VERTEX ASSET ADVISORS

Enhancing Index Fund Efficiency: Integer Programming Solutions for NASDAQ-100 Portfolio Optimization

Prepared by: Janie Chen, Siboney Cardoso, Grayson Merritt, Aditya Sindhavad

Internal Optimization Team

11/03/2024

Executive Summary

This report presents a strategy to construct a reduced-stock index fund that effectively tracks the NASDAQ-100. We used 2023 data to evaluate stock selection methods, including Integer Programming and similarity measures like Pearson correlation and Kullback-Leibler Divergence (KLD). The KLD method performed better for smaller portfolios, while the Mixed Integer Programming (MIP) approach had the lowest tracking errors overall but with higher computational cost. We recommend constructing portfolios with 30 stocks for an optimal balance between tracking accuracy and manageability, providing strong index replication with reduced complexity.

Introduction

This report presents an optimized approach for constructing an index fund that efficiently tracks the NASDAQ-100 index while holding a reduced number of constituent stocks. The primary objective is to minimize the tracking error between the fund and the index, thereby achieving effective replication with fewer assets. This reduction in the number of holdings aims to decrease transaction costs and simplify our portfolio management without significantly compromising performance.

To achieve this goal, we first employ Integer Programming (IP) for stock selection and Linear Programming (LP) to determine optimal portfolio weights. Two similarity measures are explored for selecting representative stocks: the Pearson correlation coefficient and the Kullback-Leibler Divergence (KLD). The correlation coefficient assesses the linear relationship between stock returns, whereas KLD measures the divergence between the probability distributions of stock returns, capturing nonlinear relationships.

Historical price data from 2023 is utilized to train the models and construct the portfolio. The performance of the optimized portfolio is then evaluated using out-of-sample data from 2024 to assess its effectiveness in tracking the NASDAQ-100 index over time. This analysis provides insights into the robustness of the proposed methodology and its potential application in real-world portfolio management. Our code is designed so you can use different training and testing sets to construct the fund.

Data Preparation

For this study, we sourced daily closing prices for the NASDAQ-100 index (NDX) and its constituent stocks for the years 2023 and 2024 and split them into two separate datasets. The 2023 dataset serves as the foundation for model training and portfolio construction, while the 2024 dataset allows us to evaluate the out-of-sample performance of the constructed index fund.

Handling Missing Data

During the data preprocessing phase, we identified missing values in both the index and individual stock price data. Specifically, there were gaps in the NASDAQ-100 index values on certain dates, and missing price data for ARM, which held its initial public offering in September 2023. The presence of missing data can significantly impact the reliability of statistical measures such as correlations and can introduce biases in our optimization process.

To address the missing NASDAQ-100 index values, we manually retrieved the missing data points from Yahoo! finance and inserted them into the dataset. For ARM, the absence of price data prior to its IPO presented a unique challenge. Since ARM was not publicly traded before September 2023, there were no historical prices available for that period. To incorporate ARM into our analysis without distorting the dataset, we applied forward-fill and backward-fill interpolation methods to estimate the missing values. This approach involves propagating the known price backward and forward to fill the gaps. While this method introduces an assumption of price stability for ARM before its IPO, it allows us to include the stock in the portfolio construction process without leaving null values in the dataset.

```
# Handle missing data in stock prices using interpolation and then backward and forward filling if needed
stock_prices_2023_interpolated = stock_prices_2023.interpolate(method='linear', limit_direction='both')
if stock_prices_2023_interpolated.isnull().values.any():
    stock_prices_2023_interpolated = stock_prices_2023_interpolated.fillna(method='ffill').fillna(method='bfill')
```

Code for filling in missing price data

Calculating Returns and Aligning Data

After addressing the missing data, we proceeded to calculate the daily returns for each stock and the index. Returns were computed as the percentage change in the adjusted closing price from one trading day to the next. This standardizes the data and enables the comparison and correlation of the different securities. To ensure consistency in our analysis, we aligned the dates of the stock returns with the index returns. This alignment involved selecting the common dates present in both datasets, effectively synchronizing the time series for accurate computation of similarity measures and tracking errors.

Optimizing

Stock Selection

The first step in our methodology is to select a subset of stocks that effectively represent the NASDAQ-100 index. Selecting the right stocks is crucial because it ensures that the reduced portfolio still captures the overall behavior of the index. We utilized similarity measures

to quantify the relationships between stocks, allowing us to identify those that are most representative.

Similarity Measures

Initially, we calculated the Pearson correlation coefficient between the returns of each pair of stocks. The Pearson correlation coefficient is a well-known statistical measure that evaluates the linear relationship between two variables. In the context of stock returns, it quantifies how closely the returns of two stocks move together. The coefficient is calculated using the formula:

$$\rho_{ij} = \frac{\mathrm{Cov}(R_i, R_j)}{\sigma_{R_i}\sigma_{R_j}}$$

where R_i and R_j are the return series of stocks i and j, $\mathrm{Cov}(R_i,R_j)$ is their covariance, and σ_{Ri} and σ_{Rj} are their standard deviations. A value close to 1 indicates a strong positive correlation, meaning the stocks tend to move in the same direction.

```
# compute Pearson correlation matrix
rho_df = returns_2023.corr()
# similarity matrix using correlation
rho_corr = rho_df.values
```

Code for creating the correlation similarity matrix

While the Pearson correlation provides valuable insights into linear relationships, we wanted to explore a new way to form a similarity matrix that could capture more complex similarities between stocks. This led us to discover the Kullback-Leibler Divergence (KLD) as an alternative measure.

Exploring Kullback-Leibler Divergence (KLD)

In our research for a more nuanced similarity measure, we turned to information theory and came across the Kullback-Leibler Divergence. KLD measures how one probability distribution diverges from another, essentially quantifying the difference between two distributions. We considered that by treating the return distributions of stocks as probability distributions, we could use KLD to measure the similarity between their behaviors. To apply KLD, we first discretized the return distributions of each stock into bins, creating histograms that represent the probability distributions. The code we supplied uses 100 bins. This allowed us to compare the entire distribution of returns rather than just average returns or linear relationships.

The KLD from distribution P to distribution Q is given by:

$$D_{\mathrm{KL}}(P\|Q) = \sum_{k} P(k) \log \left(\frac{P(k)}{Q(k)} \right)$$

However, KLD is not symmetric and can be infinite if Q(k)=0 for any k where P(k)>0. To address this, we used the symmetric version of KLD:

$$D_{\mathrm{sym}}(P,Q) = \frac{1}{2} \left(D_{\mathrm{KL}}(P\|Q) + D_{\mathrm{KL}}(Q\|P) \right)$$

By computing the symmetric KLD between the return distributions of stocks, we obtained a measure of how similar the distributions are in both directions. To transform this divergence into a similarity measure, we applied the following formula:

$$\rho_{ij} = \frac{1}{1 + D_{\mathrm{sym}}(P_i, P_j)}$$

```
# compute probability distributions for each stock's returns
prob_distributions = []
for stock in stocks:
    hist, _ = np.histogram(returns_2023[stock], bins=bins, density=False)
    # avoids zero probabilities by doing this
    hist = hist + 1e-10
    prob_dist = hist / hist.sum()
    prob_distributions.append(prob_dist)

# compute symmetric KL Divergence matrix
kl_divergence_matrix = np.zeros((n, n))
for i in range(n):
    P_i = prob_distributions[i]
    for j in range(n):
        P_j = prob_distributions[j]
        D_ij = entropy(P_i, P_j)
        D_ji = entropy(P_j, P_i)
        D_sym = (D_ij + D_ji) / 2
        kl_divergence_matrix[i, j] = D_sym

# convert KL divergence to similarity measure
similarity_matrix_kld = 1 / (1 + kl_divergence_matrix)
# similarity matrix using KL Divergence
rho_kld = similarity_matrix_kld
```

Code to create KLD similarity matrix

This transformation ensures that the similarity measure ρ_{ij} ranges between 0 and 1, with higher values indicating greater similarity between the return distributions of stocks i and j.

Explaining KLD a Different Way

Imagine comparing the behavior of two stocks by looking at the entire range of their returns, not just how their averages move together. Discretizing the returns into histograms, we capture the full distribution of possible returns for each stock. The symmetric KLD then measures how one histogram differs from another. If the histograms are similar—meaning the stocks have similar return patterns—the KLD will be small, and the similarity score ρ_{ij} will be close to 1. By incorporating KLD into our analysis, we aimed to capture linear relationships and differences in the overall distribution of returns, potentially providing a more comprehensive similarity measure than correlation alone.

Optimization Model for Stock Selection

After computing the similarity matrices using the Pearson correlation and KLD-based measures, we formulated an optimization model to select the optimal subset of stocks. The objective was to maximize the total similarity between all stocks in the index and the selected stocks, ensuring that our reduced portfolio is as representative as possible.

We used Integer Programming to solve this problem. The decision variables in the model indicate whether a stock is selected and which stocks it represents. The model maximizes the sum of similarities between each stock and its assigned representative from the selected subset, subject to constraints that ensure exactly m stocks are selected, every stock is represented, and a stock can only represent others if it is selected.

By solving this optimization problem, we identified the set of m stocks that collectively have the highest similarity to the entire index based on the chosen similarity measure (either correlation or KLD). This approach allows us to systematically and optimally reduce the number of stocks while maintaining the portfolio's representativeness.

Portfolio Weight Optimization

Once we selected the subset of stocks, the next step was to determine their optimal weights in the portfolio to minimize tracking error concerning the NASDAQ-100 index. Tracking error measures the divergence between the portfolio's and index's returns, and minimizing it ensures that our portfolio closely follows the index's performance.

We formulated a Linear Programming (LP) model to minimize the sum of absolute differences between the portfolio and index returns over the in-sample period (2023). The decision variables in this model are the weights assigned to each selected stock. The objective function minimizes the total tracking error. At the same time, the constraints ensure that the weights sum to 1 (indicating a fully invested portfolio) and that there is no short-selling (weights are non-negative).

By solving the LP problem, we obtained the optimal weights for the selected stocks that minimize the tracking error. This allows us to construct a portfolio that closely replicates the index's behavior using only a subset of its constituent stocks.

Mixed Integer Programming (MIP) Approach

In addition to the two-step process of stock selection and weight optimization, we explored a Mixed Integer Programming (MIP) model that integrates both steps into a single framework. The MIP model simultaneously decides which stocks to include in the portfolio and determines their optimal weights to minimize tracking error. The MIP model includes both binary variables (indicating whether a stock is selected) and continuous variables (the weights of the stocks). The objective is to minimize the total tracking error between the portfolio and the index over the in-sample period. Constraints in the model ensure that exactly m stocks are selected, weights sum to 1, weights are non-negative, and weights are linked to selection (i.e., a stock's weight is zero if it is not selected).

The advantage of the MIP approach is that it considers the interaction between stock selection and weighting directly. By integrating both decisions, the MIP model can evaluate the impact of including or excluding a stock while simultaneously adjusting its weight to optimize the overall tracking error. This holistic view can potentially yield better results compared to the two-step approach.

```
# add our varaibles one by one so I can understand what I am doing
w = mip_model.addVars(n, vtype=GRB.CONTINUOUS, lb=0.0)
y = mip_model.addVars(n, vtype=GRB.BINARY)
z = mip_model.addVars(len(index_returns_2023), vtype=GRB.CONTINUOUS, lb=0.0)

mip_model.setObjective(quicksum(z[t] for t in range(len(index_returns_2023))), GRB.MINIMIZE)

# add our constraints
# Linking weights to selection variables
# Since weights sum to 1 and are non-negative, we get a BIG M of 1
M = 1.0
mip_model.addConstrs((w[i] <= M * y[i] for i in range(n)))
mip_model.addConstr(quicksum(y[i] for i in range(n)) == m)
mip_model.addConstr(quicksum(w[i] for i in range(n)) == 1)

# get our absolute deviations
for t in range(len(index_returns_2023)):
    portfolio_return = quicksum(w[i] * returns_2023.iloc[t, i] for i in range(n))
    index_return = index_returns_2023.iloc[t]
    mip_model.addConstr(z[t] >= index_return - portfolio_return)
    mip_model.addConstr(z[t] >= portfolio_return - index_return)

mip_model.optimize()
```

Code for optimizing fund using MIP

However, the MIP model is computationally intensive and takes significantly longer to solve, especially for larger values of m. The complexity arises from the combination of integer and continuous variables and the exponential growth of possible combinations as the number of stocks increases. Solving MIP problems is known to be NP-hard, meaning that the time required

to find the optimal solution can increase dramatically with the size of the problem. To manage the computational challenges, we set time limits on the solver. This means that the solutions obtained may not always be optimal but are the best found within the allotted time. We expected that the MIP model might yield better results because it can potentially find a more effective balance between stock selection and weighting, leading to lower tracking errors.

Results and Analysis

Performance Across Different m Values

Correlation Method

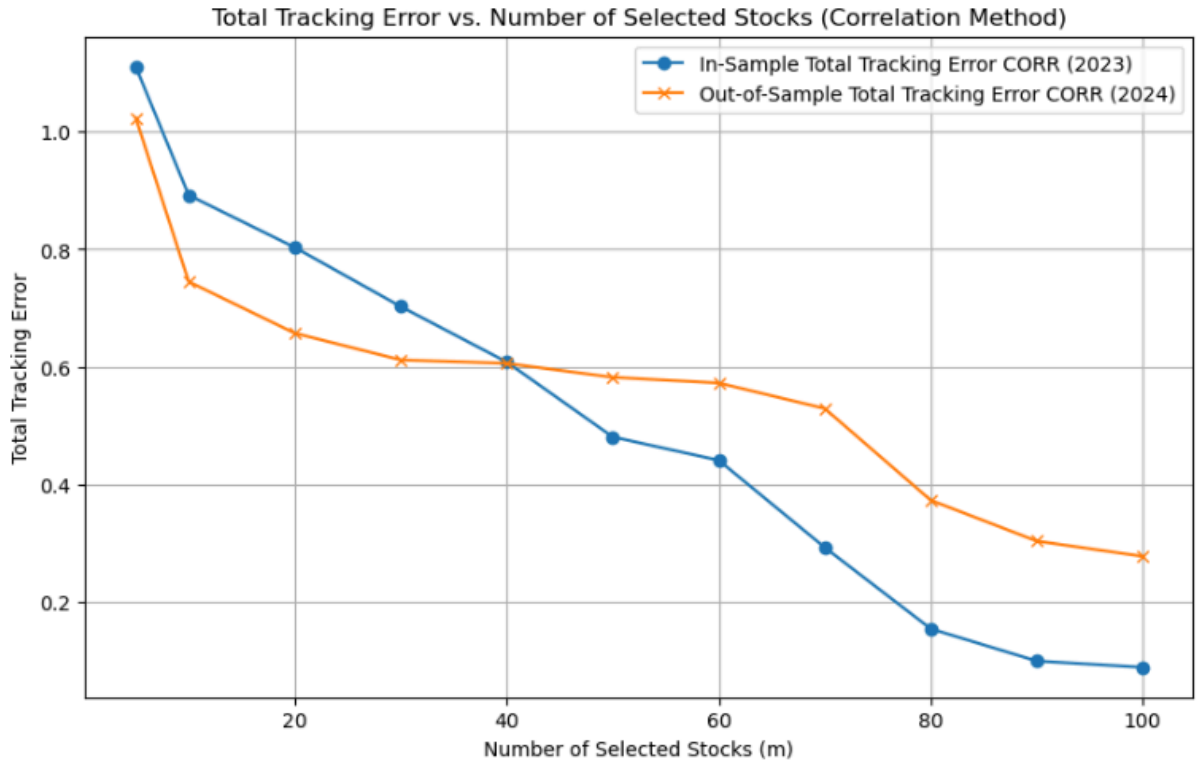
Using the Pearson correlation coefficient as the similarity measure, we selected subsets of stocks for each m and optimized their weights to minimize the tracking error with respect to the NASDAQ-100 index.

In-Sample Performance (2023):

As m increased, the total tracking error decreased. For instance, with m = 5, the in-sample tracking error was 1.110995, and it reduced to 0.088113 when m = 100. This trend indicates that adding more stocks to the portfolio improves its ability to replicate the index's performance. However, the rate of improvement diminishes beyond a certain point. The most significant reductions in tracking error occurred when increasing m from 5 to 30, after which the gains became progressively smaller.

Out-of-Sample Performance (2024):

The out-of-sample tracking error followed a similar decreasing pattern. Starting from 1.023837 with m = 5, it declined to 0.277031 when m = 100. Notably, the out-of-sample tracking errors were generally lower than the in-sample errors up until m = 40. This a somewhat surprising result, and this helps indicate just how hard this stock selection model is!



In and out of sample tracking error for a correlation matrix over number of stocks

Analysis:

The Correlation method demonstrates that increasing the number of stocks in the portfolio enhances its tracking ability. However, diminishing returns are evident in certain sections, such as beyond m = 80 for both in and out of sample.

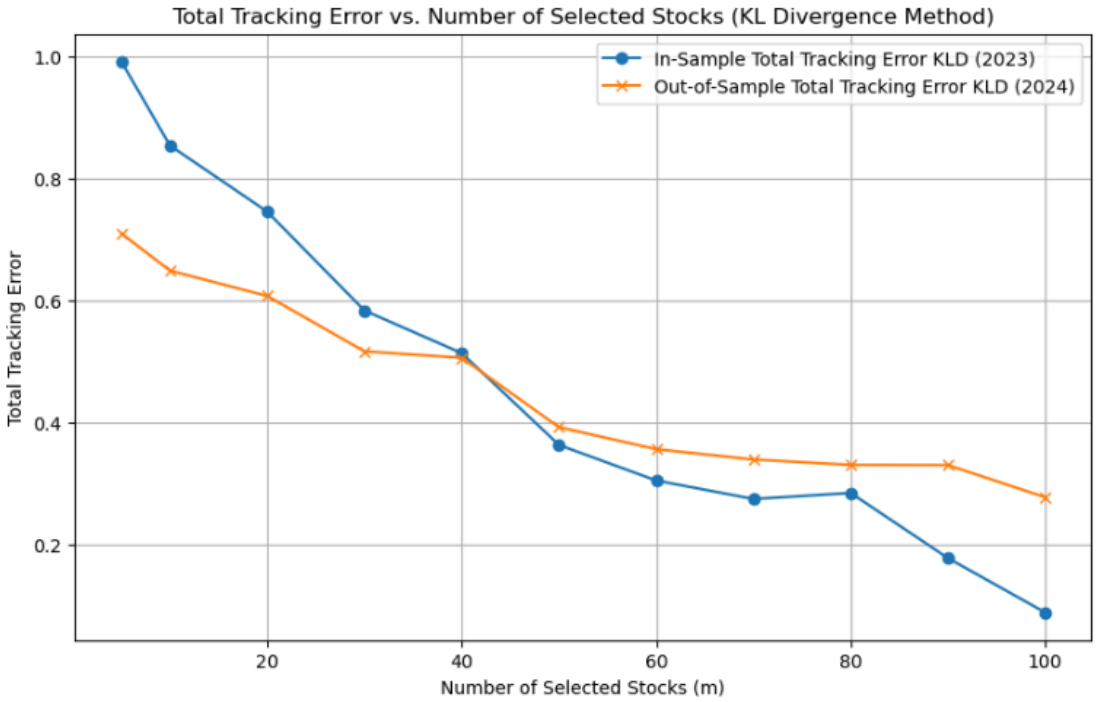
Kullback-Leibler Divergence Method

In-Sample Performance (2023):

The KLD-based portfolios consistently achieved lower tracking errors compared to the Correlation method for the same m . With m = 5, the in-sample tracking error was 0.991490, decreasing to 0.088113 when m = 100, matching the Correlation method at that point. The KLD method showed more substantial improvements for smaller m, indicating its effectiveness in selecting highly representative stocks with fewer holdings.

Out-of-Sample Performance (2024):

Similarly, the out-of-sample tracking errors were lower than those of the Correlation method for the same m values. Starting from 0.709610 with m = 5, the tracking error decreased to 0.277031 at m = 100. The KLD method maintained better performance consistency between the in-sample and out-of-sample periods compared to the Correlation method.



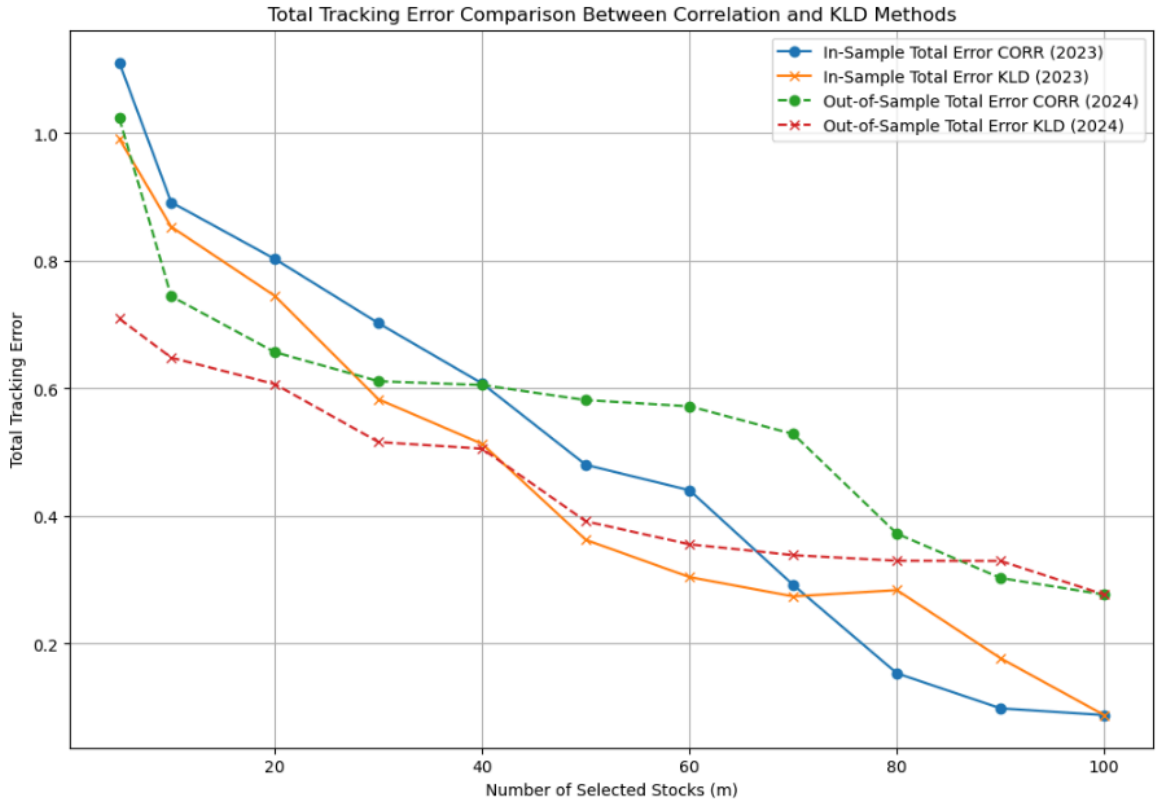
In and out of sample tracking error for a KLD matrix over number of stocks

Analysis:

The KLD method outperforms the Correlation method across most m values, particularly for smaller portfolios. The diminishing returns in tracking error reduction become noticeable beyond m = 50, suggesting that a portfolio size of around 40 stocks may offer an optimal balance between tracking accuracy and managing a more complex portfolio with more fees.

Comparison of Both Methods

To directly compare the performance of the Correlation and KLD methods, we plotted the tracking errors of both methods on the same graph. Note that the shape on the line distinguishes which method was used.



In and out of sample tracking error for a KLD matrix and

Correlation matrix over number of stocks

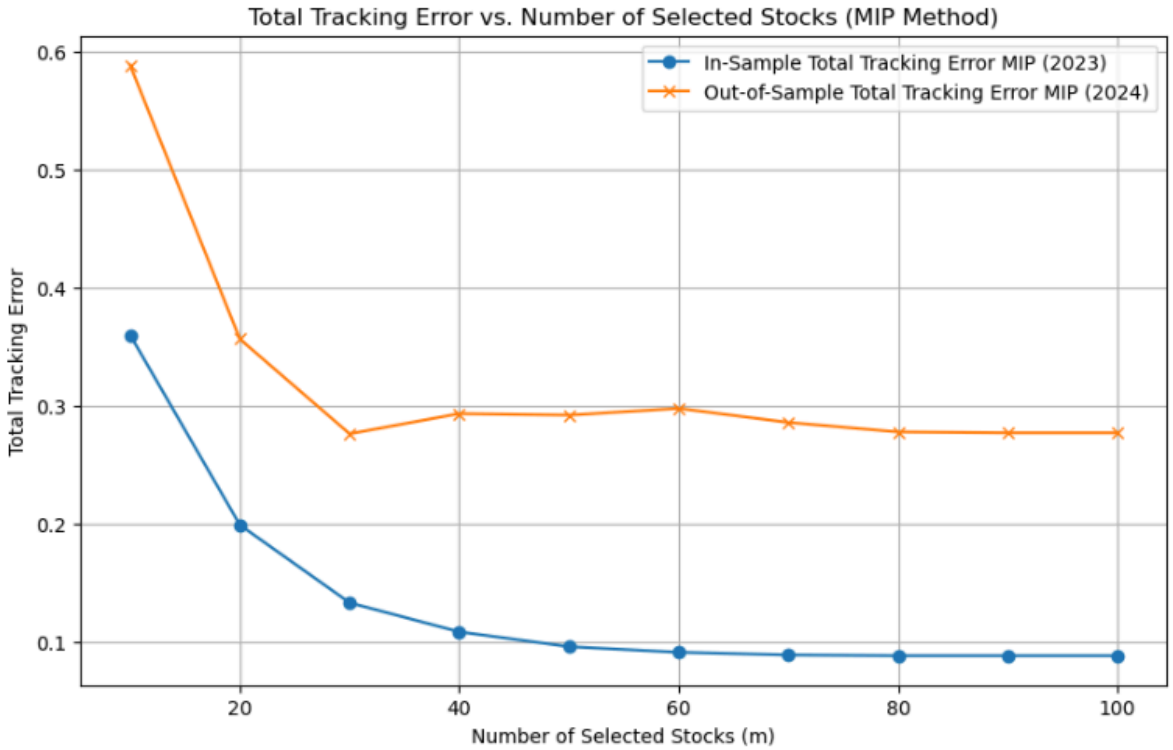
Analysis:

The comparison reveals that the KLD method consistently achieves lower tracking errors than the Correlation method for almost all m values, especially in smaller portfolios. The performance gap narrows as m increases, converging at m=100 where both methods include all stocks in the index. At m=90, KLD's tracking error rises above that of the Correlation method, indicating overfitting or less robust stock selection. However, because the lowest out-of-sample error generally flattens out past m=50, this is negligible. The KLD method's superior performance suggests that it captures stock similarities more effectively, leading to better stock selection for index replication.

MIP Method Performance

In-Sample and Out-of-Sample Performance:

For each m, the MIP method achieved the lowest tracking errors among the three methods. With m=10, the in-sample tracking error was 0.359938, significantly lower than the corresponding errors from the Correlation and KLD methods. As m increased, the tracking error continued to decrease, reaching 0.088113 at m=100. The out-of-sample tracking errors followed a similar pattern, demonstrating the MIP method's robustness. The best out of sample error is m=30 where the error dips slightly. After m=30, there are diminishing returns on adding more stocks to the fund.



In and out of sample tracking error for the MIP solution over number of stocks

Analysis:

The MIP method's superior performance underscores the advantage of simultaneously optimizing stock selection and weights. However, computational challenges were significant. The MIP model is computationally intensive as it looks at every combination of stock and forces some to not have a weight of zero. Despite these challenges, the MIP method demonstrates that integrating selection and weighting decisions can lead to better tracking performance.

Regarding the Big M parameter used in the MIP model, we set M = 1, which is appropriate because the weights of individual stocks cannot exceed 1 when the total weight sums to 1. The smallest value of M we can use is equal to the maximum possible weight any stock could have. Since we do not allow short-selling and weights are non-negative, M = 1 is the smallest possible value. Using a smaller M could inadvertently eliminate feasible solutions by overly constraining the weights.

Another important observation is that the cumulative error appears lowest at m = 30 rather than m = 100, even though m = 100 represents the full stock corpus intended for replication. This finding is counterintuitive but can be explained by minor variations resulting from the rounding off of stock prices in the index and minor discrepancies due to data scraping. Notably, this difference in error is on the order of 0.000x, indicating that it has a negligible impact on performance accuracy.

Results Discussion: IP vs. MIP Selection

In the IP method, stocks are selected by maximizing a similarity metric, and the weights are optimized separately afterward. For example, when m = 10, the chosen stocks were ['GOOGL', 'AMZN', 'AAPL', 'CHTR', 'EA', 'MCHP', 'MSFT', 'NVDA', 'ORLY', 'TSLA']. This selection process, combined with a separate weight optimization step, led to in-sample and out-of-sample tracking errors of 0.001446 (2023) and 0.003034 (2024), respectively.

Unlike IP, MIP handles both stock selection and weight optimization in a single process, which often results in better tracking performance. For m=20, the selected stocks were ['GOOGL', 'AMZN', 'AAPL', 'ADSK', 'AVGO', 'CSCO', 'EA', 'LRCX', 'MCHP', 'META', 'MSFT', 'NVDA', 'ORLY', 'PEP', 'QCOM', 'ROST', 'SBUX', 'TSLA', 'VRSK', 'VRTX']. This approach achieved lower tracking errors of 0.000801 for 2023 and 0.001837 for 2024.

The IP approach is suitable for simpler cases with a two-step process of stock selection followed by weight optimization. In contrast, MIP combines these steps, effectively reducing tracking errors, particularly as the number of stocks increases. While IP is adequate for smaller portfolios, MIP typically delivers better results for larger selections where tracking accuracy is crucial.

Zero Weights in Linear Optimization

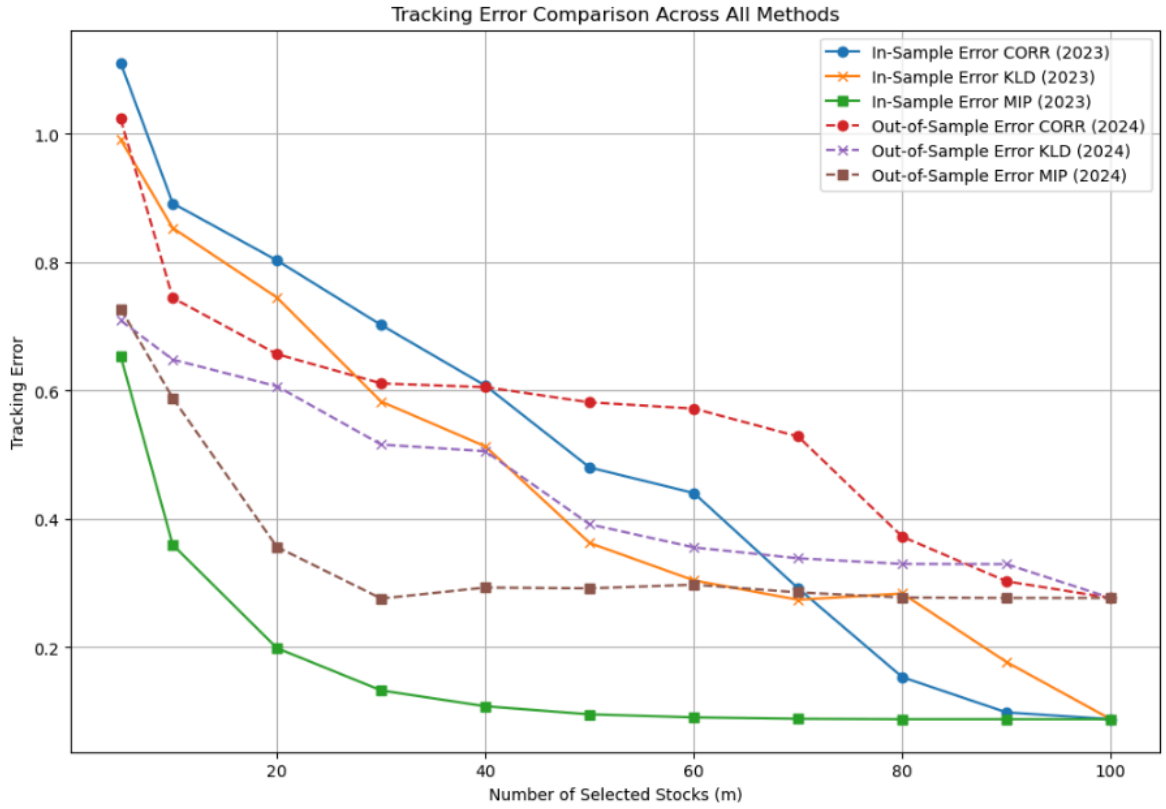
In our approach, we initially select a subset of m stocks and apply linear optimization to minimize error according to the objective function. During this process, weights are allocated to each stock based on their contribution to the optimization target. Notably, some stocks receive a weight of zero, indicating that they are not relevant to achieving the objective as per the linear optimization model. This contributes to the overall error in the solution, as stocks with zero weight do not participate in replicating the target portfolio.

By contrast, in the Mixed Integer Programming (MIP) approach, stock combinations and weights are optimized through the Gurobi solver. Unlike linear optimization, MIP allocates at

least a minimal weight to each selected stock, which contributes to a more accurate solution. However, this enhanced performance comes with a trade-off in computation time; MIP is significantly slower due to the exhaustive search process, though it delivers notably improved accuracy.

Comparison of All Methods

To provide a comprehensive comparison, we plotted the tracking errors from all three methods on the same graph.



In and out of sample tracking error for all three methods (correlation, KLD, MIP)

Analysis:

The combined visualization highlights that the MIP method consistently outperforms the other two methods across all m values! The KLD method ranks second, outperforming the Correlation method, particularly in smaller portfolios. The differences between methods diminish as m approaches 100, where all methods converge to include all stocks in the index.

The analysis indicates that the MIP method is the most efficient for tracking the index, achieving lower tracking errors with fewer stocks. The KLD method offers a balance between

performance and computational efficiency, outperforming the Correlation method while being less computationally demanding than the MIP approach.

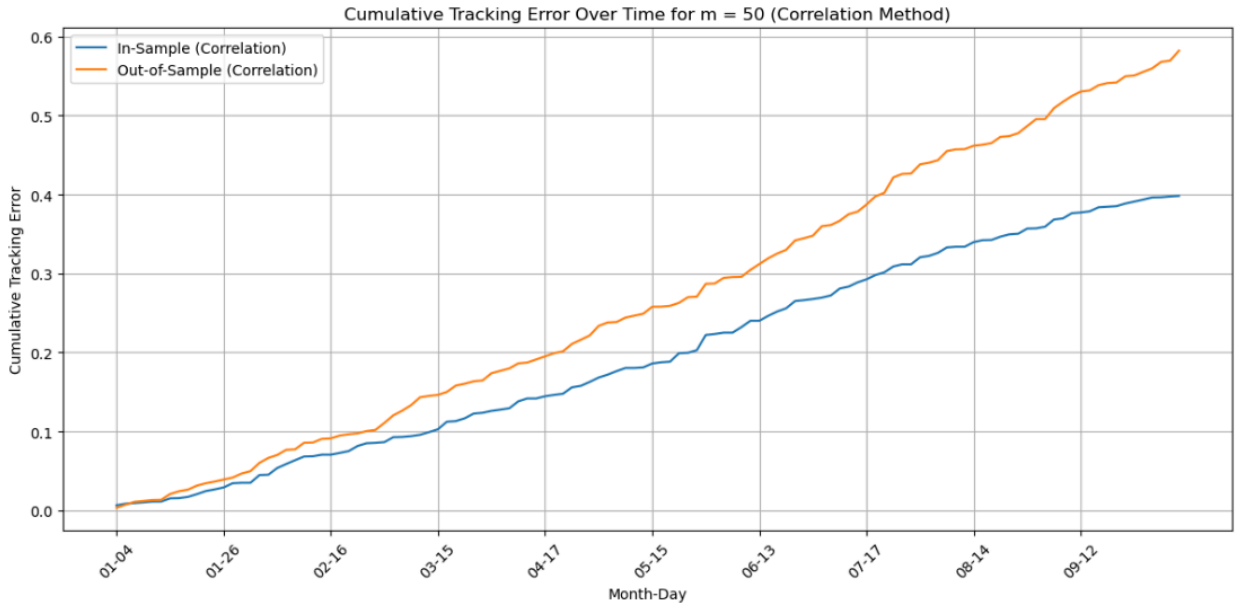
Across all methods, we observed that increasing m leads to lower tracking errors, but the rate of improvement diminishes as m becomes large. For the KLD method, significant reductions in tracking error are achieved up to m = 50 beyond which the benefits taper off. Similarly, the MIP method shows diminishing returns after m = 30. This suggests that a moderately sized portfolio can achieve near-optimal tracking performance without the need to include all 100 stocks.

Cumulative Tracking Error Over Time

To understand how the tracking error accumulates over time, we analyzed the cumulative tracking error for a selected m value in each method.

Cumulative Error: Correlation Method

We selected m = 50 for the Correlation method and plotted the cumulative tracking error over the in-sample and out-of-sample periods.



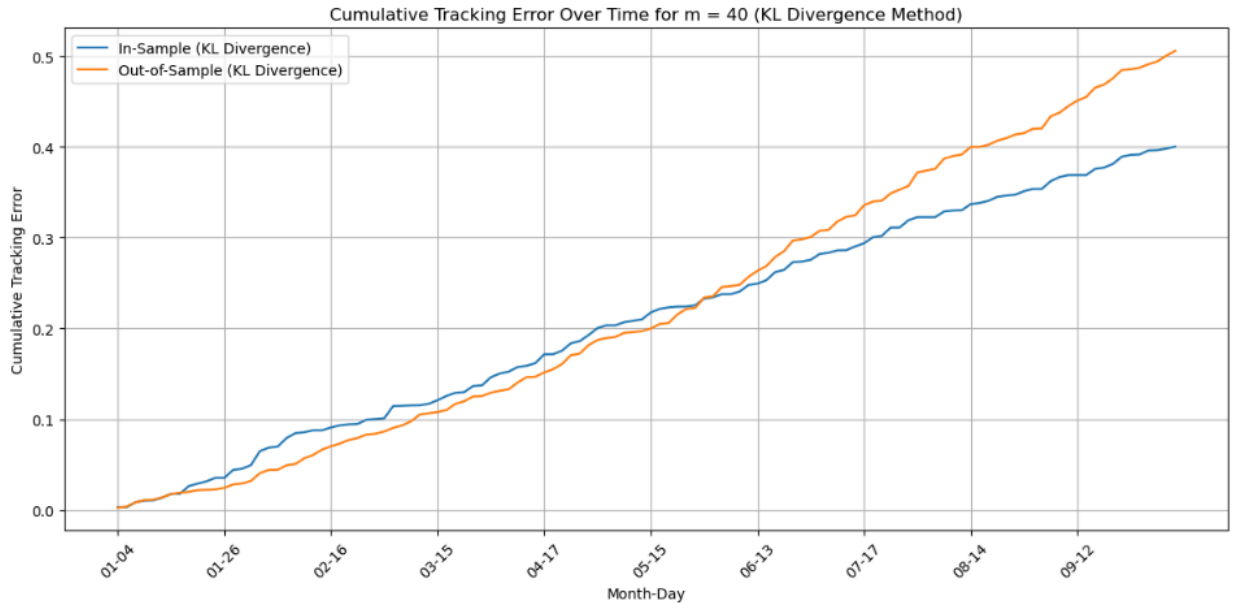
Cumulative tracking error over time for correlation

Analysis:

The cumulative tracking error increases steadily over time, reflecting the daily discrepancies between the portfolio and the index returns. There are periods where the tracking error accumulates more rapidly, possibly due to market volatility or significant events affecting specific stocks in the portfolio. The out-of-sample cumulative error grows slightly faster than the in-sample error, indicating reduced tracking performance in 2024.

Cumulative Error: KLD Method

For the KLD method, we also selected m = 40 and plotted the cumulative tracking error over time.



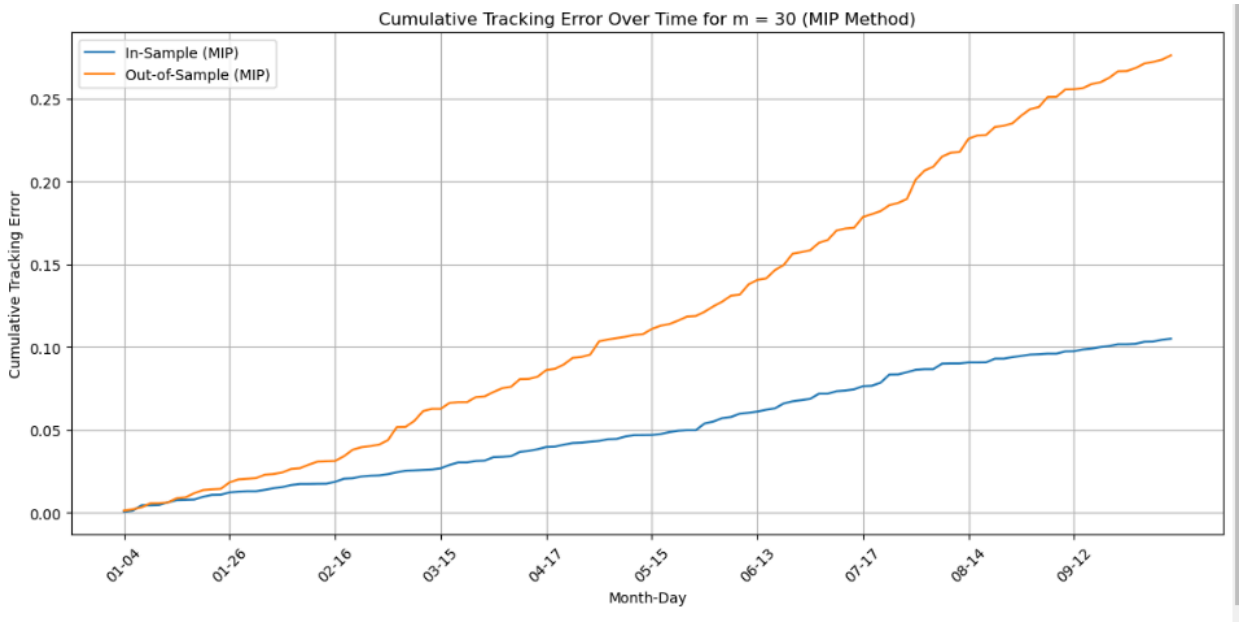
Cumulative tracking error over time for KLD

Analysis:

The KLD method shows a slower accumulation of tracking error compared to the Correlation method. The cumulative error grows more gradually, suggesting better day-to-day tracking of the index. The gap between the in-sample and out-of-sample cumulative errors is smaller, indicating more consistent performance across different periods.

Cumulative Error: MIP Method

For the MIP method, we selected m = 30 due to it having the lowest error.



Cumulative tracking error over time for MIP

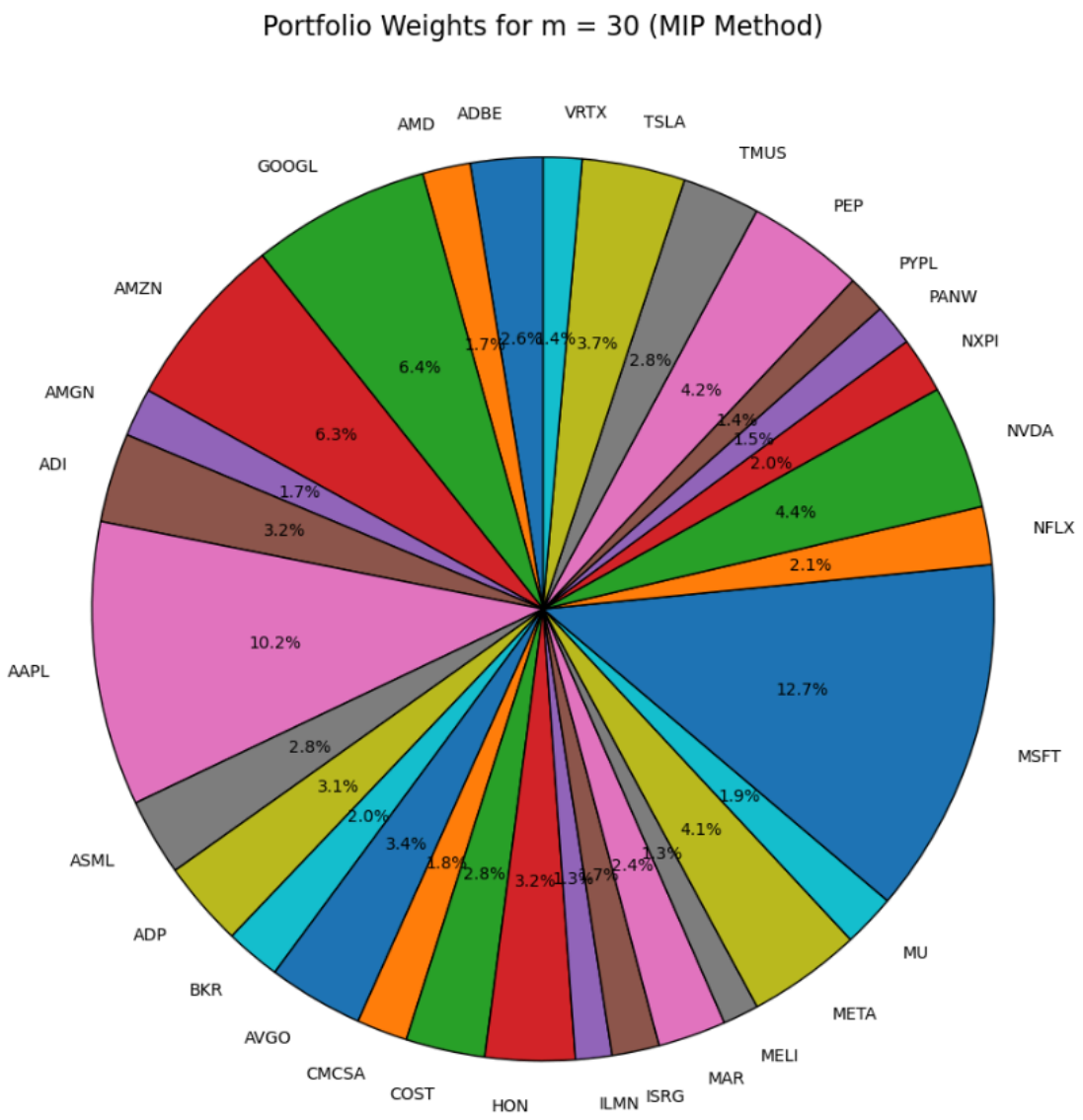
Analysis:

The MIP method exhibits the slowest accumulation of tracking error among the three methods. The cumulative error remains low throughout both the in-sample and out-of-sample periods, highlighting the method's effectiveness in closely replicating the index's performance over time. The consistency between the in-sample and out-of-sample cumulative errors suggests that the MIP-optimized portfolio adapts well to changing market conditions.

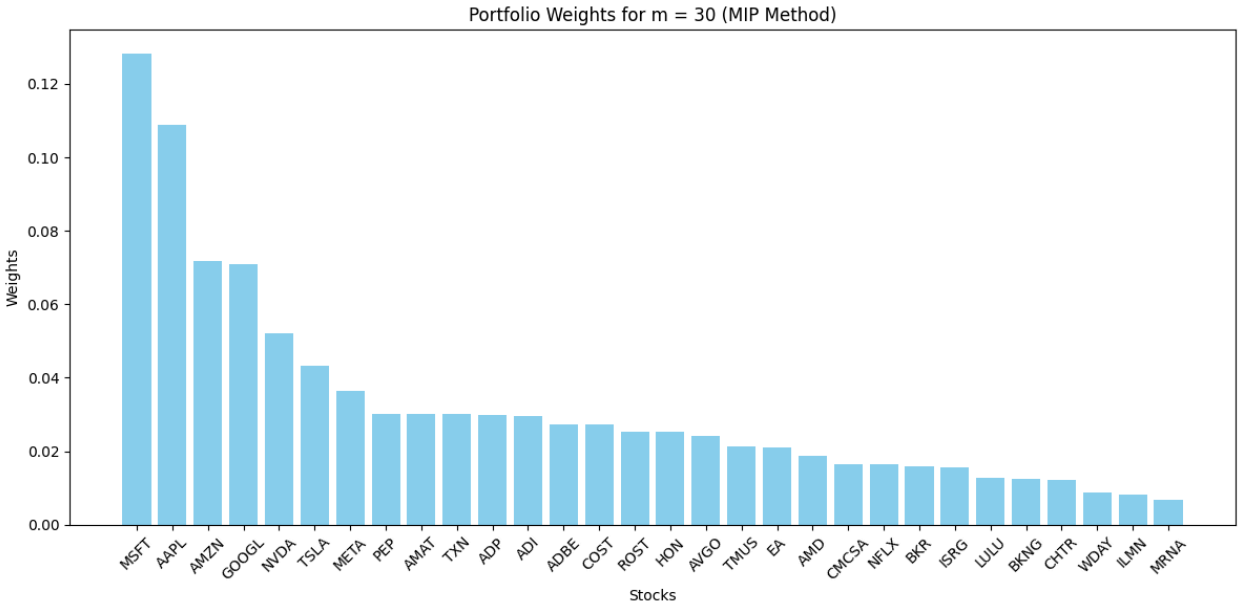
Recommendations

Based on our analysis, we recommend adopting the MIP method for portfolio construction, as it consistently outperforms the Correlation and KLD methods, achieving the lowest tracking errors across all m values. By integrating stock selection and weight optimization, the MIP method effectively balances representativeness and precision. Despite the higher computational demands, especially for larger m, the improved performance justifies its use.

For practical implementation, portfolios with m up to 50 offer an excellent trade-off between tracking accuracy and computational feasibility. If computational resources are limited, the KLD method serves as an effective alternative. It outperforms the Correlation method and requires less computational effort than the MIP approach, particularly advantageous for smaller portfolios where it achieves lower tracking errors with fewer stocks. Considering the observed diminishing returns beyond certain m values, it is unnecessary to include all 100 stocks to achieve high tracking accuracy. **A portfolio size of 30 stocks using MIP** is sufficient to closely replicate the index's performance while reducing management complexity and transaction costs.



Pie chart showing our recommend stocks and weights



Bar chart showing our recommend stocks and weights