}^{n} p_j x_j$$

$$\text{s.t} \quad \sum_{i=1}^{m} c_i y_i + s_1 = B$$

$$y_i = \sum_{j=1}^{n} a_{ij} x_j, \quad i = 1, 2 \ldots m$$

$$x_j, y_i, s_1 \geq 0 \quad \forall \; i, j$$

Or, we include slack vector $s = [s_1]$

$$-\min \; c'y - p'x$$

$$\text{s.t} \quad [c' \; I] \begin{bmatrix} y \\ s \end{bmatrix} = B$$

$$Ax = y$$

$$x, y, s \geq 0$$

1.b The same problem without using procurement decisions c_i can be modelled as

$$-\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_i a_{ij} x_j - \sum_{j=1}^{n} p_j x_j$$

$$\text{s.t} \sum_{i=1}^{m} \sum_{j=1}^{n} c_i a_{ij} x_j + s_1 = B$$

$$x_j, s_1 \geq 0 \ \forall j$$

or

$$-\min (c' A - p') x$$

$$\text{s.t} \begin{bmatrix} c' & I \end{bmatrix} \begin{bmatrix} Ax \\ s \end{bmatrix} = B$$

$$x, s \geq 0$$

2. a

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix}$$

Applying $R_3 \to R_3 - (R_1 + R_2)$

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$\therefore$ Rank of the matrix $= 2$

b We have

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}_{m \times 1} \text{ and } y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}_{n \times 1}$$

$\therefore A = xy'$

$$= \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} y'$$

$$= \begin{bmatrix} x_1 y' \\ x_2 y' \\ x_3 y' \\ \vdots \\ x_m y' \end{bmatrix}$$

Hence, the matrix consists of m rows, which are all scalar multiples of the vector y'

Alternatively, it can also be viewed as consisting of n columns, which are all scalar multiples of x

∴ There is only 1 linearly independent vector in the row space or column space of A

∴ $\underline{\text{rank}(A) = 1}$

c We have $A = [a_{ij}] = \begin{bmatrix} 1 & 0 & 1 & - & - \\ - & - & 0 & - & - \\ 1 & - & - & 0 & 1 \end{bmatrix}$

Since we know that A is of rank 1, the rows of A are linearly dependent.

Since $a_{11}/a_{31} = 1/1 = 1$, this implies that the first and third rows are the same. Furthermore, $a_{23}/a_{13} = 0/1 = 0$. Hence, the second row is a null vector

$$\therefore A = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

3.a Since the feasible set is consists of equations which are linear in the unknown variable x & y, it forms a polyhedron.

For $a = 1$, we have, $a^2 = 1$, $\sqrt{a} = \pm 1$

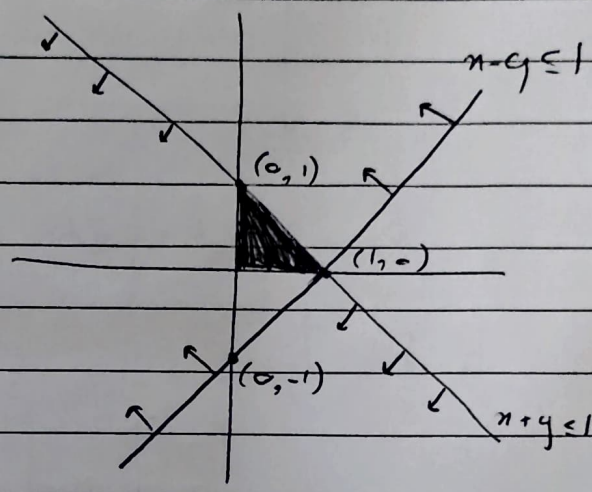
$\therefore$ The constraints are

$x + y \leq 1$

$x - y \leq 1$

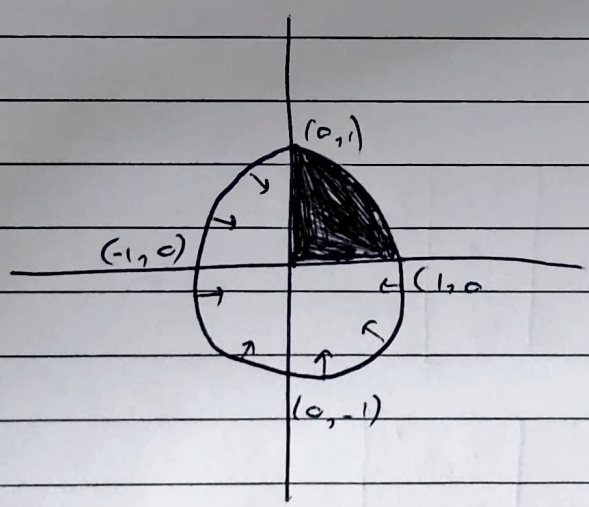
$x \geq 0$

$y \geq 0$



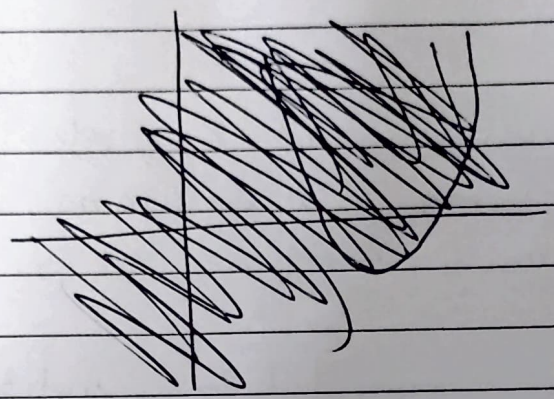
The shaded region is the feasible area, which suggests

b



The shaded area gives the feasible area. However, since the constraint are not linear, it does not form a polyhedron

c The feasible region is given by



$x \in [3, 5]$

or $x \geq 3$ and $x \leq 5$

Since there are linear constraints, the region forms a polyhedron

4 We have a total of 4 shifts, each of length 4 hours

Let x_i be the number of full time clerks coming to work in the ith shift. Note that $x_4 = 0$ as full time clerks come to work in only 3 shifts

Similarly, let y_i be the number of part time clerks in ith shift

$\therefore$ The total cost is

$$C = (40 \times 8)\left(\sum_{i=1}^{3} x_i\right) + (30 \times 4)\left(\sum_{i=1}^{4} y_i\right)$$

or $$C = 320 \sum_{i=1}^{3} x_i + 120 \sum_{i=1}^{4} y_i$$

The constraints for minimum number of clerks for each shift are formulated as

$$x_1 + y_1 \geq 4$$
$$x_1 + x_2 + y_2 \geq 8$$
$$x_2 + x_3 + y_3 \geq 10$$
$$x_3 + y_4 \geq 6$$

Additionally, no. of full time clerks in a shift cannot be strictly less than the no. of part time clerks

$$x_1 \geq y_1$$
$$x_1 + x_2 \geq y_2$$
$$x_2 + x_3 \geq y_3$$
$$x_3 \geq y_4$$

The linear programming problem is

$$\min \; 320 \sum_{i=1}^{3} x_i + 120 \sum_{i=1}^{4} y_i$$

s.t
$$x_1 + y_1 \geq 4$$
$$x_1 + x_2 + y_2 \geq 8$$
$$x_2 + x_3 + y_3 \geq 10$$
$$x_3 + y_4 \geq 6$$
$$x_1 \geq y_1$$
$$x_1 + x_2 \geq y_2$$
$$x_2 + x_3 \geq y_3$$
$$x_3 \geq y_4$$
$$x_i, y_i \geq 0 \quad \forall i$$

5 We introduce three new decision variables

$z_1 = |x_2 - 10|$

$z_2 = |x_1 + 2|$

$z_3 = |x_2|$

Hence, the linear programming problem is

$$\min \; 2x_1 + 3z_1$$

$$\text{s.t} \quad x_2 - 10 \leq z_1$$

$$10 - x_2 \leq z_1$$

$$z_2 + z_3 \leq 5$$

$$x_1 + x_2 \leq z_2$$

$$x_1 + x_2 \geq z_2$$

$$x_2 \geq z_3$$

$$x_2 \leq z_3$$

$$z_1, z_2, z_3 \geq 0$$

6

6.a) We have

s_i : no. of shares of ith stock in portfolio
p_i : purchase price of ith stock
q_i : current price of ith stock
r_i : expected price of ith stock

Let x_i be the no. of shares of ith stock the investor must sell.

Further, define a_i as

$$a_i = \begin{cases} 1, & q_i > p_i \\ 0, & q_i \leq p_i \end{cases}$$

We assign 0 for the equality sign as he does not pay capital gain tax if $p_i = q_i$

$\therefore$ Expected value of portfolio

$$E = \sum (s_i - x_i) r_i$$

Amount of money raised

$$A = \sum_{i=1}^{n} 0.99\, x_i q_i - \sum_{i=1}^{n} 0.3\, a_i x_i (q_i - p_i)$$

Since a share can't be sold partially, we limit our solution to positive integers.

Assuming that sharing is not allowed,

$$\max \sum_{i=1}^{n} (s_i - x_i) r_i$$

s.t $\quad x_i \geq s_i \quad \forall i$

$$0.99 \sum_{i=1}^{n} x_i q_i - 0.3 \sum_{i=1}^{n} a_i x_i (q_i - p_i) \geq k$$

$$a_i = \begin{cases} 1, & q_i > p_i \\ 0, & q_i \leq p_i \end{cases}$$

$x_i \geq 0 \ \forall i$

$x_i \in \mathbb{Z}_+ \ \forall i$

Assignment 1

August 31, 2022

[9]:
```
import pandas as pd
import gurobipy as gp
import numpy as np

#Import the data
data = pd.read_csv('investment.csv', index_col = 0)

#Create the column corresponding to the flag variable a(i)
data['flag'] = np.where(data['Current Price'] > data['Buying Price'], 1, 0)

#Define Necessary Variables
K = 9000
M, N = data.shape

data
```

[9]:

Stock	Buying Price	Current Share	Current Price	Expected Future Price	flag
S1	1.2	1000	2.1	2.0	1
S2	2.1	1000	3.2	3.7	1
S3	3.2	1000	4.1	5.2	1
S4	4.1	1000	5.1	7.1	1
S5	4.5	1000	6.7	9.1	1

[10]:
```
#Slice data to get necessary tables to build the objective function and the␣
↪constraints

# Table to calculate objective function
ExpectedPortfolioValue = data[['Current Share', 'Expected Future Price']]

# Table to calculate constraint for total money to be raised
MoneyRaised = data[['Current Price','Buying Price', 'flag']]

# Table to calculate constraints on max number of shares that can be sold
MaxShares = data['Current Share']
```

[11]:
```
# Setting up the model

OptimalPortfolio = gp.Model("Portfolio")

# Creating the variables
x = OptimalPortfolio.addVars(M, name = 'SharesSold', vtype = gp.GRB.INTEGER)

# Creating the objective
OptimalPortfolio.setObjective(gp.quicksum(
                              (ExpectedPortfolioValue.iloc[i,0] - x[i]) *␣
 ↪ExpectedPortfolioValue.iloc[i,1] for i in range(M)),
                               gp.GRB.MAXIMIZE)

# Adding Constraints
OptimalPortfolio.addConstrs(x[i] <= MaxShares[i] for i in range(M))
OptimalPortfolio.addConstr(0.99 * gp.quicksum(MoneyRaised.iloc[i,0] * x[i] for␣
 ↪i in range(M)) -
                           0.30 * gp.quicksum(MoneyRaised.iloc[i,2] *␣
 ↪(MoneyRaised.iloc[i,0] - MoneyRaised.iloc[i, 1]) * x[i] for i in range(M))
                           >= K)

# Solving the model

OptimalPortfolio.optimize()
```

```
Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (win64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 6 rows, 5 columns and 10 nonzeros
Model fingerprint: 0x9285a694
Variable types: 0 continuous, 5 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 6e+00]
  Objective range  [2e+00, 9e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+03, 9e+03]
Presolve removed 5 rows and 0 columns
Presolve time: 0.00s
Presolved: 1 rows, 5 columns, 5 nonzeros
Variable types: 0 continuous, 5 integer (0 binary)
Found heuristic solution: objective 14199.200000

Root relaxation: objective 1.535679e+04, 1 iterations, 0.00 seconds (0.00 work
units)

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

```
     0     0 15356.7909    0    1 14199.2000 15356.7909  8.15%     -    0s
H    0     0                    15355.100000 15356.7909  0.01%     -    0s

Explored 1 nodes (1 simplex iterations) in 0.10 seconds (0.00 work units)
Thread count was 8 (of 8 available processors)

Solution count 2: 15355.1 14199.2

Optimal solution found (tolerance 1.00e-04)
Best objective 1.535510000000e+04, best bound 1.535660000000e+04, gap 0.0098%
```

[12]:
```python
for v in OptimalPortfolio.getVars():
    print (v.Varname, v.x)

print("Max Portfolio Value: ", OptimalPortfolio.objVal)
```

```
SharesSold[0] 999.0
SharesSold[1] 1000.0
SharesSold[2] 1000.0
SharesSold[3] 118.0
SharesSold[4] 1.0
Max Portfolio Value:  15355.1
```