MGMTMSA 403: Optimization

Assignment 2: Portfolio Optimization

Due on BruinLearn by 11:59pm on January 31st.

Arnav Garg (906310841)
Oscar Trumpy (706310371)
Facundo Klappenbach (306316186)

Background

The file Prices.csv contains monthly prices for 390 stocks, collected over a 5-year period. Each column of the file corresponds to one stock. The first row of the file contains the ticker of each stock, which is a 3-4 letter that identifies the company. For example, the ticker for Microsoft is 'MSFT'.

This assignment will require you to formulate and solve three different portfolio optimization models. In order to formulate the models, you will first have to compute the monthly returns for each stock, as well as the covariance matrix. Let t = 1, . . . , 60 index the months over the 5-year period, and let Price_t be the price of a stock in period t. The average monthly return of the stock (in %) in period t can then be calculated as

$$Return_t = \frac{Price_{t+1} - Price_t}{Price_t} \times 100\% \tag{1}$$

Note that the covariance of a matrix M can be calculated using the function numpy.cov(M) from the NumPy package.

HINT: You may use the Assignment 2 Preprocessing file posted on BruinLearn to compute the average returns and covariance information.

In [ ]:
```
###############################################################################################
# Data Pre-Processing
```

```python
#################################################################################################
# Import gurobi and numpy
from gurobipy import *
import numpy as np
from numpy import genfromtxt
import csv

## Get index of 4 tickers
tick4 = ["MSFT","GS","PG","SCHP"];

# Get variable names
with open('Prices.csv') as csvFile:
    reader = csv.reader(csvFile)
    tickers = next(reader) ## stores the tickers of all 390 stocks

tickind =[];
for t in tick4:
    tickind.append(tickers.index(t)) ## retrieve index that corresponds to each ticker

# Load data
prices = genfromtxt('Prices.csv', delimiter = ',', skip_header = 1)

# get dimensions of data
d = prices.shape[0]
n = prices.shape[1]

# calculate monthly returns of each stock
returns = np.zeros((d - 1, n))
for stock in range(n):
    for month in range(d - 1):
        returns[month, stock] = prices[month + 1, stock] / prices[month, stock] - 1

# Store average return (parameter r_i in portfolio optimization model)
avg_return = np.zeros(n)
avg_return = np.mean(returns, axis = 0)

# Store covariance matrix (parameter C_ij in portfolio optimization model)
C = np.zeros((n, n))
C = np.cov(np.transpose(returns))
#################################################################################################
```

Models

The description of each model is given below.

Model 1. Start by focusing on a four-asset portfolio: Suppose you can only invest in Microsoft (MSFT), Goldman Sachs (GS), Proctor & Gamble (PG), and U.S. Treasury Bonds (SCHP). Construct a minimum-variance portfolio with an expected monthly return of at least 0.5%.

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{4}\sum_{j=1}^{4} w_i \times w_j \times C_{ij} && (2)\\
\text{s.t.} \quad & \sum_{i=1}^{4} w_i \times r_i \geq 0.5 && (3)\\
& \sum_{i=1}^{4} w_i = 1 && (4)\\
& w_i \geq 0 \quad i = 1 \ldots 4 && (5)
\end{aligned}$$

In [ ]:

```
##############################################################################################
# Initialize the Model 1
mod1 = Model()

##############################################################################################
# Define Variables
# Defining the weights variable
w1 = mod1.addVars(len(tickers))

##############################################################################################
# Define Constraints
## Expected return should atleast be 0.5
mod1.addConstr(sum(w1[i] *  avg_return[i] for i in tickind) >= 0.005)

## Weights add up to 1
mod1.addConstr(sum(w1[i] for i in tickind) == 1)

## w_i >= 0
for i in tickind:
    mod1.addConstr(w1[i] >= 0)

##############################################################################################
# Construct Objective
mod1.setObjective(sum((w1[i] * w1[j] * C[i, j]) for i in tickind for j in tickind), GRB.MINIMIZE)

##############################################################################################
# Update and solve
mod1.update()
mod1.optimize()

##############################################################################################
# Print Optimal Solution
print("Optimal Value:", mod1.objVal)

##############################################################################################
```

```
Set parameter Username
Academic license - for non-commercial use only - expires 2025-01-10
Gurobi Optimizer version 11.0.0 build v11.0.0rc2 (mac64[arm] - Darwin 23.2.0 23C71)

CPU model: Apple M1
```

```
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 6 rows, 390 columns and 12 nonzeros
Model fingerprint: 0x24944e33
Model has 10 quadratic objective terms
Coefficient statistics:
  Matrix range     [2e-04, 1e+00]
  Objective range  [0e+00, 0e+00]
  QObjective range [5e-05, 7e-03]
  Bounds range     [0e+00, 0e+00]
  RHS range        [5e-03, 1e+00]
Presolve removed 4 rows and 386 columns
Presolve time: 0.04s
Presolved: 2 rows, 4 columns, 8 nonzeros
Presolved model has 10 quadratic objective terms
Ordering time: 0.00s

Barrier statistics:
 Free vars  : 3
 AA' NZ     : 1.000e+01
 Factor NZ  : 1.500e+01
 Factor Ops : 5.500e+01 (less than 1 second per iteration)
 Threads    : 1

                  Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl     Time
   0   7.75648617e+03 -7.75648617e+03  3.50e+03 1.57e-06  1.00e+06     0s
   1   2.51636603e+02 -2.59893243e+02  2.31e+02 1.04e-07  6.92e+04     0s
   2   2.61043362e-03 -1.01272125e+01  6.64e-01 2.99e-10  3.29e+02     0s
   3   1.01144644e-03 -3.53587425e+00  6.64e-07 2.96e-16  4.53e+01     0s
   4   1.00956586e-03 -4.19494040e-03  3.14e-10 1.13e-17  6.66e-02     0s
   5   4.34187327e-04 -3.61737038e-05  6.69e-12 2.78e-18  6.02e-03     0s
   6   2.14699172e-04  1.27154313e-04  3.33e-16 3.48e-17  1.12e-03     0s
   7   1.80089010e-04  1.72718190e-04  6.94e-17 8.52e-18  9.43e-05     0s
   8   1.77537641e-04  1.76950183e-04  1.55e-15 3.33e-18  7.52e-06     0s
   9   1.77493374e-04  1.77486115e-04  2.44e-15 1.52e-17  9.29e-08     0s
  10   1.77493264e-04  1.77493257e-04  6.94e-15 1.39e-17  9.32e-11     0s

Barrier solved model in 10 iterations and 0.05 seconds (0.00 work units)
Optimal objective 1.77493264e-04
Optimal Value: 0.00017749326422824248
```

Model 2. Now suppose you can invest in all 390 stocks. Construct a minimum-variance portfolio with an expected monthly return of at least 0.5%.

$$\text{minimize} \quad \sum_{i=1}^{390} \sum_{j=1}^{390} w_i \times w_j \times C_{ij} \tag{6}$$

$$\text{s.t.} \quad \sum_{i=1}^{390} w_i \times r_i \geq 0.5 \tag{7}$$

$$\sum_{i=1}^{390} w_i = 1 \tag{8}$$

$$w_i \geq 0 \quad i = 1 \ldots 390 \tag{9}$$

```
In [ ]:
##############################################################################################
# Initialize the Model 2
mod2 = Model()

##############################################################################################
# Define Variables
# Defining the weights variable
w2 = mod2.addVars(len(tickers))

##############################################################################################
# Define Constraints
## Expected return should atleast be 0.5
mod2.addConstr(sum(w2[i] *  avg_return[i] for i in range(len(tickers))) >= 0.005)

## Weights add up to 1
mod2.addConstr(sum(w2[i] for i in range(len(tickers))) == 1)

## w_i >= 0
for i in range(len(tickers)):
    mod2.addConstr(w2[i] >= 0)

##############################################################################################
# Construct Objective
mod2.setObjective(sum((w2[i] * w2[j] * C[i, j]) for i in range(len(tickers)) for j in range(len(tickers))), GRB.M

##############################################################################################
# Update and solve
mod2.update()
mod2.optimize()

##############################################################################################
# Print Optimal Solution
print("Optimal Value:", mod2.objVal)

##############################################################################################
```

```
Gurobi Optimizer version 11.0.0 build v11.0.0rc2 (mac64[arm] - Darwin 23.2.0 23C71)

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads
```

```
Optimize a model with 392 rows, 390 columns and 1170 nonzeros
Model fingerprint: 0xd3a53ccf
Model has 76245 quadratic objective terms
Coefficient statistics:
  Matrix range     [1e-06, 1e+00]
  Objective range  [0e+00, 0e+00]
  QObjective range [2e-07, 8e-02]
  Bounds range     [0e+00, 0e+00]
  RHS range        [5e-03, 1e+00]

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 392 rows, 390 columns and 1170 nonzeros
Model fingerprint: 0xd3a53ccf
Model has 76245 quadratic objective terms
Coefficient statistics:
  Matrix range     [1e-06, 1e+00]
  Objective range  [0e+00, 0e+00]
  QObjective range [2e-07, 8e-02]
  Bounds range     [0e+00, 0e+00]
  RHS range        [5e-03, 1e+00]
Presolve removed 390 rows and 0 columns
Presolve time: 0.01s
Presolved: 2 rows, 390 columns, 780 nonzeros
Presolved model has 76245 quadratic objective terms
Ordering time: 0.00s

Barrier statistics:
 Free vars  : 59
 AA' NZ     : 1.830e+03
 Factor NZ  : 1.891e+03
 Factor Ops : 7.753e+04 (less than 1 second per iteration)
 Threads    : 8

                  Objective                Residual
Iter       Primal          Dual         Primal    Dual     Compl     Time
   0   1.10520633e-17 -1.10520633e-17  3.90e+05 1.86e-08  1.00e+06     0s
   1   1.32403095e+02 -1.40151246e+02  1.14e+04 5.42e-10  2.92e+04     0s
   2   1.53493902e+00 -9.57482645e+00  1.02e+02 4.86e-12  2.67e+02     0s
   3   1.17587702e-03 -7.92081818e+00  1.02e-04 9.12e-16  2.59e+00     0s
   4   1.17509605e-03 -1.01300882e-02  4.36e-08 1.20e-17  3.70e-03     0s
   5   8.22186010e-04 -1.13464664e-03  4.77e-09 1.18e-17  6.41e-04     0s
   6   3.78013352e-04 -5.97431608e-04  5.77e-15 2.22e-16  3.19e-04     0s
   7   2.05319383e-04 -1.66218904e-04  1.33e-15 8.33e-17  1.22e-04     0s
   8   9.53707757e-05 -4.26667932e-05  3.55e-15 8.33e-17  4.52e-05     0s
   9   6.93295928e-05  3.06840100e-06  1.33e-15 4.16e-17  2.17e-05     0s
  10   4.16952119e-05  2.32922123e-05  2.00e-15 2.08e-17  6.02e-06     0s
  11   3.39225828e-05  2.74051911e-05  8.88e-16 2.08e-17  2.13e-06     0s
  12   3.04655599e-05  2.85032903e-05  3.44e-15 2.78e-17  6.42e-07     0s
  13   2.89580888e-05  2.87642150e-05  1.24e-14 2.08e-17  6.35e-08     0s
  14   2.87917525e-05  2.87842956e-05  2.15e-14 3.82e-17  2.44e-09     0s

Barrier solved model in 14 iterations and 0.04 seconds (0.01 work units)
Optimal objective 2.87917525e-05

Optimal Value: 2.879175251400085e-05
```

Model 3. In practice, there are transaction fees associated with buying stocks. One way of keeping transaction fees low while still attaining desirable performance is to limit the total number of stocks that are purchased (i.e. limit the number of stocks that have a strictly positive weight). Construct a minimum-variance portfolio that selects at most 4 of the 390 stocks, and has an expected monthly return of at least 0.5%. (Note: By introducing binary variables into a quadratic program, we obtain a **quadratic integer program.** Fortunately, this particular quadratic integer program can be solved by Gurobi.)

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{390}\sum_{j=1}^{390} w_i \times w_j \times C_{ij} && (10)\\
\text{s.t.} \quad & X_i \in \{0,1\} \quad i = 1\ldots390 && (11)\\
& \sum_{i=1}^{390} X_i <= 4 && (12)\\
& \sum_{i=1}^{390} w_i \times r_i \geq 0.5 && (13)\\
& \sum_{i=1}^{390} w_i = 1 && (14)\\
& w_i <= X_i \quad i = 1\ldots390 && (15)\\
& w_i \geq 0 \quad i = 1\ldots390 && (16)
\end{aligned}$$

In [ ]:

```
##############################################################################################
# Initialize the Model 3
mod3 = Model()

##############################################################################################
# Define Variables
# Defining the weights variable
w3 = mod3.addVars(len(tickers))
# Dfeining the binary variable for choice
x = mod3.addVars(len(tickers), vtype = GRB.BINARY)

##############################################################################################
# Define Constraints
## Expected return should atleast be 0.5
mod3.addConstr(sum(w3[i] *  avg_return[i] for i in range(len(tickers))) >= 0.005)

## Weights add up to 1
mod3.addConstr(sum(w3[i] for i in range(len(tickers))) == 1)

## w_i >= 0
for i in range(len(tickers)):
    mod3.addConstr(w3[i] >= 0)

## Sum of X_i <= 4
mod3.addConstr(sum(x[i] for i in range(len(tickers))) <= 4)

## Weights should be lessthan binary variable
for i in range(len(tickers)):
    mod3.addConstr(w3[i] <= x[i])

##############################################################################################
# Construct Objective
mod3.setObjective(sum((w3[i] * w3[j] * C[i, j]) for i in range(len(tickers)) for j in range(len(tickers))), GRB.

##############################################################################################
# Update and solve
mod3.update()
mod3.optimize()
```

```python
##########################################################################################
# Print Optimal Solution
print("Optimal Value:", mod3.objVal)

##########################################################################################
```

```
Gurobi Optimizer version 11.0.0 build v11.0.0rc2 (mac64[arm] - Darwin 23.2.0 23C71)

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 783 rows, 780 columns and 2340 nonzeros
Model fingerprint: 0x6a25f44d
Model has 76245 quadratic objective terms
Variable types: 390 continuous, 390 integer (390 binary)
Coefficient statistics:
  Matrix range     [1e-06, 1e+00]
  Objective range  [0e+00, 0e+00]
  QObjective range [2e-07, 8e-02]
  Bounds range     [1e+00, 1e+00]
  RHS range        [5e-03, 4e+00]
Found heuristic solution: objective 0.0136011

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 783 rows, 780 columns and 2340 nonzeros
Model fingerprint: 0x6a25f44d
Model has 76245 quadratic objective terms
Variable types: 390 continuous, 390 integer (390 binary)
Coefficient statistics:
  Matrix range     [1e-06, 1e+00]
  Objective range  [0e+00, 0e+00]
  QObjective range [2e-07, 8e-02]
  Bounds range     [1e+00, 1e+00]
  RHS range        [5e-03, 4e+00]
Found heuristic solution: objective 0.0136011
Presolve removed 390 rows and 0 columns
Presolve time: 0.02s
Presolved: 393 rows, 780 columns, 1950 nonzeros
Presolved model has 76245 quadratic objective terms
```

```
Variable types: 390 continuous, 390 integer (390 binary)

Root relaxation: objective 2.878501e-05, 129 iterations, 0.01 seconds (0.01 work units)

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

     0     0    0.00003    0   21    0.01360    0.00003   100%     -    0s
H    0     0                       0.0001377    0.00003  79.1%     -    0s
H    0     0                       0.0001357    0.00003  78.8%     -    0s
     0     0    0.00003    0   21    0.00014    0.00003  78.8%     -    0s
H    0     0                       0.0000744    0.00003  61.3%     -    0s
H    0     0                       0.0000723    0.00003  58.1%     -    0s
     0     0    0.00003    0   21    0.00007    0.00003  58.1%     -    0s
     0     0    0.00003    0   21    0.00007    0.00003  57.7%     -    0s
     0     0    0.00003    0   21    0.00007    0.00003  57.5%     -    0s
     0     2    0.00003    0   21    0.00007    0.00003  57.5%     -    0s
H  249   171                       0.0000676    0.00003  54.6%  49.4    0s
H  252   171                       0.0000675    0.00003  54.5%  48.9    0s
  7881  2109    0.00006   51   20    0.00007    0.00005  31.4%  56.9    5s
 18201  2422     cutoff   33         0.00007    0.00006  12.0%  62.9   10s
 24751   752     cutoff   53         0.00007    0.00006  4.47%  83.9   15s

Explored 26833 nodes (2384781 simplex iterations) in 16.16 seconds (36.14 work units)
Thread count was 8 (of 8 available processors)

Solution count 7: 6.75347e-05 6.75857e-05 7.23026e-05 ... 0.0136011

Optimal solution found (tolerance 1.00e-04)
Best objective 6.753470760728e-05, best bound 6.753470760728e-05, gap 0.0000%
Optimal Value: 6.753470760728129e-05
```

Questions

1. Formulate and solve each of the three models in Python, and then answer the following questions. For each part, also include your Gurobi output.

a) For **Model 1**, write down the optimal risk (i.e. the optimal objective function value), solver time, and the weight on each of the four stocks.

```
In [ ]: print("Optimal Value:", round(mod1.objVal, 6))

        print("Solver Time:", round(mod1.Runtime, 6), "s")

        for i in tickind:
            print("Weight of index", i, ":", round(w1[i].x, 6))
```

```
Optimal Value: 0.000177
Solver Time: 0.053893 s
Weight of index 315 : 0.237118
Weight of index 216 : 0.02586
Weight of index 372 : 0.0
Weight of index 388 : 0.737023
```

b) For **Model 2**, write down the optimal risk and solver time.

```
In [ ]: print("Optimal Value:", round(mod2.objVal, 6))

        print("Solver Time:", round(mod2.Runtime, 6), "s")
```

```
Optimal Value: 2.9e-05
Solver Time: 0.042813 s
```

c) For **Model 3**, report the optimal risk, solver time, and the ticker and weight on each of the four stocks selected by the model.

```
In [ ]: print("Optimal Value:", round(mod3.objVal, 6))

        print("Solver Time:", round(mod3.Runtime, 6), "s")

        for i in range(len(tickers)):
            if x[i].x == 1:
                print("Weight of index", i, ":", round(w3[i].x, 6))
```

```
Optimal Value: 6.8e-05
Solver Time: 16.162996 s
Weight of index 118 : 0.126411
Weight of index 285 : 0.075476
Weight of index 348 : 0.043754
Weight of index 389 : 0.754359
```

2. Use your solution to Question 1 above to answer the following questions:

a) Is the optimal risk in **Model 1** higher or lower than **Model 2**? Explain why in 1-2 sentences.
The optimal risk in Model 2 is lower than the optimal risk in Model 1 because in Model 2 we are considering all 390 assets to create an optimal portfolio. In Model 1, we are limiting the scope of our optimisation problem to only 4 assets. Therefore, Model 2 is able to choose a better combination of assets which can give a lower optimal risk for the same return.

b) Is the optimal risk in **Model 2** higher or lower than **Model 3**? Explain why in 1-2 sentences.
The optimal risk in Model 3 is higher than the optimal risk in Model 2. Even though Model 3 finds the best combination of 4 assets with minimum variance, the scope of its optimisation is limited as compared to Model 2 because Model 2 is able to choose the best combination of assets out of all 390 assets. Therefore, Model 2 is able to choose a better combination of assets which can give a lower optimal risk for the same return.

3. In some cases, we may want to get an approximate solution quickly by terminating the branch- and-bound algorithm before it finds an optimal solution. There are two ways to terminate Gurobi early: (a) by setting a maximum time limit, and (b) by setting a maximum acceptable optimality gap (the tolerance). Use **Model 3** to answer the following two questions. For each part, also include your Gurobi output.

a) Set Gurobi to terminate after 3 seconds by including XYZ.Params.TimeLimit = 3.0 in your code for **Model 3**, where 'XYZ' is the name of your model. How does the objective function value at termination compare the optimal value obtained in question 1c)?

The objective function value at termination is exactly equal to the value obtained in question 1c).

In [ ]:
```
##############################################################################################
# Initialize the Model 3
mod3 = Model()
mod3.Params.TimeLimit = 3.0
```

```
###############################################################################################
# Define Variables
# Defining the weights variable
w3 = mod3.addVars(len(tickers))
# Dfeining the binary variable for choice
x = mod3.addVars(len(tickers), vtype = GRB.BINARY)

###############################################################################################
# Define Constraints
## Expected return should atleast be 0.5
mod3.addConstr(sum(w3[i] *  avg_return[i] for i in range(len(tickers))) >= 0.005)

## Weights add up to 1
mod3.addConstr(sum(w3[i] for i in range(len(tickers))) == 1)

## w_i >= 0
for i in range(len(tickers)):
    mod3.addConstr(w3[i] >= 0)

## Sum of X_i <= 4
mod3.addConstr(sum(x[i] for i in range(len(tickers))) <= 4)

## Weights should be lessthan binary variable
for i in range(len(tickers)):
    mod3.addConstr(w3[i] <= x[i])

###############################################################################################
# Construct Objective
mod3.setObjective(sum((w3[i] * w3[j] * C[i, j]) for i in range(len(tickers)) for j in range(len(tickers))), GRB.M

###############################################################################################
# Update and solve
mod3.update()
mod3.optimize()

###############################################################################################
# Print Optimal Solution
print("Optimal Value:", mod3.objVal)

###############################################################################################
```

```
Set parameter TimeLimit to value 3
```

```
Gurobi Optimizer version 11.0.0 build v11.0.0rc2 (mac64[arm] - Darwin 23.2.0 23C71)

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 783 rows, 780 columns and 2340 nonzeros
Model fingerprint: 0x6a25f44d
Model has 76245 quadratic objective terms
Variable types: 390 continuous, 390 integer (390 binary)
Coefficient statistics:
  Matrix range     [1e-06, 1e+00]
  Objective range  [0e+00, 0e+00]
  QObjective range [2e-07, 8e-02]
  Bounds range     [1e+00, 1e+00]
  RHS range        [5e-03, 4e+00]
Found heuristic solution: objective 0.0136011
Presolve removed 390 rows and 0 columns
Presolve time: 0.01s
Presolved: 393 rows, 780 columns, 1950 nonzeros
Presolved model has 76245 quadratic objective terms
Variable types: 390 continuous, 390 integer (390 binary)

Root relaxation: objective 2.878501e-05, 129 iterations, 0.00 seconds (0.01 work units)

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

     0     0    0.00003    0   21    0.01360    0.00003   100%     -    0s
H    0     0                       0.0001377    0.00003  79.1%     -    0s
H    0     0                       0.0001357    0.00003  78.8%     -    0s
     0     0    0.00003    0   21    0.00014    0.00003  78.8%     -    0s
H    0     0                       0.0000744    0.00003  61.3%     -    0s
H    0     0                       0.0000723    0.00003  58.1%     -    0s
     0     0    0.00003    0   21    0.00007    0.00003  58.1%     -    0s
     0     0    0.00003    0   21    0.00007    0.00003  57.7%     -    0s
     0     0    0.00003    0   21    0.00007    0.00003  57.5%     -    0s
     0     2    0.00003    0   21    0.00007    0.00003  57.5%     -    0s
H  249   171                       0.0000676    0.00003  54.6%  49.4    0s
H  252   171                       0.0000675    0.00003  54.5%  48.9    0s

Explored 4103 nodes (193197 simplex iterations) in 3.00 seconds (5.24 work units)
```

```
Thread count was 8 (of 8 available processors)

Solution count 7: 6.75347e-05 6.75857e-05 7.23026e-05 ... 0.0136011

Time limit reached
Best objective 6.753470760728e-05, best bound 4.031736023886e-05, gap 40.3013%
Optimal Value: 6.753470760728129e-05
```

b) Set Gurobi to terminate after reaching a gap of 10% by including XYZ.Params.MIPGap = 0.1 in your code for **Model 3**, where 'XYZ' is the name of your model. (Note: The default gap in Gurobi is 0.0001.) How does the solver time compare with the solution time obtained in question 1c)?

The solver time after reaching a gap of 10% is shorter than the solver time obtained in question 1c).

In [ ]:

```
##############################################################################################
# Initialize the Model 3
mod3 = Model()
mod3.Params.MIPGap = 0.1
##############################################################################################
# Define Variables
# Defining the weights variable
w3 = mod3.addVars(len(tickers))
# Dfeining the binary variable for choice
x = mod3.addVars(len(tickers), vtype = GRB.BINARY)

##############################################################################################
# Define Constraints
## Expected return should atleast be 0.5
mod3.addConstr(sum(w3[i] *  avg_return[i] for i in range(len(tickers))) >= 0.005)

## Weights add up to 1
mod3.addConstr(sum(w3[i] for i in range(len(tickers))) == 1)

## w_i >= 0
for i in range(len(tickers)):
    mod3.addConstr(w3[i] >= 0)

## Sum of X_i <= 4
mod3.addConstr(sum(x[i] for i in range(len(tickers))) <= 4)
```

```python
## Weights should be lessthan binary variable
for i in range(len(tickers)):
    mod3.addConstr(w3[i] <= x[i])

###############################################################################################
# Construct Objective
mod3.setObjective(sum((w3[i] * w3[j] * C[i, j]) for i in range(len(tickers)) for j in range(len(tickers))), GRB.M

###############################################################################################
# Update and solve
mod3.update()
mod3.optimize()

###############################################################################################
# Print Optimal Solution
print("Solver Time:", mod3.Runtime)

###############################################################################################
```

```
Set parameter MIPGap to value 0.1
Gurobi Optimizer version 11.0.0 build v11.0.0rc2 (mac64[arm] - Darwin 23.2.0 23C71)

CPU model: Apple M1
Thread count: 8 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 783 rows, 780 columns and 2340 nonzeros
Model fingerprint: 0x6a25f44d
Model has 76245 quadratic objective terms
Variable types: 390 continuous, 390 integer (390 binary)
Coefficient statistics:
  Matrix range     [1e-06, 1e+00]
  Objective range  [0e+00, 0e+00]
  QObjective range [2e-07, 8e-02]
  Bounds range     [1e+00, 1e+00]
  RHS range        [5e-03, 4e+00]
Found heuristic solution: objective 0.0136011
Presolve removed 390 rows and 0 columns
Presolve time: 0.02s
Presolved: 393 rows, 780 columns, 1950 nonzeros
Presolved model has 76245 quadratic objective terms
Variable types: 390 continuous, 390 integer (390 binary)
```

```
Root relaxation: objective 2.878501e-05, 129 iterations, 0.00 seconds (0.01 work units)

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

     0     0    0.00003    0   21    0.01360    0.00003   100%     -    0s
H    0     0                       0.0001377    0.00003  79.1%     -    0s
H    0     0                       0.0001357    0.00003  78.8%     -    0s
     0     0    0.00003    0   21    0.00014    0.00003  78.8%     -    0s
H    0     0                       0.0000744    0.00003  61.3%     -    0s
H    0     0                       0.0000723    0.00003  57.0%     -    0s
     0     0    0.00003    0   21    0.00007    0.00003  57.0%     -    0s
     0     0    0.00003    0   21    0.00007    0.00003  56.1%     -    0s
     0     0    0.00003    0   21    0.00007    0.00003  56.1%     -    0s
     0     2    0.00003    0   21    0.00007    0.00003  56.1%     -    0s
H  514   346                       0.0000676    0.00003  53.1%  49.2    0s
H  521   346                       0.0000675    0.00003  53.0%  48.6    0s
  8311  2257     cutoff   38         0.00007    0.00005  30.7%  53.6    5s
 19255  2323     cutoff   37         0.00007    0.00006  11.1%  57.9   10s

Explored 20782 nodes (1271463 simplex iterations) in 10.79 seconds (21.42 work units)
Thread count was 8 (of 8 available processors)

Solution count 7: 6.75347e-05 6.75857e-05 7.23026e-05 ... 0.0136011

Optimal solution found (tolerance 1.00e-01)
Best objective 6.753470760728e-05, best bound 6.109202603821e-05, gap 9.5398%
Solver Time: 10.789532899856567
```