SCALABLE FINANCIAL INDEX TRACKING WITH GRAPH NEURAL NETWORKS

Zepeng Zhang and Ziping Zhao

School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China.
Email: {zhangzp1, zhaoziping}@shanghaitech.edu.cn

ABSTRACT

As a prevailing passive investment strategy in the financial world, index tracking aims at replicating or surpassing the performance of a financial index. The core part of an index tracking strategy is to design a sparse index tracking portfolio (ITP) from a basket of candidate financial assets. In this paper, a scalable two-stage approach is developed for ITP design under the minimax criterion, which consists of an asset selection stage (i.e., to select a subset of the assets from the index constituent stocks) and a capital allocation stage (i.e., to allocate the capital among the selected assets). The asset selection problem is tackled via a well-calibrated graph neural network (GNN), followed by a light-weight linear programming problem for capital allocation resolved via a standard solver. The idea proposed in this paper is novel for the area of ITP design in that it is especially scalable for tracking large-scale and dynamic-updating financial indices. Numerical simulations validate the scalability and high-efficiency of the proposed GNN-based approach with comparisons to the standard solver-based approach.

Index Terms—Index tracking, portfolio design, asset selection, graph neural network, mixed-integer programming.

1. INTRODUCTION

In the world of financial fund management (a.k.a. portfolio management), there are two prevailing investment ideas, namely active investment and passive investment [1]. Active investment strategies assume that the market is not perfectly efficient, and hence the active managers will try to beat the market by finding those “diamond in the rough” assets they believe in. Passive investment strategies, on the other hand, assume the market is efficient and cannot be beaten in the long run, and therefore the passive managers are trying to follow the markets and more specifically a specific financial index [2]. Vast analyses of historical data have shown that most active funds failed to outperform the market [3]. However, the market was rising in the long run, indicating that passive strategies can make a decent profit by following the market.

As one of the most popular investment strategies, index tracking aims at reproducing or exceeding the performance of a financial index [4]. The core part of an index tracking strategy is to design an index tracking portfolio (ITP) from a basket of candidate financial assets. A straightforward way to achieve such a goal is to invest in all the underlying assets constituting an index, which leads to the full replication index tracking procedure. However, the full replication strategy will cause small positions and high liquidity, resulting in high transaction costs [5]. To overcome the aforementioned defects, an alternative method is to maximally reproduce the index performance by holding only a subset of the assets in an index (i.e., designing a sparse portfolio), leading to the well-known sparse ITP design problem [6, 7]. In practice, the sparse ITP design problem can be formulated as a mixed-integer programming (MIP) problem [8], which is commonly fed into the off-the-shelf solvers, such as CPLEX [9], SCIP [10], and Gurobi [11] for problem resolution. Such solvers can achieve the optimal results for a class of MIP problems based on optimality-guaranteed methods like the branch-and-bound algorithm [12]. However, they are only suitable for small-scale MIP problems, and it is prohibitive in practice when handling large-scale MIP problems since the worst-case complexity of such optimality-guaranteed methods is exponential [12]. Therefore, researchers have been seeking for more efficient algorithms practical for large-scale problems [13]. Especially, such large-scale problem settings are normal in the context of ITP design [14]. For instance, the global index S&P Global 1200 consists of around 1200 equities and the U.S. index Wilshire 5000 contains around 5000 equities.

Deep learning techniques have made impressive development in diverse fields [15, 16]. Not coincidentally, many researchers have been trying to improve the MIP solvers leveraging such techniques. In the context of index tracking, given that the dimension and the constituent stocks of an index are always dynamically changing over time, a scalable algorithm that can be adaptive to problems with changing scales is desirable. Besides that, the demand for a scalable algorithm is also beneficial in that the financial training data (such as the stock and index returns) is scarce compared to other application scenarios. Therefore, we propose a scalable ITP design approach based on the graph neural network (GNN) [17–19] model which can be trained on small-scale problems with fewer data.

This work was supported in part by the National Nature Science Foundation of China (NSFC) under Grant 62001295 and in part by the Shanghai Sailing Program under Grant 20YF1430800.
and can generalize to large-scale problems for practical setting. In fact, several attempts have been carried out to apply the GNN-based methods on MIP problems [20]. And as shown in [21], the GNN-based approach can outperform many other learning-based methods [22–24] in terms of solution quality.

In fact, although there has been a surge of interest in using deep learning methods to accelerate the MIP solvers, it is not well explored in areas like signal processing and financial engineering. Given this, this paper suffices to be an attempt to leverage the power of GNNs for efficient resolving of a specific financial engineering problem and also sheds light on how to deal with a large number of problems that arise in the areas of signal processing and financial engineering with similar problem structures. In this paper, the ITP design problem is formulated to minimize the maximum absolute deviation of the portfolio returns from the target index returns, resulting in a minimax [25] MIP problem. A two-stage approach is developed for problem solving, which consists of an asset selection stage (i.e., to select a subset of assets of the index) and a capital allocation stage (i.e., to allocate the capital among the selected assets). The asset selection problem is considered as a classification problem, with a well-calibrated GNN serving as the classifier. In the capital allocation stage, a standard solver is used to cope with a light-weight linear programming problem. Numerical simulations are conducted to demonstrate the effectiveness and the scalability of the proposed approach.

2. MINIMAX MODEL FOR ITP DESIGN

Given an index composed of \( N \) assets, we denote the returns of the index over time 1 to \( T \) as \( r_{\text{ind}} \triangleq [r_{\text{ind},1}, \ldots, r_{\text{ind},T}]^T \) and the returns of its constituents at time \( t \) (\( t = 1, \ldots, T \)) as \( r_t \triangleq [r_{1,t}, \ldots, r_{N,t}]^T \). To construct an ITP defined by portfolio weights \( w \triangleq [w_1, \ldots, w_N]^T \), a commonly used measure of the tracking performance is the maximum absolute deviation (a.k.a. minimax criterion) between the ITP returns and the index returns. This minimax criterion leads to a worst-case robust portfolio design problem, which stands for a “worst-case protection strategy” in index tracking [25]. The resulting ITP design problem is given as follows:

\[
\begin{align*}
\text{minimize} & \quad \max_{t=1,\ldots,T} |r_{\text{ind},t} - w^T r_t| \\
\text{subject to} & \quad 1^T w = 1 \\
& \quad 1^T z \leq k \\
& \quad z \odot l \leq w \leq z \odot h \\
& \quad z \in \{0,1\}^N,
\end{align*}
\]

where \( k \) limits the number of assets to hold in the sparse ITP. By further introducing variable \( s \), Problem (2) can be equivalently converted into an epigraph form as follows:

\[
\begin{align*}
\text{minimize} & \quad s \\
\text{subject to} & \quad 1^T w = 1 \\
& \quad 1^T z \leq k \\
& \quad z \odot l \leq w \leq z \odot h \\
& \quad s \geq r_{\text{ind},t} - w^T r_t, \ t = 1, \ldots, T \\
& \quad s \geq -r_{\text{ind},t} + w^T r_t, \ t = 1, \ldots, T \\
& \quad z \in \{0,1\}^N.
\end{align*}
\]

Evidently, the resulting ITP design problem (3) becomes a classical MIP problem.

3. TWO-STAGE APPROACH FOR ITP DESIGN

To realize an efficient problem resolution procedure for large-scale problem settings, a two-stage approach will be developed, which consists of an asset selection stage and a capital allocation stage.

3.1. Asset Selection

The asset selection problem will be addressed via a well-calibrated GNN. Firstly, Problem (3) is modeled as a graph \( G = \{V, E\} \) as depicted in Fig. 1, where \( V_{\text{cv}}, V_{\theta}, V_{\lambda}, \) and \( V_{\varepsilon} \) are the vertex sets that contains all the continuous variables, binary variables, constraints, and objective, respectively, and \( E \) is the edge set. As shown in Fig. 1, there is no edge between \( V_{\text{cv}} \) and \( V_{\theta} \) or within a vertex set. Besides capturing the connections among the vertices (i.e., the variables, constraints, and objective), the detailed coefficients of Problem (3) are extracted as edge features and their statistics are served as vertex features. Note that the coefficient of a constraint in the objective can be obtained from the dual problem of (3).  

After the collection of features, the feature vectors of vertices and edges are embedded to the same dimension, with the vertex embedding denoted by \( \mathbf{v}_i, i \in V \) and the edge embedding represented by \( \mathbf{e}_{ij}, (i, j) \in E \).

To include the edge embedding information into the message passing process and recognize the different importance of the embeddings of other vertices to a vertex \( i \in V \), the first-order attention mechanism is adopted [26] in our model. That is, only the embeddings of vertices in \( N_i \) (the set of first-order neighbor vertices of vertex \( i \)) are aggregated for the update of \( \mathbf{v}_i \) with the attention coefficient of vertex \( j \in N_i \) to
where \( v_\text{dated} \) by \( W \) is based on \( v \) and \( \text{ELU} \).

\[ \sum_{i \in V_i} \exp(\alpha_{ij}) \]

It should be noted that \( \tilde{\alpha}_{ij} = \tilde{\alpha}_{ji} \) does not always hold since the normalization procedures of them are conducted with different vertex sets. Once the normalized attention coefficients are obtained, they can be used to identify the importance that one vertex has on another.

In each stage of the message passing process, the embeddings of neighbor vertices from different vertex sets are calculated by

\[ \sigma(a \cdot \text{CAT}(v_i, e_{ij}, v_j)) \]

where \( a \) is a trainable weight vector, \( \sigma(\cdot) \) and \( \text{CAT}(\cdot) \) represent the Leaky ReLU function and the concatenate function that joins two vectors, respectively. To make the attention coefficients comparable, they are further normalized across the neighbor two vectors, respectively. To make the attention coefficients \( \{V_{\tilde{\alpha}} \} \) to \( i \in V_{\tilde{\alpha}} \) is performed in two stages as follows:

\[ v_i \leftarrow \sigma(W_1 \cdot \text{CAT}(v_i, \sum_{j \in V_i \cap N_i} \alpha_{ij} v_j)) \]

and

\[ v_i \leftarrow \sigma(W_2 \cdot \text{CAT}(v_i, \sum_{j \in V_i \cap N_i} \alpha_{ij} v_j)) \]

where \( W_1 \) and \( W_2 \) are trainable weight matrices and \( \sigma(\cdot) \) denotes the ELU function. The whole message passing process is conducted as follows. Firstly, \( v_i, i \in V_o \) is updated by aggregating \( v_i, i \in \{V_{\tilde{\beta}}, V_{\tilde{\gamma}}\} \), then \( v_i, i \in \{V_{\tilde{\beta}}, V_{\tilde{\gamma}}, V_o\} \) are used to update \( v_i, i \in V_c \), followed by the update of \( v_i, i \in V_o \) based on \( v_i, i \in V_c \), and in the end \( v_i, i \in \{V_{\tilde{\beta}}, V_{\tilde{\gamma}}, V_c\} \) is updated by \( v_i, i \in \{V_c, V_o\} \). The message passing process will be executed recurrently for several times, followed by two fully-connected layers to realize the prediction of binary variables which takes the form of

\[ z_i \leftarrow \text{sigmoid}(W_{\text{out}} \cdot v_i), \forall i \in V_{\tilde{\alpha}} \]

where \( W_{\text{out}} \) is a trainable weight matrix and \( \text{sigmoid}(\cdot) \) is the sigmoid activation function.

The GNN will be optimized by minimizing the binary cross-entropy loss between the output of the GNN and the solution given by CPLEX under a supervised learning scheme. Since the output values of the GNN are not binary, i.e., \( 0 < z_i < 1, i = 1, \ldots, N, \) we will first sort them in descending order and then set \( z_i = 1 \) for the first \( k \) elements (i.e., the corresponding assets are active/selected in the ITP) and \( z_i = 0 \) for the remaining ones (i.e., the corresponding assets are inactive/deselected in the ITP and the capital invested in them are zero).

### 3.2. Capital Allocation

When the asset selection process is fulfilled via GNN, the ITP design problem suffices to assign the portion of dollars to the \( k \) active assets, which is a linear programming problem as follows:

\[
\begin{align*}
\text{minimize} & \quad s \\
\text{subject to} & \quad 1^T w = 1 \\
& \quad 1 \leq w \leq h \\
& \quad s \geq r_{\text{ind},t} - w^T r_1, \quad t = 1, \ldots, T \\
& \quad s \geq -r_{\text{ind},t} + w^T r_1, \quad t = 1, \ldots, T,
\end{align*}
\]

where \( w \) contains only \( k \) unknown elements. In the original ITP design problem (3), there are \( N \) binary variables and \( N + 1 \) continuous variables, while the capital allocation problem (4) is much easier with only \( k + 1 \) continuous variables, in which case the computational burden is reduced significantly, and Problem (4) can be solved via the off-the-shelf solvers efficiently.

### 4. NUMERICAL SIMULATIONS

In this section, the performance of the proposed GNN-based two-stage approach (“GNN”) is used to represent the proposed two-stage approach for short in this section) will be evaluated via numerical simulations on a synthetic dataset, while the standard solver CPLEX will be used for data generation and performance comparisons.

#### 4.1. Synthetic Dataset Generation

Assuming an index consists of \( N \) assets, firstly we generate the return \( r_{it} \) of asset \( i (i = 1, \ldots, N) \) at time \( t (t = 1, \ldots, T) \) from a normal distribution with mean of \( \mu = 0.01 \) and variance of \( \sigma^2 = 0.1 \). Then the price of asset \( i \) at time \( t \) is calculated by \( p_{it} = \prod_{i=1}^{t-1}(r_{it} + 1)p_{i0} \) with \( p_{i0} \) being the initial price of asset \( i \). With the price-weighted setting of the index (i.e., the portion of the assets in the index are weighted based on the ratio of their price to the sum of all prices of the underlying assets) [2], the price of the index at time \( t \) is given by

\[ p_{\text{ind},t} = \sum_{i=1}^{N} \frac{p_{it}}{\sum_{i=1}^{N} p_{it}} p_{i0} \]

and the index return at time \( t \) is computed as \( r_{\text{ind},t} = \frac{p_{\text{ind},t} - p_{\text{ind},t-1}}{p_{\text{ind},t-1}} \).

#### 4.2. Numerical Evaluations of Tracking Performance

To evaluate the tracking performance of GNN, several synthetic cases with different problem scales are generated: Case...
Table 1. Tracking performance of GNN and CPLEX (\(AD_{\text{max}}\)
and \(AD_{\text{mean}}\) are averaged over all test data).

<table>
<thead>
<tr>
<th>Case</th>
<th>Approach</th>
<th>(AD_{\text{max}})</th>
<th>(AD_{\text{mean}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N = 30, k = 10)</td>
<td>CPLEX</td>
<td>0.02821</td>
<td>0.01599</td>
</tr>
<tr>
<td></td>
<td>GNN</td>
<td>0.04224</td>
<td>0.02234</td>
</tr>
<tr>
<td>(N = 50, k = 20)</td>
<td>CPLEX</td>
<td>0.02600</td>
<td>0.01036</td>
</tr>
<tr>
<td></td>
<td>GNN</td>
<td>0.02688</td>
<td>0.01653</td>
</tr>
<tr>
<td>(N = 70, k = 20)</td>
<td>CPLEX</td>
<td>0.02690</td>
<td>0.00925</td>
</tr>
<tr>
<td></td>
<td>GNN</td>
<td>0.03092</td>
<td>0.01874</td>
</tr>
</tbody>
</table>

Table 2. Tracking performance of Case 2 and Case 3 with GNN trained on Case 1.

<table>
<thead>
<tr>
<th>Case</th>
<th>(AD_{\text{max}})</th>
<th>(AD_{\text{mean}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N = 50, k = 20)</td>
<td>0.02851</td>
<td>0.01701</td>
</tr>
<tr>
<td>(N = 70, k = 20)</td>
<td>0.03112</td>
<td>0.01846</td>
</tr>
</tbody>
</table>

From Table 1, we can see that the tacking performance of GNN is comparable to CPLEX. The tracking performance in terms of cumulative returns of one instance from Case 1 (\(N = 30, k = 10\)) is further depicted in Fig. 2.

The underlying GNN framework endows the proposed two-stage approach with scalability, which means it can be trained on small-scale instances and generalize to large-scale instances. We apply the GNN trained on Case 1 (\(N = 30, k = 10\)) to Case 2 (\(N = 50, k = 20\)) and Case 3 (\(N = 70, k = 20\)), with the tracking performances presented in Table 2.

From Table 1 and Table 2, it can be concluded that the GNN trained on small-scale instances can attain acceptable tracking performances when generalized to large-scale instances.

The comparisons in terms of runtime between GNN and CPLEX are recorded in Table 3, where two larger datasets, i.e., Case 4 (\(N = 100, k = 20\)) and Case 5 (\(N = 150, k = 30\)), are included. Considering that the training data for Case 4 (\(N = 100, k = 20\)) and Case 5 (\(N = 150, k = 30\)) are expensive to obtain, the GNN model trained on Case 1 (\(N = 30, k = 10\)) is used for testing. The upper bound of the runtime is set to be 1800s, and the runtime and the success rate are listed in Table 3. We can see that CPLEX fails to output a converging solution for Problem (3) under the time limit for some instances, while the GNN approach can handle all the problems with orders of magnitude faster than CPLEX. Compared to the standard solvers which can be prohibitive for large-scale ITP design problems due to their high computational complexity, the superiority of the proposed two-stage approach is more pronounced.

5. CONCLUSIONS

In this paper, the minimax model for index tracking portfolio design has been considered. A two-stage approach has been developed for problem solving, which contains a calibrated graph neural network to realize asset selection in the first stage, and a standard numerical solver for the resolution of the capital allocation problem in the second stage. Numerical simulations demonstrate the proposed graph neural network-based approach is scalable and can achieve comparable tracking performance with orders of magnitude improvement on runtime compared with the standard solver-based approach.
6. REFERENCES


