A Deep Learning-Aided Approach to Portfolio Design for Financial Index Tracking

Zepeng Zhang and Ziping Zhao
School of Information Science and Technology
ShanghaiTech University, Shanghai 201210, China
Email: zepengzhang8@gmail.com, zhaoziping@shanghaitech.edu.cn

Abstract—This paper considers the index tracking portfolio (ITP) design problem in financial markets, which aims at reproducing the performance of a financial index by investing in a subset of the assets constituting it. From a regression-based point of view, the ITP design problem is formulated as a mixed-integer programming (MIP). Leveraging the graph convolutional network (GCN), a calibrated GCN is proposed for asset selection followed by a lightweight MIP problem to realize asset allocation. Numerical simulations show that compared to existing methods the proposed learning-aided approach can generate comparable ITP design results and significantly accelerate the computation which is favorable for practical index tracking targets in finance.

I. INTRODUCTION

A financial index is constructed from a collection of assets, such as stocks and bonds, that is used to indicate the trend of a market or a part of it. In the world of financial fund management (a.k.a., portfolio management), there are two prevailing investing ideas, namely active investment and passive investment [1]. The active fund managers aim at outperforming the indexes through sophisticated portfolio design techniques, while historical data shows that most active funds are hard to succeed in the long run [2]. As an alternative, passive fund managers target to replicate the performance of a financial index. A straightforward way to achieve such a goal is to invest in all the underlying assets substituting the index which stands for full replication of the index. However, this is not practical since it will cause small positions and high liquidity, leading to high transaction costs. A commonly used approach in passive fund management is to maximally reproduce the index performance by holding only subset assets of it, which leads to the crucial index tracking portfolio (ITP) [3], [4] design problem in finance.

The classical way for ITP design is to formulate it as a mixed-integer programming (MIP) problem [5], [6]. Then the off-the-shelf MIP solvers such as CPLEX [7], SCIP [8], and Gurobi [9] can be used for problem-solving. These solvers are developed based on optimality-guaranteed methods like the branch-and-bound algorithm [10] and optimal results can be achieved theoretically for certain MIP problems. However, the worst-case complexity of such methods can be exponential, which makes the solver-based approach not amenable for high-dimensional problems [1]. Especially, the high-dimensional problem setting is normal in ITP design problems, say, for the global index S&P Global 1200, the dimension of the asset universe is around 1200 and for the U.S. regional index Wilshire 5000, the dimension of the asset universe is around 5000.

Deep learning technique has recently emerged as a promising decision supporting approach in many research areas. And not coincidentally many researchers have been trying to introduce learning-based methods for accelerating the solving process of MIP problems [11]–[14]. In particular, graph convolutional network (GCN) models [15]–[18] emerged recent years have been extensively used for efficient resolution of MIP problems. For example, in [19], GCN is used to predict the binary variables in MIP problems, and promising results are achieved on different types of MIP problems. Authors of [19] firstly extract the variables, the constraints, and the objective from a MIP as vertices to construct a tripartite graph, which is followed by embedding and message processing steps over the graph and finally predictions of the binary variables suffice to be the output of the network. Numerical results show that the GCN has the ability to extract structure information from the MIP formulation. Using a similar graph structure, paper [20] focuses on the variable selection task in branch-and-bound. The variables and constraints of a MIP problem are abstracted as vertices in a bipartite graph. By applying only one round of information transmission, the GCN model can outperform many other machine learning-based methods [11], [12], [21] and SCIP’s default algorithm in most cases according to the experiment results.

In this paper, inspired by the variable prediction target realized by GCN in MIP solving, GCN methods will be explored to solve the ITP design problem. The ITP design problem is first formulated as a MIP, and then the GCN method is applied to predict the binary variables in the MIP (asset selection stage). After that, the original MIP problem reduces to be a lightweight one, and to solve the reduced MIP problem (asset allocation stage), standard solvers can be used. Numerical simulations show that compared to existing methods the proposed learning-aided approach can generate comparable ITP design results and significantly accelerate the computation which is favorable for practical index tracking targets in finance. To the best of our knowledge, this is the first time that a deep learning approach has been developed for index tracking portfolio design problems.

This work was supported by the Shanghai Sailing Program under Grant 20YF1430800.
II. INDEX TRACKING PORTFOLIO (ITP) DESIGN

A. The Design Objective in Financial Index Tracking

In finance, an index tracking portfolio is usually designed to reproduce the return performance (profit and loss in percentage) of a financial index [1]. Suppose an index consists of $N$ assets with the price of asset $i$ ($i = 1, \ldots, N$) at time $t$ ($t = 1, \ldots, T$) denoted by $p_{it} \in \mathbb{R}_{++}$ and the return of asset $i$ at time $t$ denoted by $r_{it} \triangleq \frac{p_{i(t+1)} - p_{i(t)}}{p_{i(t)-1}} \in \mathbb{R}$, then the portfolio return of the designed ITP at time $t$ can be expressed as

$$r_{p,t} \triangleq \sum_{i=1}^{N} w_i r_{it},$$

where $w_i \in \mathbb{R}$ is the portfolio weight denoting the dollar proportion invested in asset $i$ and hence satisfying $\sum_{i=1}^{N} w_i = 1$. If the units of asset $i$ held in the ITP is $x_i$, we have $w_i = x_i p_{iT} / \sum_{i=1}^{N} x_i p_{iT}$. Suppose the total value invested in this ITP is $B$, and we can assume $B$ is time invariant without significant loss of generality, then we have $B = \sum_{i=1}^{N} x_i p_{iT}$ and hence $w_i = x_i p_{iT} / B$.

From a regression-based point of view, an ideally designed ITP should have intercept of zero and slope of one when we regress the returns of the ITP against the index returns. We denote the index returns by $r_{ind,t}$ for $t = 1, \ldots, T$. If the returns of asset $i$ is regressed against the index returns, a regression line can be attained as follows:

$$r_{it} = \alpha_i + \beta_i r_{ind,t} + \epsilon_{it}, \ i = 1, \ldots, N,$$

where $\alpha_i$ and $\beta_i$ are the intercept and slope respectively and $\epsilon_{it}$ denotes the residual. Adding up the regression lines for all assets weighted by the portfolio weights, we have

$$\sum_{i=1}^{N} w_i r_{it} = \sum_{i=1}^{N} w_i \alpha_i + \sum_{i=1}^{N} w_i \beta_i r_{ind,t} + \sum_{i=1}^{N} w_i \epsilon_{it},$$

where the left-hand-side of the equation becomes the ITP return $r_{p,t}$. Defining the regression intercept and slope of the regression line when we regress the returns of the ITP against the index returns as

$$\alpha(x) \triangleq \sum_{i=1}^{N} w_i \alpha_i \quad \text{and} \quad \beta(x) \triangleq \sum_{i=1}^{N} w_i \beta_i,$$

respectively, note that portfolio weight is with $x \triangleq [x_1, \ldots, x_N]^T$ and $e_{p,t} \triangleq \sum_{i=1}^{N} w_i \epsilon_{it}$, we have

$$r_{p,t} = \alpha(x) + \beta(x) r_{ind,t} + e_{p,t}.$$

Since the ideal ITP should have $\alpha(x) = 0$ and $\beta(x) = 1$, by considering the investment upper and lower position constraints [1], the ITP design problem can be expressed as follows:

$$\begin{align*}
\min_{x} & \lambda | \alpha(x) - 0 | + (1 - \lambda) | \beta(x) - 1 | \\
\text{subject to} & \varepsilon_i \leq x_i p_{iT} / B, \ i = 1, \ldots, N \\
& \delta_i \geq x_i p_{iT} / B, \ i = 1, \ldots, N \\
& \sum_{i=1}^{N} x_i p_{iT} / B = 1 \\
& x_i \in \mathbb{Z}, \ i = 1, \ldots, N
\end{align*}$$

where $\lambda \in (0, 1)$ is a predefined weighting parameter and constants $\varepsilon_i$ and $\delta_i$ are the least and largest proportions we can hold on asset $i$. Introducing variables $s \triangleq | \alpha(x) - 0 |$ and $t \triangleq | \beta(x) - 1 |$, we have the following equivalent epigraph form of problem (6) as

$$\begin{align*}
\min_{x,s,t} & \lambda s + (1 - \lambda) t \\
\text{subject to} & s \geq \alpha(x) \\
& s \geq -\alpha(x) \\
& t \geq \beta(x) - 1 \\
& t \geq -(\beta(x) - 1) \\
& x \in \mathcal{X},
\end{align*}$$

where the objective is linear in $s$ and $t$.

B. A MIP Formulation for ITP Design

As mentioned before, a practical ITP cannot invest in all the substituent assets in an index. To limit the number of assets, we introduce binary variables $z_i \in \{0, 1\}, i = 1, 2, \ldots, N$, where $z_i = 1$ denotes asset $i$ is selected by the ITP and $z_i = 0$ denotes asset $i$ is deselected by the ITP. Considering the asset selection variables, the ITP design problem (7) is transformed to the following MIP formulation:

$$\begin{align*}
\min_{x,z,s,t} & \lambda s + (1 - \lambda) t \\
\text{subject to} & s \geq \alpha(x) \\
& s \geq -\alpha(x) \\
& t \geq \beta(x) - 1 \\
& t \geq -(\beta(x) - 1) \\
& \sum_{i=1}^{N} x_i p_{iT} / B = 1 \\
& x_i \in \mathbb{Z}, \ i = 1, \ldots, N \\
& \sum_{i=1}^{N} z_i = k \\
& z_i \in \{0, 1\}, \ i = 1, \ldots, N
\end{align*}$$

where $k$ is the number of assets to be chosen in the ITP. It is easy to verify that problem (8) is a MIP. It can be solved by off-the-shelf solvers like CPLEX [7], however, the computational burden can be huge when the problem size (i.e., $N$) is big [1]. To efficiently solve problem (8), a deep learning-aided approach will be put forward.

III. A LEARNING-AIDED ITP DESIGN APPROACH

In this section, a learning-aided approach will be proposed to solve problem (8). We will firstly give a brief discussion on the graph structure constructed for asset selection, and then the detailed solving approach will be outlined.

Note that the proposed methods in this paper are applicable not only to the index tracking portfolio, but also to other tracking portfolios in finance.
A. Asset Selection via a Calibrated GCN

The asset selection stage is to decide whether an asset in the index is selected or deselected in the designed ITP. In this paper, we will use the tripartite graph profile as in [19]. A tripartite graph $G = \{V, E\}$ is constructed to represent the MIP formulation (8). It contains three sets of vertices: the variable vertex set $V_V$, the constraint vertex set $V_C$, and the objective vertex set $V_O$. And there exists an edge between two vertices from different vertex set if their corresponding coefficient in (8) is not zero. To extract the information of MIP formulation in a tripartite graph, we collect a class of features for each variable vertex $v_i$, $\forall i \in V_V$ (the variable type and the statistics of variable coefficients in constraints and objective), each constraint vertex $v_i$, $\forall i \in V_C$ (the constraint type and the statistics of coefficients of the constraint entries) and each edge $e_j$, $\forall j \in E$ (the corresponding coefficient between vertices connected by the edge). And for the edge between a constraint and objective, the corresponding coefficient of the constraint in the objective is obtained from the dual problem of (8).

For details on the graph construction and features collection, please refer to [11], [19], [20].

A calibrated GCN is used to realize the asset selection target, i.e., to predict the value of the binary variables $z_i, \forall i \in \{0, 1\}$, $i = 1, 2, \ldots, N$. There are three phases in the forward propagation process of the GCN. The first phase is the graph embedding for vertices and edges. The second phase is the message passing stage processed among different vertices. And the prediction of binary variables is conducted in the last phase. The overall procedure of forward propagation of the GCN is demonstrated in Algorithm 1.

In Algorithm 1, $\sigma$ and $\sigma_t$ denote the ELU (·) function and LeakyReLU (·) function, and CONCAT (·) denotes the concatenation operation that joins two feature arrays, $\mathbf{a}$ and $\mathbf{W}$ are the parameters to be trained, and $V_z$ denotes the vertices set of $z_i, i = 1, 2, \ldots, N$. It should be mentioned that the embedding layer for different types of vertices and edges are different. And to recognize the different extent of influence to each vertex from its different neighbors, we apply the graph attention [17] idea and introduce graph attention coefficient $\alpha_{ij}, \forall j \in V, \forall i \in N_j$, where $N_j$ denotes the neighbors set of vertex $j$. In the message passing process, $T$ is a predefined constant that indicates the number of iterations, and there are six steps for information exchanging among the vertices. Finally, we apply two fully-connected layers with a sigmoid function only for those binary variable vertex embeddings and get the final output. And the GCN is trained by minimizing the binary cross-entropy loss.

After the prediction of binary variables $z_i, \forall i \in \{0, 1\}$, $i = 1, 2, \ldots, N$ using GCN, we can sort the output of GCN in descending order. For the first $k$ assets we set $z_i = 1$, i.e., they are active in the ITP and for the remaining assets, we set $z_i = x_i = 0$.

B. Asset Allocation via a Lightweight MIP

After the asset selection stage, the ITP design problem suffices to decide the dollar proportion invested in each selected asset, which is called the asset allocation stage. In comparison to the $2N$ integer variables and the 2 continuous variables involved in the original MIP, the resulting MIP problem becomes a much easier one with only $k$ integer variables and 2 continuous variables, in which case the computation burden is reduced significantly.

Finally, we outline the whole process of the deep learning-aided ITP design scheme in Fig. 1.

IV. NUMERICAL SIMULATIONS

In this section, the performance of our proposed learning-aided approach for ITP design will be presented via numerical simulations on synthetic datasets. The famous MIP solver CPLEX will be used for training data generation and performance comparison.
A. Synthetic Dataset Generation

In finance, an index can be capitalization-weighted or price-weighted [1] where the latter one will be used in this paper. Assuming there is an index consists of \( N \) assets, then we can generate the return \( r_{it} \) of asset \( i \) (\( i = 1, 2, \ldots, N \)) over a time period \( T \) from a normal distribution with mean of 0 and variance of \( \sigma^2 = 0.01 \) in this paper. If the initial price for asset \( i \) is \( p_{i0} \), its price at time \( t \) is calculated by

\[
p_{it} = \prod_{i=1}^{t} (r_{it} + 1) p_{i0}.
\]  

Then the price \( p_{ind,t} \) of the price-weighted index at time \( t \) is

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

and the index return at time \( t \) is given by

\[
r_{ind,t} = \frac{p_{ind,t} - p_{ind,t-1}}{p_{ind,t-1}}, \quad t = 1, \ldots, T.
\]

After the data generation process, the ordinary least-squares method is used to calculate the linear regression intercept \( \alpha_i \) and slope \( \beta_i \) for the asset returns against the index returns. And then we apply CPLEX for the solving of problem (8) to generate the GCN training cases.

B. Numerical Evaluations on Asset Selection

To evaluate the asset selection performance, an ITP design problem with \( N = 30 \) and \( k = 10 \) is conducted. We generate 500 instances, of which 300 instances are for training, 100 instances are for evaluating, and 100 instances are for testing. The objective coefficient is set as \( \lambda = 0.5 \) and the number of iterations is set as \( T = 16 \). The performance of asset selection via GCN is presented in Table I. Results show that the GCN method for asset selection can attain a good generalization performance, given that the accuracy (the number of the ground-truth cases in the GCN results divided by \( N \)) and precision (the number of successfully selected assets in the GCN results divided by \( k \)) measurements for the testing dataset are higher than the training dataset.

<table>
<thead>
<tr>
<th>Instance</th>
<th>Accuracy</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>0.875</td>
<td>0.812</td>
</tr>
<tr>
<td>Evaluate</td>
<td>0.894</td>
<td>0.848</td>
</tr>
<tr>
<td>Test</td>
<td>0.883</td>
<td>0.822</td>
</tr>
</tbody>
</table>

In paper [19], GCN is used to predict binary variables for eight different MIP problems, namely fixed charge network flow (FCNF), capacitated facility location (CFL), generalized assignment (GA), maximal independent set (MIS), multi-dimensional knapsack (MK), set covering (SC), traveling salesman problem (TSP), and vehicle routing problem (VRP). The average precision (AP) [22] is used to measure the precision and recall of the sorted results which is defined as

\[
AP = \sum_{l=1}^{N} P_l \Delta R_l,
\]

where \( l \) denotes the rank of the predicted binary variables list after sorting, \( P_l \) is the precision at cut-off \( l \) in the list, and \( \Delta R_l \) is the difference in recall from cut-off \( l - 1 \) to cut-off \( l \) in the list. The AP values from paper [19] and our experiments are presented in Table II. It can be seen that using basic features (variable descriptions) or basic features and structure features (statistics of the variable coefficients in constraints and objective), only MK attains higher AP than ITP which is solved via our learning-based approach. With all features (57 for variable and 26 for constraint) used, only half of the problems attain higher AP than our ITP problem, while we use much fewer features (17 for variable and 11 for constraint). This indicates that our calibrated GCN is effective to extract higher hierarchical features, and also indicates that the prediction of binary variables in ITP problem is doable.

<table>
<thead>
<tr>
<th>Instances</th>
<th>Basic</th>
<th>Basic and Structure</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCNF</td>
<td>0.261</td>
<td>0.317</td>
<td>0.788</td>
</tr>
<tr>
<td>CFL</td>
<td>0.590</td>
<td>0.629</td>
<td>0.850</td>
</tr>
<tr>
<td>GA</td>
<td>0.744</td>
<td>0.797</td>
<td>0.937</td>
</tr>
<tr>
<td>MIS</td>
<td>0.355</td>
<td>0.337</td>
<td>0.325</td>
</tr>
<tr>
<td>MK</td>
<td>0.840</td>
<td>0.843</td>
<td>0.927</td>
</tr>
<tr>
<td>SC</td>
<td>0.748</td>
<td>0.753</td>
<td>0.959</td>
</tr>
<tr>
<td>TSP</td>
<td>0.358</td>
<td>0.353</td>
<td>0.413</td>
</tr>
<tr>
<td>VRP</td>
<td>0.403</td>
<td>0.424</td>
<td>0.459</td>
</tr>
<tr>
<td>ITP</td>
<td>-</td>
<td>-</td>
<td>0.808</td>
</tr>
</tbody>
</table>

Despite the performance of asset selection shown in Table I, we further discuss the reason for the misselections. First, we sort the assets based on labels \( x_i \) in descending order, which represents the position we would like to hold in the optimal ITP (ITP designed from CPLEX). Then record the possibility that an asset with sequence \( i \) would be selected in the ITP obtained by GCN. The distribution of the assets selected in ITP is presented in Fig. 2, we can see that the asset with the largest \( x_i \) are positioned in our ITP in all cases, while for the asset with the tenth-largest \( x_i \), only has 70% possibility to be held in our ITP. The distribution of assets elaborates that the assets with larger positions are more distinctive and more predictable, and the assets with fewer positions are not easy to distinguish from other assets that should not be held in the optimal ITP. So, for some more diverse markets, we suggest our GCN should have better performance.

C. Numerical Evaluations on Asset Allocation

After examining the asset selection performance of GCN, the asset allocation performance compared to CPLEX will be investigated. For comparisons, several synthetic index cases are generated, namely, Case 1 \( (N = 50, k = 10) \), Case 2 \( (N = 50, k = 20) \), and Case 3 \( (N = 100, k = 30) \). For each
the possibility of being selected

in terms of computation time, despite the slight inferiority in all instances and outperforms the CPLEX method significantly. While our learning-based portfolio design method works for simulations to be 1800s, the CPLEX method sometimes cannot solve by standard solvers. Numerical simulations show that the MSE values remain on the same scale which is independent of the dimension of indices, indicating that our learning-based ITP design scheme is amenable to high-dimensional index tracking problems.

Finally, comparisons on runtime are presented in Table IV. Since we set the upper bound for runtime in the CPLEX simulations to be 1800s, the CPLEX method sometimes cannot output a converging solution for the ITP design problem. While our learning-based portfolio design method works for all instances and outperforms the CPLEX method significantly in terms of computation time, despite the slight inferiority in the solution quality.

V. CONCLUSIONS

A deep learning-aided approach has been proposed for index tracking portfolio design in this paper. The index tracking portfolio design problem is first formulated as a MIP. Then a calibrated graph convolutional network is applied for asset selection. After that, a lightweight MIP is obtained which is solved by standard solvers. Numerical simulations show that the proposed learning-aided approach is favorable for practical index tracking targets in finance.

TABLE IV

<table>
<thead>
<tr>
<th>kN</th>
<th>Approach</th>
<th>Runtime</th>
<th>S.D. of Runtime</th>
<th>Succ. Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/30</td>
<td>CPLEX</td>
<td>37.2641s</td>
<td>93.4973</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>GCN</td>
<td>8.2652s</td>
<td>6.9944</td>
<td>100%</td>
</tr>
<tr>
<td>10/50</td>
<td>CPLEX</td>
<td>130.4864s</td>
<td>138.4955</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>GCN</td>
<td>11.3447s</td>
<td>24.4520</td>
<td>100%</td>
</tr>
<tr>
<td>20/50</td>
<td>CPLEX</td>
<td>264.0888s</td>
<td>470.9651</td>
<td>95%</td>
</tr>
<tr>
<td></td>
<td>GCN</td>
<td>9.2106s</td>
<td>12.6044</td>
<td>100%</td>
</tr>
<tr>
<td>30/100</td>
<td>CPLEX</td>
<td>590.5623s</td>
<td>482.2835</td>
<td>92.5%</td>
</tr>
<tr>
<td></td>
<td>GCN</td>
<td>10.0451s</td>
<td>9.3661</td>
<td>100%</td>
</tr>
</tbody>
</table>

REFERENCES