THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

IEDA 3010: Prescriptive Analytics

Project C: MIP for Portfolio Construction

Rasmus Laansalu, 21043299
Enrico Pescara, 21044853
Edoardo Tarcisio Visconti, 201044685
Alessandro Wiget, 21044841

17/11/2023

Contents

List of Figures

1 Introduction

This report discusses the portfolio construction approach implemented by Grantham, Mayo, Van Otterloo and Company (GMO) using mixed-integer programming (MIP) methods. The purpose is to construct portfolios that closely match a target portfolio in terms of sector and security exposure, while maintaining liquidity, maximizing expected return, and controlling frictional costs. It also aims to reduce the number of distinct stocks and transactions involved.

2 The Papers

2.1 The Markowitz Approach

The Markowitz approach, developed by Harry Markowitz, is a widely used method for constructing investment portfolios. It is based on the principle of diversification, which aims to reduce risk by spreading investments across different assets.The key idea behind is to consider both expected returns and the variance of returns when selecting assets for a portfolio. In particular, investors should seek to maximize expected returns while minimizing the portfolio's overall variance.

To implement the Markowitz approach, the investor begins by identifying available securities and estimating their expected returns and variances. These estimates can be based on historical data, financial analysis, or other relevant information. The goal is to find an optimal combination of assets that offers the highest expected return for a given level. The set of portfolios that offer the highest expected returns for each level of risk is known as "the efficient frontier".

In a complete market, the market portfolio, a portfolio where every stocks has a weight corresponding to its relative market capitalization, belongs to this efficient frontier. If a risk free asset is also available, every portfolio (with specified return) can be built as a linear combination of the very market portfolio and the risk free asset. The set obtained in a such a way is know as "Capital Market Line (CML)".

Overall, the Markowitz approach revolutionized portfolio construction as it provides a systematic framework for investors to make informed decisions and optimize their portfolios based on their individual preferences and risk appetite.

2.2 The MIP Approach

The Mixed-Integer Programming (MIP) approach, offers a novel method for constructing investment portfolios with specific characteristics. Unlike traditional portfolio optimization methods, this approach incorporates integer variables to address complexities associated with portfolio construction.

In the paper, Grantham, Mayo, Van Otterloo and Company LLC (GMO) utilizes mathematical programming techniques to build portfolios that closely resemble target portfolios while considering various factors such as sector and security exposure, liquidity, turnover, and expected return. The objective is to minimize trading costs by reducing the number of distinct stocks and transactions within the portfolio.

Both continuous and discrete decision variables are used to determine the allocation of assets within the portfolio, ensuring that it aligns with the desired characteristics and constraints. In this way, GMO can effectively control frictional costs, maintain liquidity, and achieve similar expected returns to the target portfolios. This approach has also demonstrated significant benefits, including retaining clients, creating growth opportunities, reducing trading costs, improving the trading process, and enhancing performance in simulated fund scenarios.

3 Problem Formulation

3.1 The Variables

Using a MIP approach to solve the single portfolio selection problem, we have two kinds of binary variables: *Names* and *Tickets*. The *Names* variable indicates whether we hold (1) or don't hold (0) stock i in the portfolio. The *Tickets* variable represents our monthly trading activity for each stock, where 1 denotes trading (either buying or selling) while 0 denotes the opposite. In this project, we observe a total of 20 different stocks, labeled as $i = 1, ..., 20$, each with its corresponding *Names* and *Tickets* variables. Minimizing the number of *Names* and *Tickets* is desirable due to custodial fees and transaction costs associated with large portfolios.

For our continuous variables, we introduce the *Weights* variable, which represents the fraction of the portfolio invested in each stock i ($i = 1, ..., 20$). The *Weights* variable ranges from 0.0 to 1.0. Consequently, the target portfolio, after rebalancing, consists of weights $w_t = (w_t(1), w_t(2), ..., w_t(20))$ that sum up to 1.

Additionally, we utilize auxiliary continuous variables Y, F, and X, where $y(i)$, $f(i) \geq 0$ for all i, and $x(s) \geq 0$ for all s. These variables are employed to model different weight functions: $|w_f(i) - w_t(i)|$, $|w_f(i) - w_0(i)|$, and $|\sum_{w_i=1}^{N} M_s(i)(w_f(i) - w_t(i))|$. These weight functions capture our objective of minimizing the number of names and tickets in the portfolio, aiming to limit the number of different stocks and minimize transaction volume.

To represent the piecewise linear function $f_{tc}(i, |w_f(i) - w_0(i)|)$ and the convex illiquidity function $f_{lq}(i, w_f(i))$, we introduce auxiliary continuous variables $x_1(i)$, $x_2(i)$, and $x_3(i)$. These variables, with $x_j(i) \geq 0$ for $j = 1, 2, 3$, assist in modeling the various components of the functions.

3.2 Closeness to Market Portfolio

Our first requirement in the objective function is captured by the term

$$\sum_{i=1}^{N} |w_f(i) - w_t(i)|$$

where $w_t(i)$ is the target portfolio that we aim to achieve after rebalancing, and $w_f(i)$ is the final portfolio that represents our objective portfolio. This term ensures that the sum of the differences between the weights of individual stocks in the final and target portfolios is minimized.

3.3 Exposure to Different Economic Sectors

To mitigate risk, we aim to distribute the stocks across different sectors of the economy. By diversifying our investments across sectors, we reduce the impact of poor performance in a specific sector on the overall portfolio.

By choosing the user-specified penalty λ_s, we can control the importance of each sector in our final portfolio. This is captured by the term

$$\sum_{s=1}^{S} \lambda_s |\sum_{i=1}^{N} M_s(i)(w_f(i) - w_t(i))|$$

in the objective function. A higher penalty for a specific sector encourages a smaller number of stocks for that very sector.

3.4 Limited Number of Different Stocks

To minimize custodial fees and transaction costs, our objective is to limit the number of stocks (Names) in our portfolio. To achieve this, we introduce a constraint variable $y_{\text{names}}(i)$ that represents whether a stock i is included in the final portfolio or not. This constraint captures the total number of different stocks we have in the final portfolio.

$$y_{\text{names}}(i) = \begin{cases} 1 & \text{if } w_f(i) > 0 \\ 0 & \text{if } w_f(i) = 0 \end{cases}$$

By incorporating the user-specified penalty λ_{names}, we can control the number of different stocks in our final portfolio. This is achieved by including the term

$$\lambda_{\text{names}} \sum_{i=1}^{N} y_{\text{names}}(i)$$

in the objective function. The penalty value determines the trade-off between having a smaller number of stocks (higher penalty) or encouraging more diversity in the final portfolio (lower penalty). A higher penalty value will result in fewer different stocks being selected, while a zero or small penalty will encourage the inclusion of a greater variety of stocks in the final portfolio.

3.5 Limited number of Transactions

Similarly, we aim to minimize the number of transactions to reduce costs. To achieve this, we introduce a binary variable $y_{\text{tickets}}(i)$ that captures whether a transaction occurs for each stock i. This variable represents the difference between the final and current weight of stock i.

$$y_{\text{tickets}}(i) = \begin{cases} 1 & \text{if } |w_f(i) - w_0(i)| > 0 \\ 0 & \text{if } |w_f(i) - w_0(i)| = 0 \end{cases}$$

If the weights for stock i remain unchanged during the month, $y_{\text{tickets}}(i)$ is set to zero. However, if we buy or sell some of stock i, $y_{\text{tickets}}(i)$ is set to 1.

The objective function includes the term

$$\lambda_{\text{tickets}} \sum_{i=1}^{N} y_{\text{tickets}}(i)$$

where λ_{tickets} is a user-specified penalty. This term allows us to control the number of transactions in the final portfolio. The penalty value determines whether we encourage more or fewer transactions. A higher penalty value encourages fewer transactions, while a lower penalty value allows for more transactions.

3.6 Return of the Portfolio

One secondary objective is to achieve a high return from our final portfolio. This objective is captured by the term:

$$-\lambda_{\alpha} \sum_{i=1}^{N} \alpha(i) w_f(i)$$

In this term, $\alpha(i)$ represents the expected return of stock i, while $w_f(i)$ represents the weight of stock i in the final portfolio. The term λ_α is a user-specified penalty that determines the importance placed on the overall return of the portfolio.

A higher value of λ_α indicates a greater emphasis on achieving high returns. Conversely, a lower value of λ_α decreases the importance of high returns.

The negative sign in front of the term is due to the nature of the minimization objective. All other terms in the objective function need to be minimized, but the term representing the return of the portfolio is the only one we want to maximize. By negating the term, we align it with the objective of minimizing the overall objective function.

3.7 Illiquidity component of the Portfolio

Our objective is to achieve a portfolio with high liquidity, meaning we aim to minimize the illiquidity of the portfolio. Higher illiquidity makes it more difficult to adjust the weights of our stocks. When a portfolio has a large amount of a particular stock, selling that stock can significantly impact its market price. This is because selling a substantial portion of a stock can drive down its price and thus leading to a loss. Therefore, it is crucial to keep the overall portfolio's liquidity high.

Illiquidity is modeled as a piecewise function, which captures the relationship between the amount of a stock held in the portfolio and its impact on market price.

$$f_{lq}(i,x) = \begin{cases} ls(1,i)x & \text{if } 0 < x < lq(i) \\ ls(2,i)(x - lq(i)) + ls(1,i)lq(i) & \text{if } lq(i) < x < 2lq(i) \\ ls(3,i)(x - 2lq(i)) + ls(2,i)2lq(i) & \text{if } 2lq(i) < x < 4lq(i) \end{cases}$$

By minimizing the illiquidity component of the portfolio, we aim to ensure that the portfolio remains easily tradable and that adjustments to the weights of stocks can be made with minimal impact on market prices. This helps to maintain the overall liquidity and flexibility of the portfolio.

The liquidity consideration in our portfolio optimization model is captured by the term:

$$\lambda_{\text{illiquidity}} \sum_{i=1}^{N} f(i, w_f(i))$$

Here, $\lambda_{\text{illiquidity}}$ is the user-specified penalty that helps us control the illiquidity of the portfolio. The term $\sum_{i=1}^{N} f(i, w_f(i))$ represents the illiquidity function, which is dependent on the stock i and its corresponding weight $w_f(i)$ in the final portfolio.

It's important to note that in a small-sized portfolio, the illiquidity component didn't really play a significant role. This could be due to the fact, that the amount of stock held in the portfolio is not substantial enough to impact market prices. However, in larger firms with substantial investment capital, dealing with illiquidity could be more impactful.

3.8 Minimise transaction Costs

As the trading position grows in comparison to the daily volume of a stock, transaction costs rise due to the influence of portfolio trades on the market. It is evident that as the traded quantity increases, the price impact becomes more substantial, resulting in a corresponding increase in the overall effect.

The transaction cost for trading stock i is captured by a piecewise linear convex function $f_{\text{tc}}(i, |w_f(i) - w_0(i)|)$. The transaction cost function starts to substantially grow based on the volume of trading done in one day.

$$f_{tc}(i,x) = \begin{cases} cs(1,i)x & \text{if } 0 < x < 0.1vol(i) \\ cs(2,i)(x - 0.1vol(i)) + cs(1,i)0.1vol(i) & \text{if } 0.1vol(i) < x < 0.3vol(i) \\ cs(3,i)(x - 0.3vol(i)) + cs(2,i)0.3vol(i) & \text{if } 0.3vol(i) < x < 0.5vol(i) \end{cases}$$

In this model, you can define the relationship between the trading volume (x) and the associated transaction costs for stock i.

The transaction cost consideration is captured by the following term:

$$\lambda_{\text{tc}} \sum_{i=1}^{N} f(i, w_f(i))$$

Here, λ_{tc} is the user-specified penalty that enforces keeping the transaction costs as low as possible. The value of λ_{tc} determines the trade-off between the number of trades and the resulting transaction costs.

By incorporating a specific transaction cost model and adjusting the value of λ_{tc} according to one's own requirements, he can effectively minimize transaction costs while considering the trading position relative to the daily volume of each stock.

3.9 Complete Model Formulation

In the end were able to capture the whole model formulation with the following:

Minimize

$$\sum_{i=1}^{N} y(i) + \sum_{i=1}^{K} \lambda_{\text{sec}}(s)x(s)$$
$$+\lambda_{\text{sec}}(s)x(s)\sum_{i=1}^{N} y_{\text{names}}(i)$$
$$+\lambda_{\text{tickets}}\sum_{i=1}^{N} y_{\text{tickets}}(i) - \lambda_{\alpha}\sum_{i=1}^{N}\alpha(i)w_f(i)$$
$$+\lambda_{\text{illiquidity}}\sum_{i=1}^{N} ls_(1,i)x_1(i) + ls(2,i)x_2(i)$$
$$+ls(3,1)x_3(i)$$
$$\lambda_{\text{tc}}\sum_{i=1}^{N}(cs(1,i)z_1(i) + cs(2,i)z_2(i))$$
$$+cs(3,i)z_3(i)$$

subject to

$$\sum_{i=1}^{N} w_f(i) = 1,$$
$$w_f(i) - w_t(t) \leq y(i), \forall i$$
$$-(w_f(i) - w_t(t)) \leq y(i), \forall i$$
$$w_f(i) - w_0(t) \leq f(i), \forall i$$
$$-(w_f(i) - w_t(t)) \leq f(i), \forall i$$
$$x(s) \geq \sum_{i=1}^{N} M_s(i)(w_f(i) - w_t(i)), \forall s$$
$$x(s) \geq \sum_{i=1}^{N} M_s(i)(w_f(i) - w_t(i)), \forall S$$
$$w_f(i) \leq y_{\text{names}}(i), \forall i$$
$$f(i) \leq y_{\text{tickts}}(i), \forall i$$
$$w_f(i) = x_1(i) + x_2(i) + x_3(i), \forall i$$
$$f(i) = z_1(i) + z_2(i) + z_3(i), \forall i$$
$$0 \leq x_1(i) \leq lq(i), \forall i$$
$$0 \leq x_2(i) \leq lq(i), \forall i$$
$$0 \leq x_3(i) \leq 2 * lq(i), \forall i$$
$$0 \leq z_1(i) \leq 0.1 * vol(i), \forall i$$
$$0 \leq z_2(i) \leq 0.2 * vol(i), \forall i$$
$$0 \leq z_3(i) \leq 0.2 * vol(i), \forall i$$
$$y(i), w_f(i), f(i) \geq 0, \forall i$$
$$x(s) \geq 0, \forall s$$
$$y_{\text{names}}(i), y_{\text{tickets}}(i) \in 0, 1, \forall i$$

The model is built using the consists of all our 5 aforementioned objective functions and variables, with their respective constraints added in.

4 An example: Simulation of the market portfolio

4.1 Baseline strategy

Firstly, according to Markowitz's theory, every efficient portfolio lies on the CML and can therefore be constructed as a combination of a risk-free asset and the market portfolio. This way, every risk profile maintained by the portfolio owner can be satisfied and respected.

As an initial step, based on the theory of Portfolio Selection (Markowitz, 1952), we considered the market portfolio as the target portfolio. In particular, we considered a period of 48 months, during which at the end of each month we calculated the market portfolio using the stock data present in Excel files.

Once we obtained the Markowitz portfolio, at the end of each month, our portfolio tries to replicate the respective month's target portfolio, while taking into account the parameters and objectives desired by the portfolio owner (these decisions are highlighted by varying the values of the lambdas).

4.2 Implementation

In implementing the project, we simulated a set of 20 stocks (provided by the professor in CSV files) and calculated the market portfolio based on the following formula (multiplying the price of a stock by the number of shares):

$$w_{\mathrm{M,i}} = \frac{\text{capital in asset i}}{\text{total capital}} = \frac{N_{\mathrm{i}} P_{\mathrm{i}}}{\sum_{i=1}^{n} N_{\mathrm{i}} P_{\mathrm{i}}}$$

At each month, we calculated the weights target the portfolio, the volume of a stock, and the return of every stock (as the average return in the month).

Some considerations follow:

- In particular, we calculated the parameters for liquidity and volume as follows:
  Lq = ask - bid (doesn't change over time in our model)
  Vol = average volume in the last 10 days / average volume in the whole period (changes every month)
- Choice of the convex function:
  The bigger the weight of a stock, the more illiquid it becomes, which pushes for more diverse portfolios. In our case, we chose x^2 as a convex function. However, since we only considered medium to large-cap stocks, it does not affect the model. It would have been different if we had considered more illiquid assets.

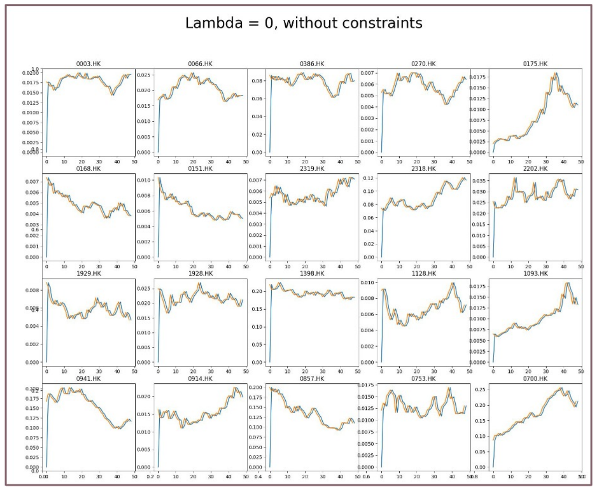
Each user may have a different risk profile, and therefore a difference in the choice of lambda values. In particular, we decided to consider different values of our lambdas to give more importance to certain objectives rather than others.

5 Model tuning

In the next stages, we will illustrate examples and studies that we have conducted.
In particular, we will vary lambda values and show the results of the portfolios.
From these, it will be possible to observe the different behavior of the different stocks within the portfolio based on the objectives that are deemed more important (based on the choice of lambdas).

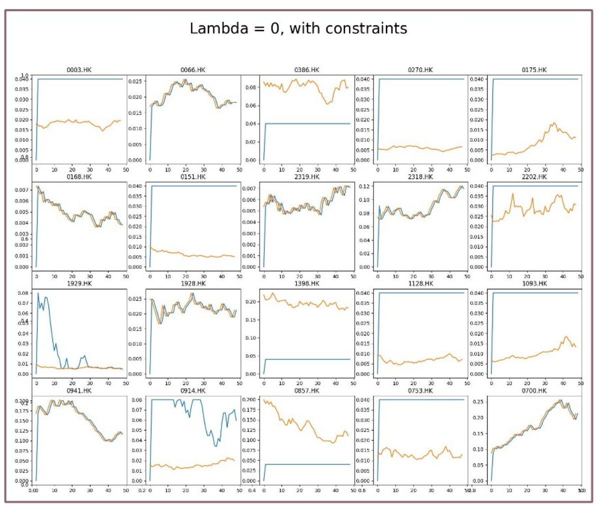
5.1 $\lambda = 0$, without constraints

In particular, with the choice of all lambda values equal to zero, we obtain a w_F portfolio that very accurately replicates (with one month delay) the target portfolio of the different months. In fact, in this way, the only remaining term in the objective function is the one that minimizes the absolute value distances between the weights of w_F and w_t. The result is therefore a w_F portfolio that replicates the target one without any restrictions.



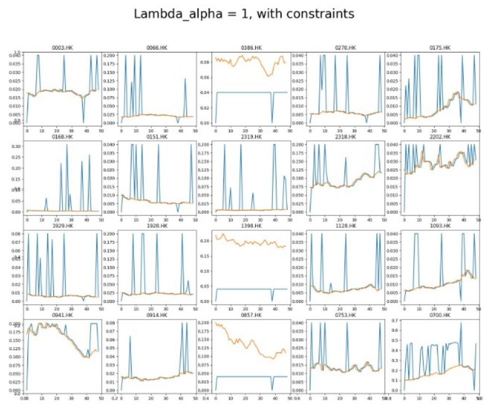
5.2 $\lambda = 0$, with constraints

Taking into account the constraints of volatility and liquidity, we obtain a portfolio where, although some stocks replicate the values of the target portfolio well, others, due to the constraints imposed, do not accurately replicate the market portfolio.



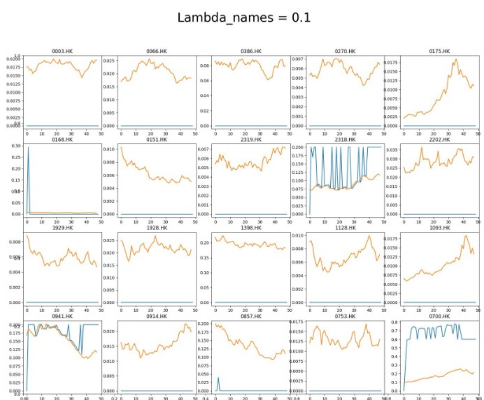
5.3 $\lambda_\alpha = 1$

After testing, we found that a reasonable size for the parameter could be around 1. In particular, this way we consider a relatively high value of return from the stocks.



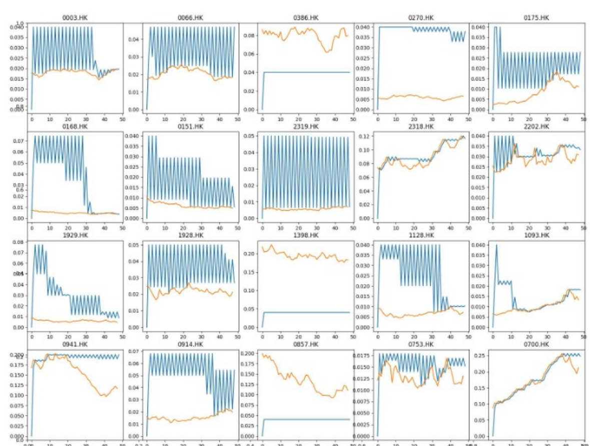
5.4 $\lambda_{\mathbf{names}} = 0.1$

By choosing the parameter equal to 0.1, we ask our portfolio to own a small number of different stocks. This can be seen from the fact that many stocks are not held in the following simulation.



5.5 $\lambda_{\mathbf{illy}} = 0.0001$ and $\lambda_{\mathbf{tc}} = 0.1$

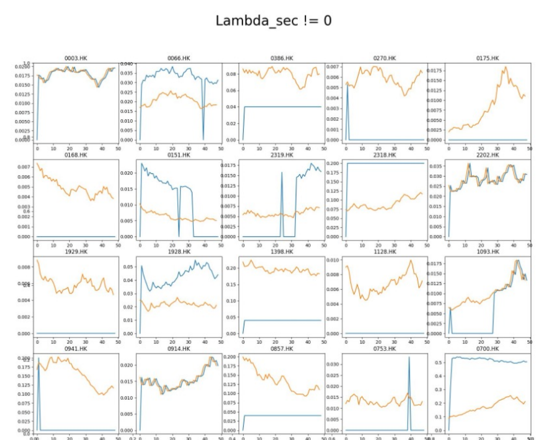
With these choices of lambda values, we aim to obtain a portfolio that, while replicating the target portfolio, focuses on stocks with high liquidity. Additionally, by considering a small value for λ_{tc}, we reduce the transition cost and therefore do not require particular attention in making few transitions. This can be explained by the high number of changes in the weights of the stocks within the portfolio.



5.6 Focus on one particular sector

We consider the desire to have a low number of stocks within our portfolio and in particular to prioritize the 'Communication Services' sector. As a result, we obtain that the stock '0.700.HK' corresponds to 50% of the portfolio. According to the graph, this stock is strictly related to the sector of Communication Services

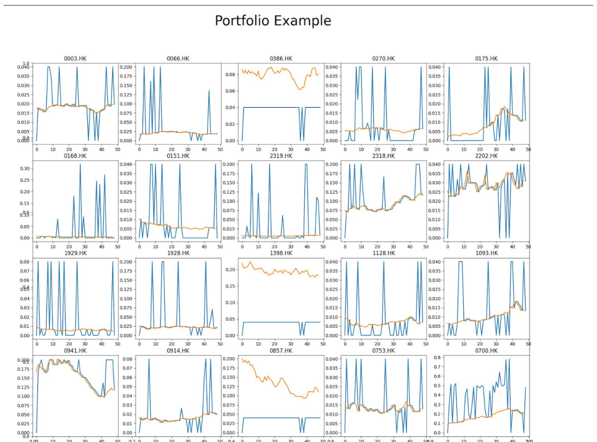
L_SEC	100	UTILITIES
	100	INDUSTRIALS
	100	ENERGY
	100	UTILITIES
	100	CONSUMER CYCLICAL
	100	CONSUMER DEFENSIVE
	100	FINANCIAL SERVICES
	100	REAL ESTATE
	100	HEALTHCARE
	0	COMMUNICATION SERVICES
	100	BASIC MATERIALS
L_names	1	
L_tickets	0	
L_alpha	0	
L_illiquidity	0	
L_tc	0	



5.7 Example of a reasoned choice of lambdas

Idea: Replicate the target portfolio while attempting to reduce the number of stocks, the volume of each stock, and their transitions.

L_SEC	100	UTILITIES
	100	INDUSTRIALS
	0	ENERGY
	100	UTILITIES
	100	CONSUMER CYCLICAL
	100	CONSUMER DEFENSIVE
	0	FINANCIAL SERVICES
	100	REAL ESTATE
	0	HEALTHCARE
	0	COMMUNICATION SERVICES
	100	BASIC MATERIALS
L_names	0,1	
L_tickets	1E-05	
L_alpha	1	
L_illiquidity	1E-04	
L_tc	1E-04	



6 Conclusions and further work

In implementing the construction of a portfolio, classical models may not take into account important objectives or constraints that the portfolio owner may want to respect.

To achieve this goal, we used the Mixed Integer Programming (MIP) approach, through which we considered parameters such as the number of stocks within the portfolio, the number of transitions, the cost of these transitions, and other parameters.

In particular, in accordance with the studied papers, we modeled not only the replication of our portfolio to the target one, but we also required to meet certain objectives and weighted the importance of these objectives through different lambda parameters.

An additional implementation of the project could be to consider other objectives that the investor may want to pursue.
In particular, the reference sectors do not include the green sector, which could be interesting to consider given the modern investors' inclination towards pursuing sustainable investments.
Therefore, incorporating ESG factors could help create portfolios that align with investors' values and preferences, while also potentially providing long-term financial benefits.

7 Appendix

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import gurobipy as gp
from gurobipy import GRB

def MIP(N, S, L, w_0, w_t, alpha, lq, vol, M, ls, cs):

    L_sec = L[0:S]
    L_names = L[S]
    L_tickets = L[S+1]
    L_alpha = L[S+2]
    L_illiquidity = L[S+3]
    L_tc = L[S+4]

    #gp.Env.setParam("OutputFlag",0)

        #Model
    m = gp.Model("MIP")

    #Variables
    Names = m.addMVar((N, 1), vtype=GRB.BINARY, name="Names")
    Tickets = m.addMVar((N, 1), vtype=GRB.BINARY, name="Tickets")
    Weights = m.addMVar((N, 1), lb=0.0, vtype=GRB.CONTINUOUS, name="Weigths")

    Y = m.addMVar((N, 1), lb = 0.0, vtype=GRB.CONTINUOUS, name="Y")
    F = m.addMVar((N, 1), lb = 0.0, vtype=GRB.CONTINUOUS, name="F")
    X = m.addMVar((S, 1), lb = 0.0, vtype=GRB.CONTINUOUS, name="X")

    x_1 = m.addMVar((N, 1), lb = 0.0, vtype=GRB.CONTINUOUS, name="x_1")
    x_2 = m.addMVar((N, 1), lb = 0.0, vtype=GRB.CONTINUOUS, name="x_2")
    x_3 = m.addMVar((N, 1), lb = 0.0, vtype=GRB.CONTINUOUS, name="x_3")

    z_1 = m.addMVar((N, 1), lb = 0.0, vtype=GRB.CONTINUOUS, name="z_1")
    z_2 = m.addMVar((N, 1), lb = 0.0, vtype=GRB.CONTINUOUS, name="z_2")
    z_3 = m.addMVar((N, 1), lb = 0.0, vtype=GRB.CONTINUOUS, name="z_3")

    #Constraints

    #Sum of Weights
    m.addConstr(Weights.sum() == 1, 'Sum of weights')
    #Constraints on Y[i]
    m.addConstrs((Weights[i] - w_t[i] <= Y[i] for i in range(N)), name='Y+')
    m.addConstrs((-(Weights[i] - w_t[i]) <= Y[i] for i in range(N)), name='Y-'
    )
    #Constraints on F[i]
    m.addConstrs((Weights[i] - w_0[i] <= F[i] for i in range(N)), name='F+')
    m.addConstrs((-(Weights[i] - w_0[i]) <= F[i] for i in range(N)), name='F-'
    )
    #Constraints with X[s]
    m.addConstrs((X[s] >= np.sum(M[s][i]*(Weights[i] - w_t[i]) for i in range(
    N)) for s in range(S)), name='X+')
    m.addConstrs((X[s] >= -np.sum(M[s][i]*(Weights[i] - w_t[i]) for i in range
    (N)) for s in range(S)), name='X-')
    #Binary flag for Names
    m.addConstrs((Weights[i] <= Names[i] for i in range(N)), name='W<N')
    #Binary flag for Tickets
    m.addConstrs((F[i] <= Tickets[i] for i in range(N)), name='F<T')
    #Constraints for Liquidity
```

```
    m.addConstrs((Weights[i] == x_1[i] + x_2[i] + x_3[i] for i in range(N)), name='W=x1+x2+x3')
    #Components of Liquidity Constraints
    m.addConstrs((x_1[i] <= lq[i] for i in range(N)), name='x1')
    m.addConstrs((x_2[i] <= lq[i] for i in range(N)), name='x2')
    m.addConstrs((x_3[i] <= 2*lq[i] for i in range(N)), name='x3')
    #Constraints for Volatility
    m.addConstrs((F[i] == z_1[i] + z_2[i] + z_3[i] for i in range(N)), name='F=z1+z2+z3')
    #Components of Volatility Constraints
    m.addConstrs((z_1[i] <= 0.1*vol[i] for i in range(N)), name='z1')
    m.addConstrs((z_2[i] <= 0.2*vol[i] for i in range(N)), name='z2')
    m.addConstrs((z_3[i] <= 0.2*vol[i] for i in range(N)), name='z3')

    #Objective Function
    m.setObjective((Y.sum() +
                   np.sum(L_sec[i] * X[i] for i in range(S)) +
                   L_names*Names.sum() +
                   L_tickets*Tickets.sum() -
                   L_alpha*np.sum(alpha[i]*Weights[i] for i in range(N)) +
                   L_illiquidity*np.sum((ls[i][0]*x_1[i] + ls[i][1]*x_2[i] + ls[i][2]*x_3[i]) for i in range(N)) +
                   L_tc*np.sum((cs[i][0]*z_1[i] + cs[i][1]*z_2[i] + cs[i][2]*z_3[i]) for i in range(N)))
                   , GRB.MINIMIZE)

    m.optimize()

    for v in m.getVars():
        print(f'{v.VarName} = {v.x}')

    if m.Status == GRB.INFEASIBLE:
        m.computeIIS()
        print('\nThe following constraints and variables are in the IIS:')
        for c in m.getConstrs():
            if c.IISConstr: print(f'\t{c.constrname}: {m.getRow(c)} {c.Sense} {c.RHS}')


    return Weights.x
```

Listing 1: Packages Constraints and Variables

```python
data = pd.read_excel(r"/Users/edotarci/Desktop/exchange HK/ieda_op_research/
    Project/DATA.xlsx", sheet_name="GENERAL")

N = pd.DataFrame(data, columns=["N"])
N = int(N.dropna().to_numpy()[0])

S = pd.DataFrame(data, columns=["S"])
S = int(S.dropna().to_numpy()[0])

L = pd.DataFrame(data, columns=["L"])
L = L.dropna().to_numpy()

data1 = pd.read_excel(r"/Users/edotarci/Desktop/exchange HK/ieda_op_research/
    Project/DATA.xlsx", sheet_name="WT in time")
w_t = pd.DataFrame(data1)
w_t = w_t.dropna().to_numpy()

data2 = pd.read_excel(r"/Users/edotarci/Desktop/exchange HK/ieda_op_research/
    Project/DATA.xlsx", sheet_name="VOLUME")
vol = pd.DataFrame(data2)
vol = vol.dropna().to_numpy()

data3 = pd.read_excel(r"/Users/edotarci/Desktop/exchange HK/ieda_op_research/
    Project/DATA.xlsx", sheet_name="alpha")
alpha = pd.DataFrame(data3)
alpha = alpha.dropna().to_numpy()

lq = pd.DataFrame(data, columns=["LQ"])
lq = lq.dropna().to_numpy()


M = pd.DataFrame(data, columns=["0003.HK","0066.HK","0386.HK","0270.HK","0175.
    HK",
                                "0168.HK","0151.HK","2319.HK","2318.HK","2202.
    HK",
                                "1929.HK","1928.HK","1398.HK","1128.HK","1093.
    HK",
                                "0941.HK","0914.HK","0857.HK","0753.HK","0700.
    HK"])
M = M.dropna().to_numpy()


w_0 = np.ones([N,1])/N
w_f = np.zeros([N,49])

names = ["0003.HK","0066.HK","0386.HK","0270.HK","0175.HK",
                                "0168.HK","0151.HK","2319.HK","2318.HK","2202.
    HK",
                                "1929.HK","1928.HK","1398.HK","1128.HK","1093.
    HK",
                                "0941.HK","0914.HK","0857.HK","0753.HK","0700.
    HK"]

for i in range(48):
    w_t_loop = w_t[:,i]
    vol_loop = vol[:,i]

    alpha_loop = np.transpose(alpha[i][:])

    #I suppose that both are following the law f(x) = x^2 for ease of use
    ls = np.concatenate([np.power(lq,2), np.power(2*lq,2), np.power(4*lq,2)],
    axis=1)
```

```
    cs = np.zeros([3,20])
    cs[:][0] = np.transpose(np.power(0.1*vol_loop,2))
    cs[:][1] = np.transpose(np.power(0.3*vol_loop,2))
    cs[:][2] = np.transpose(np.power(0.5*vol_loop,2))

    cs = np.transpose(cs)

    print(cs)

    w_f[:,i+1] = MIP(N, S, L, w_0, w_t_loop, alpha_loop, lq, vol_loop, M, ls,
    cs).reshape(-1)
    w_0 = w_f[:,i]
```

Listing 2: Optimization Step

```

fig = plt.subplots(figsize=(20,15))
for i in range(20):
    fig = plt.subplot(4,5,i+1)
    plt.title(names[i])
    plt.plot(range(49), w_f[i,:])
    plt.plot(range(48), w_t[i,:])

plt.suptitle("Portfolio Example", fontsize=30)
plt.savefig("Portfolio Example.jpg")
```

Listing 3: Plots