Technical Report – Portfolio Optimization with Physics-Inspired Graph Neural Networks

Vertiefungsprojekt II (FS23), Master of Eng. in Data Science

Student: Roman Loop

Academic Supervisor: Shao Jü Woo

Project duration: 10. March 2023 to 14. July 2023

Contents

Prephase

This work is strongly based on the recent publication "*Combinatorial Optimization with Physics-Inspired Graph Neural Networks*" by Schuetz et al.

Introduction

Optimization is ubiquitous in science and industry. In particular, the field of combinatorial optimization—the search for the minimum of an objective function within a finite, but often large, set of candidate solutions—is one of the most important areas of optimization with practical (but notoriously difficult) applications found in virtually every industry, including both the private and public sectors, and in areas such as transportation and logistics, telecommunications, and finance.

A well-known optimization problem in finance is risk diversification of a portfolio. The optimization objective is to construct a portfolio, whose constituents consist of a subset of *n* assets selected from a large universe of *N* assets, exhibiting the lowest possible risk. These minimum risk portfolios are of great interest to investors that are relatively risk averse. In practice, the universe can potentially be very large. The *N* assets in the universe can be modelled as a graph in which the assets are represented by the vertices and their relationships are derived from the correlations between the assets. The correlations play a pivotal role in finding minimum risk portfolios since the overall portfolio risk is determined by them. In the ideal case, the minimum risk portfolio contains assets that are uncorrelated.

When constructing low risk portfolios, one may strive for finding the largest possible set of uncorrelated assets. From a computational point of view this poses a major challenge as the set of possible solution candidates grows exponentially with the number of nodes. This implies that a brute force approach is infeasible for large sets. There are approximation algorithms, but none of them scale to graphs with thousands or even hundreds of thousands of nodes.

Schuetz et al. developed a self-supervised deep learning approach to approximate the maximal independent set (MIS) on large graphs. Their Graph Neural Networks (GNNs) based approach is radically different from traditional operations research approaches and allows solving canonical NP-hard problems in the form of quadratic unconstrained binary optimization problems. GNNs are fundamentally deep neural network architectures specifically designed for graph structure data, with the ability to learn effective feature representations of nodes, edges, or even entire graphs. They have recently become very popular due to their ability to learn complex systems of relations or interactions arising in a broad spectrum of problems. The proposed approach claims to be very fast and accurate. The promising solution runtime of $\sim n^{1.7}$ scales very well compared to the solution runtime of the Boppana-Halldorsson algorithm – a state of the art algorithm for the MIS problem – with a solution runtime of $\sim n^{2.9}$.

The main objective of this project is to apply the approach of Schuetz et al. to the S&P500 assets to construct a minimum risk portfolio consisting of lowly correlated assets and compare its volatility and performance against benchmarks.

Basic theory

In this section, the most relevant basic theories are covered. First, the main points of the paper “Combinatorial Optimization with Physics-Inspired Graph Neural Networks” by Schuetz et al. are summarised. In particular, the end-to-end process and the problem formulation within the QUBO framework

are highlighted. Furthermore, the basics of graph theory and graph neural networks (GNN) are explained.

Summary of "Combinatorial Optimization with Physics-Inspired Graph Neural Networks" by Schuetz et al.

Schuetz et al. propose a highly scalable unsupervised GNN approach for solving QUBO and PUBO problems. Figure 1 describes the end-to-end process of their approach. It consists of five steps:

a. **Problem setup**: Create an input graph, a differentiable Q matrix and the cost function.
b. **Training strategy**: Define the GNN architecture and the hyperparameters.
c. **GNN training**: Train the GNN against the problem specific loss function.
d. **Projection scheme**: Apply a projection scheme (hard thresholding) to convert probabilities into binary variables.
e. **Final evaluation**: Evaluate the result and check for possible violations.

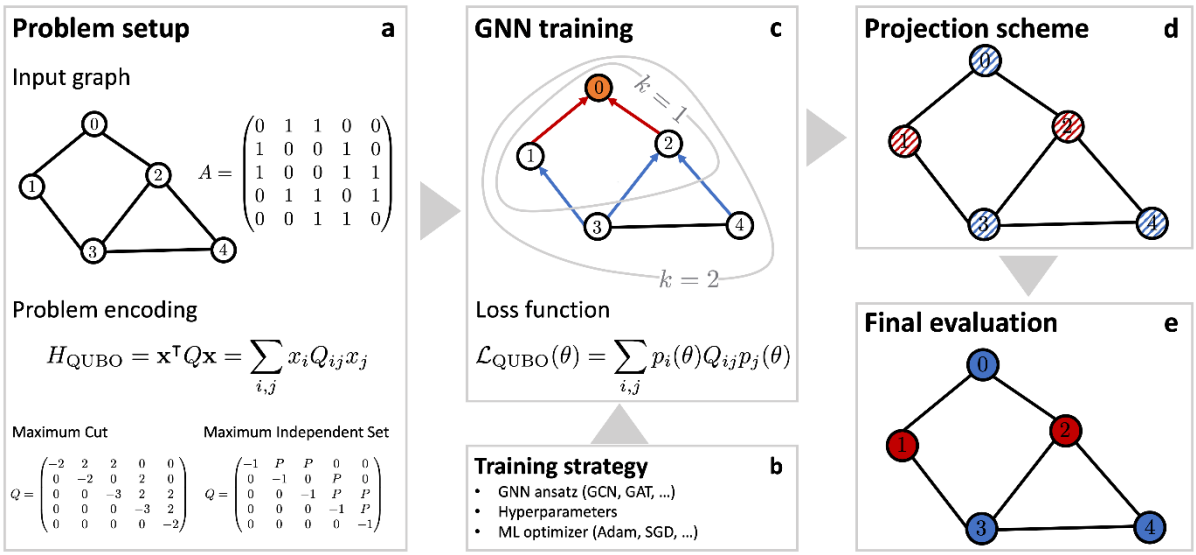


Figure 1: End-to-end process from Schuetz et al.

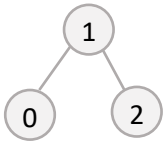
The QUBO framework

The Quadratic Unconstrained Binary Optimization (QUBO) framework unifies several NP-hard combinatorial optimization problems. The **cost function** for a QUBO problem can be expressed in a compact form:

$$H_{QUBO} = x^T Q x = \sum_{i,j} x_i \, Q_{i,j} x_j$$

where x is a vector of binary decision variables and the QUBO matrix Q is a square matrix encoding the actual problem. For undirected graphs the Q matrix can be assumed to be symmetric and in the upper diagonal form. As an illustration, consider the simplest possible example, a graph with three nodes given by A:

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

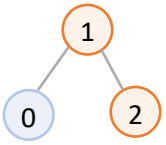


With a penalty term P of 2, the Q matrix for the above graph is given by:

$$Q = \begin{pmatrix} -1 & 2 & 0 \\ 0 & -1 & 2 \\ 0 & 0 & -1 \end{pmatrix}$$

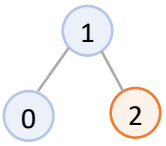
If only node 0 is selected for the MIS, a loss of -1 is obtained. This is a valid independent set, but not the largest.

$$H_{QUBO} = x^T Q x = (1 \quad 0 \quad 0)\begin{pmatrix} -1 & 2 & 0 \\ 0 & -1 & 2 \\ 0 & 0 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = -1$$



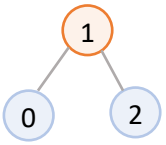
If nodes 0 and 1 are selected, the MIS is violated, resulting in a loss of 0.

$$H_{QUBO} = x^T Q x = (1 \quad 1 \quad 0)\begin{pmatrix} -1 & 2 & 0 \\ 0 & -1 & 2 \\ 0 & 0 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = 0$$



However, if we select nodes 0 and 2 for the MIS, the loss becomes -2.

$$H_{QUBO} = x^T Q x = (1 \quad 0 \quad 1)\begin{pmatrix} -1 & 2 & 0 \\ 0 & -1 & 2 \\ 0 & 0 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = -2$$



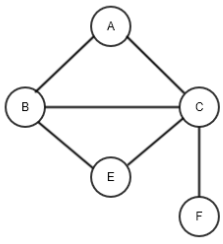
We have found the optimal solution for this example.

Graph theory

The basic elements of any graph are nodes and edges. Nodes (or vertices) can represent a variety of different objects, such as people, items, places and more. Edges show how the nodes are connected. A simple graph consists of nodes and edges of only one type. Let's say the nodes represent people and the edges show whether person A knows person B.

Edges can be directed or undirected. Person A might know person B, but person B has never heard of person A – that's a directed edge. An undirected edge might be "married". If person A is married to person B, then person B is automatically married to person A – at least in Switzerland. As we can see from these examples, some edge types are naturally directed and others undirected. However, any directed graph can be transformed into an undirected graph by adding inverted edges.

Figure 2 shows a simple graph with nodes A to F. These nodes are connected by undirected edges. Let's stick with the example of people who know each other. Person A knows person B and C, person B and C know person A and others. These relationships can be converted into an adjacency matrix, where each row and each column represent a person. The matrix values indicate whether or not there is an edge between two nodes.



Adjacency Matrix

	A	B	C	E	F
A	0	1	1	0	0
B	1	0	1	1	0
C	1	1	0	1	1
E	0	1	1	0	0
F	0	0	1	0	0

Figure 2: Simple graph with adjacency matrix. Source: https://www.oreilly.com/

If the matrix values are Boolean, we call it an unweighted graph. As we can imagine, the edges could have a numerical value. Think of cities and the flight distances between them. An edge could hold information about how long the distance is. In the adjacency matrix we would see numerical values instead of Booleans – this would be a weighted graph.

For a portfolio optimization use case, an undirected and unweighted graph is constructed. Each node represents a stock – or more generally a financial asset – and the edges between nodes indicate, that the daily stocks returns are correlated with each other.

Properties of graph data

Graph data is different from most other data structures we know in machine learning. Two properties that set graph data apart from images or tabular data.

- **Arbitrary size and shape**: It might be argued that this is also true for image data, for example. But images can easily be resized, padded or cropped to the same size. Such operations are not defined for graph data. Additional nodes or edges cannot be removed. Therefore, methods are required that can handle arbitrary input sizes and shapes.
- **Permutation invariance:** Graphs that look different can still be structurally identical. If an image is flipped, the result is a different image. However, flipping a graph only changes the order of nodes, but its structure is still the same. Therefore, algorithms that deal with graph data must be permutation invariant.

Graph Neural Networks (GNN)

In recent years, GNNs have emerged and established themselves as state of the art models in many fields. GNNs can learn appropriate representations of graph data to solve many graph problems such as node classification, link prediction or graph level prediction.

Message Passing Layers (MPLs) form the core of any GNN. A MPL collects information about the neighbourhood of a node, aggregates this information and updates the current node embedding with the new information. This approach is also called graph convolution and can be seen as an extension of convolutions on graph data.

Figure 3 shows on the left-hand side an image convolution and a graph convolution on the right-hand side. In the case of images we can slide a learnable kernel over the grid structure of an image, which then extracts the most important information. This can also be seen as combining the information from a neighbourhood in a local area.

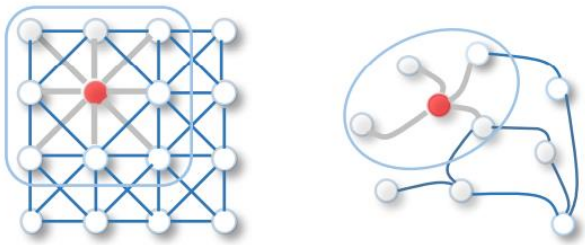


Figure 3: 2D-convolution vs. graph convolution. Source: Wu et al., 2019, A comprehensive Survey on Graph Neural Networks, https://arxiv.org/pdf/1901.00596.pdf

For graph convolutions, this idea is extended. Instead of kernels, we simply combine the information of neighbouring nodes (and edges) and create new node embeddings including the neighbourhood information.

Figure 4 illustrates how message passing works. In this example graph we have one yellow node, three blue nodes and one green node. After one message passing layer, the yellow node's embedding contains information about its blue neighbours. After a second MPL, the embedding of node one also contains information about the green node, or in other words, it contains information about its neighbours of its neighbours. This knowledge is stored in the node embeddings. These embeddings contain knowledge about the structure of the graph and about the properties of the nodes.

Thus, the more MPLs, the larger the neighbourhood. The number of MPLs in a model is an important hyperparameter and must be chosen carefully. It depends on the learning task and the graph data whether a rather small or rather large neighbourhood is relevant. A well-known problem is over-smoothing. In Figure 4 we see that after two MPLs *Node 1* knows something about all other nodes in the graph. More layers would lead to over-smoothing and result in bad node embeddings.

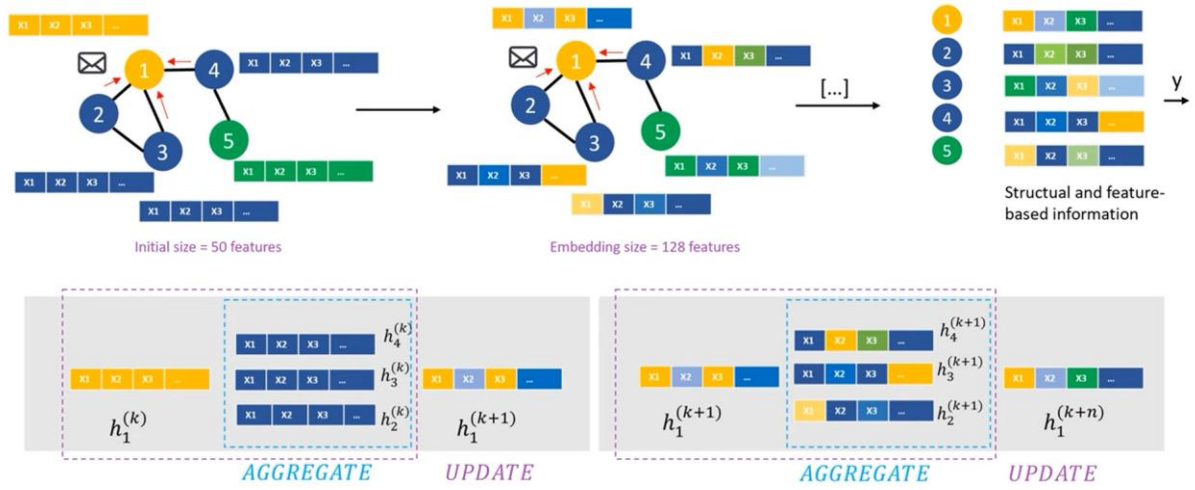


Figure 4: Illustration of message passing.
Source: DeepFinr, Understanding Graph Neural Networks | Part 2/3, Youtube

Physics-Inspired GNN approach for low-risk portfolio optimization

The main goal of the project is to adapt Schuetz et. al's approach of "Combinatorial Optimization with Physics-Inspired Graph Neural Networks" to the use case of portfolio optimization with the objective of creating a low-risk portfolio. More specifically, the goal is to find a maximum independent set of stocks among all S&P500 stocks. The hypothesis is that the volatility of a portfolio of independent stocks is lower than the volatility of a portfolio of highly correlated stocks.

A secondary objective is to test and compare different measures of correlation. Specifically, Pearson's correlation, distance correlation and quantile correlation will be compared. Each of these correlation measures have unique characteristics. We want to find the measure that best serves the overall goal of reducing a portfolio's risk.

In this section, the portfolio optimization use case we have chosen for this project will be described. First, the data set will be explained in detail. Then the preprocessing steps including the graph construction will be described. Next, the GNN setup and its training will be covered and finally the evaluation and conclusion will be discussed.

Dataset

In this project we decided to limit the **asset universe** to stocks in the Standard and Poor's 500 index. The S&P500 is a stock market index tracking the returns of the 500 largest companies listed on US stock exchanges. The index includes about 80% of the US equity market by capitalization and is one of the most widely known stock indices. It includes some of the largest companies in the world such as Apple, Microsoft, Amazon, Johnson & Johnson, ExxonMobil and many others. A full list of the stocks included in the S&P500 can be found on Wikipedia.

Fortunately, historical asset price data is widely available for free. Yahoo Finance is one of the providers of historical financial data, and also comes with its own Python library for downloading asset data.

The daily adjusted closing prices from 2014 to 2023 have been used. Historical data does not go back to 2014 for all S&P500 stocks. Although, missing data can be handled to calculate the correlation measures. It becomes difficult to handle in the backtest. Therefore, stocks that do not have the full history of price data are dropped from the dataset, which was about 5% of the stocks.

Data-Preprocessing

In this section, the transformation and computational steps are described that were necessary to transform the raw data into a correlation graph and a QUBO matrix on which a GNN can be trained. The main steps of the transformation are:

1. Transform the price time series into returns and split into train and test sets.
2. Compute the correlation matrices.
3. Binarize the correlation matrices.
4. Generate the QUBO matrices and the graphs.

Since we want to compare different correlation measures, the steps two to four must be proceeded separately for each correlation measure.

Steps 1 (Transform the price time series into returns and split into train and test sets): The actual prices were converted into daily returns for two reasons. First, it makes the time series stationary and second, it reduces the overall correlation.

Next, the time series was split into a training and two testing sets. The training set covers the period from 2014 to 2019. The first testing set covers the period from 2019 to 2021. As during the training period, we have experienced a bullish market segment during the first testing period. During the period from 2021 to 2023 the market segment has changed from a bullish to a stagnating or even slightly bearish segment. Therefore, it makes sense to use the period from 2021 to 2023 as a second testing set, as the market segment has changed. Figure 5 shows the development of the S&P500 index during the train, first and second testing period.

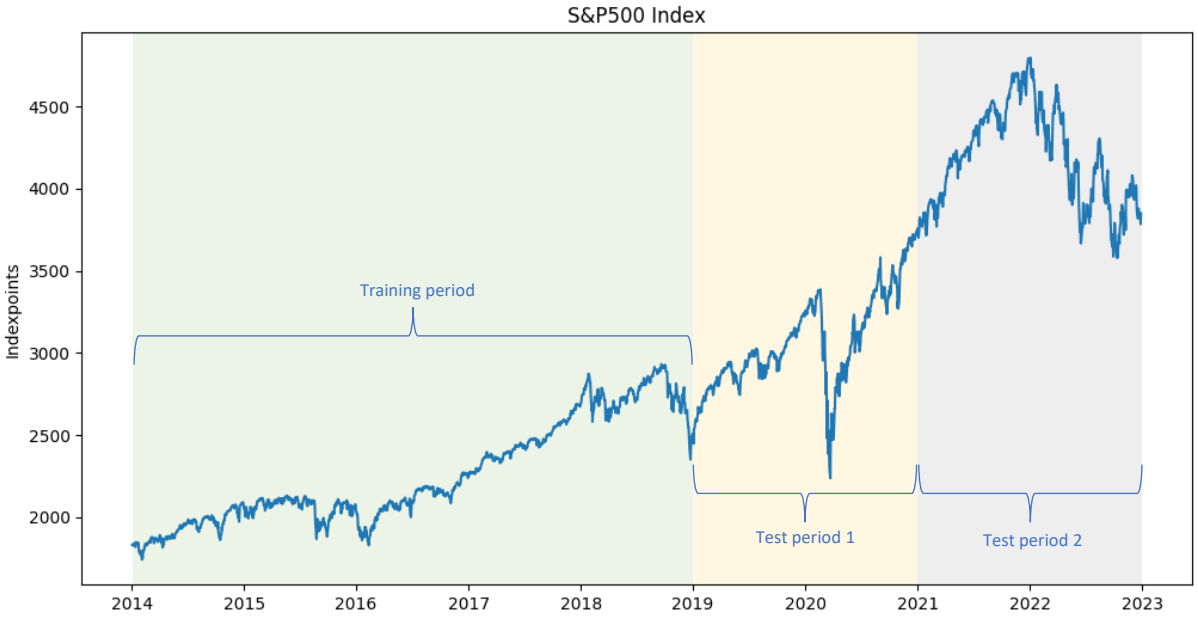


Figure 5: Train-test split.

Step 2 (Compute the correlation matrix): The use of an appropriate correlation measure to compute the correlation matrix is critical. We decided to test and compare the following correlation measures:

1. The **Pearson's correlation coefficient** is a widely used measure of linear correlation. Mathematically, it is defined as “the covariance between two vectors normalized by the product of their standard deviations”.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

2. The **distance correlation coefficient** is defined as the "distance covariance" normalized by the "distance standard deviation". Rather than assessing how two variables tend to covary in their distance from their respective means, the distance correlation assesses how they tend to covary in their distances from all other points. This has the potential to better capture non-linear dependencies between variables.

$$dCor(X,Y) = \frac{dCov(X,\ Y)}{\sqrt{dVar(X)\ dVar(Y)}}$$

3. The **quantile correlation coefficient** is defined as the geometric mean of two quantile regression slopes – that of X on Y and that of Y on X – in the same way that Pearson's correlation coefficient is related to regression coefficients. The quantile correlation is a measure of the overall sensitivity of a conditional quantile of a random variable to changes in the other variable.

$$\rho_\tau^{X,Y} = \begin{cases} sign(\beta_{2.1}(\tau))\sqrt{\beta_{2.1}(\tau)\beta_{1.2}(\tau)}, & \text{if } \beta_{2.1}(\tau)\beta_{1.2}(\tau) \geq 0, \\ 0, & \text{otherwise,} \end{cases}$$

Four different correlation matrices have been computed based on the above-mentioned correlation measures. Python's Pandas library provides a very simple and efficient way to compute the Pearson's correlation coefficient on a Pandas DataFrame object. To compute the distance correlation, the Python library *dcor* has been used which provides a computationally efficient method to compute the distance correlation between two arrays. The quantile correlation was the most difficult and least efficient. The *statsmodels.formula.api.quantreg* method has been used to compute the quantile slopes while the remaining part of the formula was implemented using *numpy*. However, this implementation does not scale to large datasets. It took about 2.5 hours to compute the quantile correlation matrix for our 470 stocks. This computational time compares unfavourably with the Pearson's correlation matrix which was computed in sub seconds.

Daily returns of the trainings set are used to calculate the correlation matrix. Table 1 shows exemplarily the correlation matrix. The final correlation matrix is a square matrix with dimensions 470 by 470.

	AAPL	**ADBE**	**…**	**XEL**	**ZTS**
AAPL	1.000000	0.406557	…	0.212159	0.298356
ADBE	0.406557	1.000000	…	0.293385	0.363729
…	…	…	…	…	…
XEL	0.212159	0.293385	…	1.000000	0.141236
ZTS	0.298356	0.363729	…	0.141236	1.000000

Table 1: Correlation matrix (example).

The correlation matrix can be thought of as the adjacency matrix and is therefore the basis for constructing a graph. It is therefore important to use a correlation measure that serves the purpose of reducing the volatility of a portfolio.

Step 3 (Binarize the correlation matrix): To binarize the correlation matrix, a threshold was defined and correlations above or equal to the threshold are assigned a value of 1 and a value of 0 otherwise. The result is an adjacency matrix where highly correlated stocks are connected through edges. In

general, the higher the threshold, the lower the average degree[1] of the graph and the larger the MIS. The optimal value for the threshold should be evaluated by means of backtesting. However, this is beyond the scope of this project, so the decision was made to use a fixed threshold of 0.35.

Step 4 (Generate the QUBO matrix and the graph): The *NetworkX* library has been used to create graphs from the adjacency matrices and just removed the self-loops. Figure 6 shows the node degree distribution of the four different graphs.

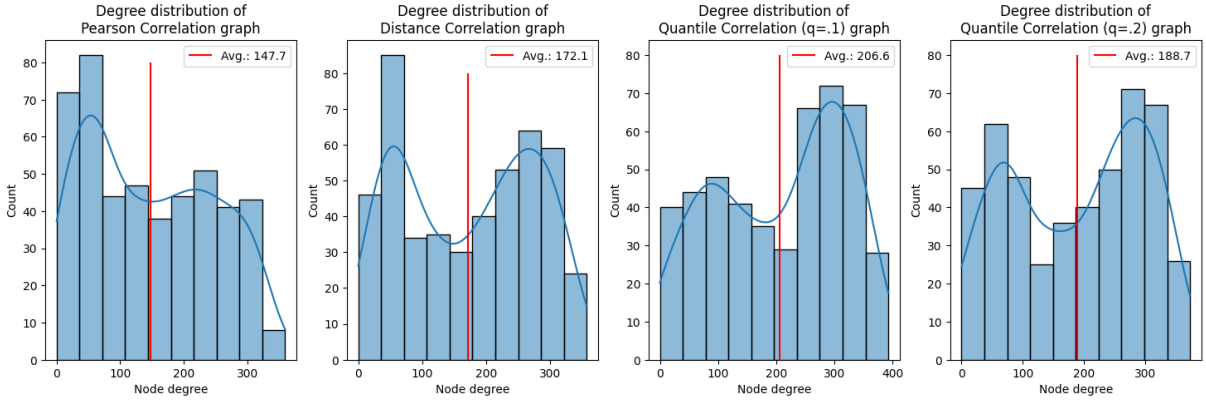


Figure 6: Node degree distribution

Depending on the correlation measure, the resulting graph looks different. The graphs have an average degree between 147.7 and 206.6, and for all graphs the degree distribution is asymmetric. These graph structures are fundamentally different from those used by Schuetz et al. in their paper. The authors used 3-regular and 5-regular graphs. In graph theory a regular graph is a graph where each node has the same number of neighbours. Thus, in a 3-regular graph each node has three neighbours and each node has a degree of three.

Figure 7 shows the correlation graphs. Low degree nodes are located at the edge while high degree nodes are placed in the centre. There are between 34'000-45'000 edges in the graphs, which makes it almost impossible to create a visualization that clearly shows which stocks are correlated. However, a closer look reveals that the graph structures are slightly different. For the sake of clarity, the fourth graph, which is based on the quantile correlation (q=.1) has not been plotted.

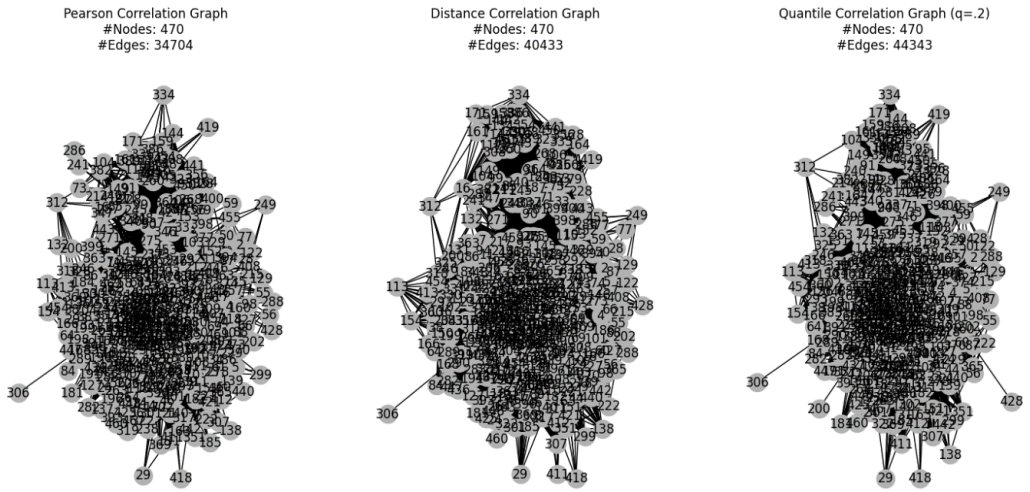


Figure 7: Correlation Graphs

[1] The degree of a node is the number of connections that it has to other nodes in the graph.

Before the GNN can be trained, the optimization problem must be encoded in a QUBO matrix Q. For the graphs shown above, the MIS problem encoding proposed by Schuetz et al. has been used. In unsupervised machine learning rewards and penalties are often used to reward good and punish bad behaviour. A reward value of -1 and a penalty value of 2 have been applied in this project. That means the loss is reduced by -1 for each node which gets added to the MIS. However, for each added node that violates the independence constraint the loss gets penalized by 2.

Although this problem encoding is appropriate to test whether the approach of Schuetz et al. can approximate a good MIS, it may not be optimal for optimizing a portfolio. Risk may not be the only criterion for determining an optimal portfolio. For investors, the returns of the portfolio are also important. Therefore, to account for the return the Q matrix was modified accordingly. Instead of assigning static rewards of -1 the negative average Sortino ratio of the training period was used as a reward.

The Sortino ratio is a well-known portfolio evaluation metric that measures the ratio of return to downside deviation. As such it captures both risk and return. More details on the Sortino ratio can be found in the chapter “Portfolio backtest”.

In the “Evaluation” chapter this approach is called “Handcrafted”. This is one way to implement (soft) constraints in a QUBO problem.

GNN setup and training

Schuetz et al. used a two-layer GCN architecture based on PyTorch GraphConv units. Between the two GCN layers, ReLU is used as the activation function. The authors also give some heuristics for sizing the layer dimensions d_0 and d_1. If the graph is large ($n > 10^5$) they suggest setting $d_0 = int(\sqrt{n})$ else $d_0 = int(\sqrt[3]{n})$ and $d_1 = int(d_0/2)$.

The architecture of Schuetz et al. was applied with different hyperparameter sets to the input graphs. However, the original model proposed by Schuetz et al. was not able to find a reasonable MIS. Obviously, the architecture of Schuetz et al. works well for 3-regular and 5-regular graphs, as they showed in their paper. The graph structures in this project are quite different as they are not regular and have much higher average degrees.

A search for better architectures and hyperparameter sets was undertaken through a grid-search. This was performed using Ray Tune, an industry standard tool for distributed hyperparameter tuning. We mainly experimented with different layer units, architectures, learning rates and dropout rates. In total, 60 different combinations of architectures and hyperparameter sets have been tested. Table 2 shows the five best performing models along with their hyperparameters.

Run time (s)	Best Loss	MIS Size	# Violations	Dropout-rate	Learning-rate	Model / Architecture
1’074.99	-38.98	39	0	0.05	0.0001	SAGE_2L_Model
1’151.59	-37.00	37	0	0.05	0.0010	SAGE_2L_Model
2’064.49	-36.00	36	0	0.05	0.0010	GAT_1L_2H_Model
1’159.50	-35.00	35	0	0.10	0.0010	SAGE_2L_Model
648.04	-33.00	33	0	0.10	0.0010	SAGE_1L_Model

Table 2: Best performing models and hyperparameter-sets.

The experiments demonstrated that the SAGE unit significantly outperforms the GraphConv unit as well as the Graph-Attention unit. Furthermore, small dropout rates between 5-10% aid in stabilizing the training process and increase the overall performance. Learning rates between 1^{-3} and 1^{-4} showed consistently good results.

Table 3 shows the best performing model and the associated hyperparameter set. This model forms the basis for the model evaluation phase in the next chapter.

Model architecture	Hyperparameters
SAGE_2L_Model((conv1): SAGEConv((feat_drop): Dropout(p=0.05, inplace=False) (fc_pool): Linear(in_features=22, out_features=22, bias=True) (fc_neigh): Linear(in_features=22, out_features=11, bias=False) (fc_self): Linear(in_features=22, out_features=11, bias=True)) (conv2): SAGEConv((feat_drop): Dropout(p=0.05, inplace=False) (fc_pool): Linear(in_features=11, out_features=11, bias=True) (fc_neigh): Linear(in_features=11, out_features=1, bias=False) (fc_self): Linear(in_features=11, out_features=1, bias=True)))	{'lr': 0.0001, 'dim_embedding': 22, 'hidden_dim': 11, 'dropout': 0.05, 'number_classes': 1, 'prob_threshold': 0.5, 'number_epochs': 25000, 'tolerance': 0.0001, 'patience': 1000, 'model': 'SAGE_2L_Model'}

Table 3: Top performing model and hyperparameters.

Model evaluation

The model evaluation consists essentially of two parts. First, the GNN approximated independent sets will be evaluated. Second, we test the hypothesis that the risk of portfolios created from the MIS obtained from a universe whose constituents are the S&P500 stocks is less than that of the S&P500 index.

MIS evaluation

The training of the selected model, shown in Table 3, was carried out for the set of input graphs over 25'000 epochs on a CPU, which took about 230 seconds on average. For each input graph the GNN found a MIS with zero violations. Furthermore, we compared the approximated MIS against the MIS obtained using a traditional solver (Boppana-Halldorsson algorithm). On average the traditional solver took about 4.5 seconds to find a MIS on the input graphs. Table 4 shows the results of the approximated MIS:

	GNN			Boppana-Halldorsson		
	MIS size	Violations	Difference	MIS size	Violations	Difference
Pearson graph	80	0	+6	74	0	-6
DCOR graph	38	0	-10	48	0	+10
Quantile (q=.1) graph	61	0	+2	59	0	-2
Quantile (q=.2) graph	59	0	+2	57	0	-2

Table 4: MIS comparison GNN vs. Boppana-Halldorsson

For three of the four graphs tested, the GNN was able to find a larger MIS than the traditional Boppana-Halldorsson algorithm. Only for the DCOR input graph a smaller MIS was found by the GNN. As with any deep neural network, the GNN can potentially get stuck in a local minimum during the training process. To mitigate this problem Schuetz et al. suggest running the unsupervised learning several times and keeping track of the results. However, due to computational constraints this was beyond the scope of this project.

Overall, the results reveal that the approach of Schuetz et al. works. Hence, it is possible to find good MIS approximations with a GNN. However, the GNN model proposed by the authors is lacking in robustness and handled the graph models in this project poorly. The SAGE model proposed by us is more robust and performed much better on graphs possessing higher degrees.

Finally, an evaluation of the primary benefit of Schuetz et. al's approach – its scalability to large graphs – could not be carried out in this project.

Portfolio backtest

Finally, the uncorrelated portfolios performances are tested against two benchmark portfolios.

- **Benchmark 1**: Equally distributed asset allocation over the hole asset universe (470 stocks).
- **Benchmark 2**: Randomly selected assets, portfolio size equal to the average portfolio size of the uncorrelated portfolios (57 stocks), asset allocation equally distributed, sampled ten times and averaged.

The two benchmarks are not significantly different, so Benchmark 1 will be used in the upcoming figures. Figure 8 shows the cumulative returns for each portfolio over the training period:

Figure 8: Cumulative portfolio returns over the trainings period.

Interestingly, the uncorrelated portfolios (GNN portfolios) outperform the benchmark in terms of returns, which is in a sense the opposite of the hypothesis. We would have expected lower returns but also lower volatility (risk). Not surprisingly, the *Handcrafted* portfolio outperforms the other portfolios. We should not be overwhelmed by the performance of the *Handcrafted* portfolio during the training period, because the (true) returns are encoded in the Q matrix. Thus, the GNN was able to select a portfolio that is not only uncorrelated but also contains the best stocks in terms of their Sortino ratio.

Figure 9 shows the maximum drawdown of the portfolios during the training period:

Figure 9: Portfolio Drawdowns over the trainings period.

The chart shows that the uncorrelated portfolios do not significantly reduce the maximum drawdown. Only the quantile correlation measure (q=.2) slightly reduced the maximum drawdowns. Without having any proof, we suspect that it is nearly impossible to reduce maximum drawdowns on an asset universe consisting only of stocks from the S&P500 index.

Figure 10 shows the cumulative returns for each portfolio over the first test period:

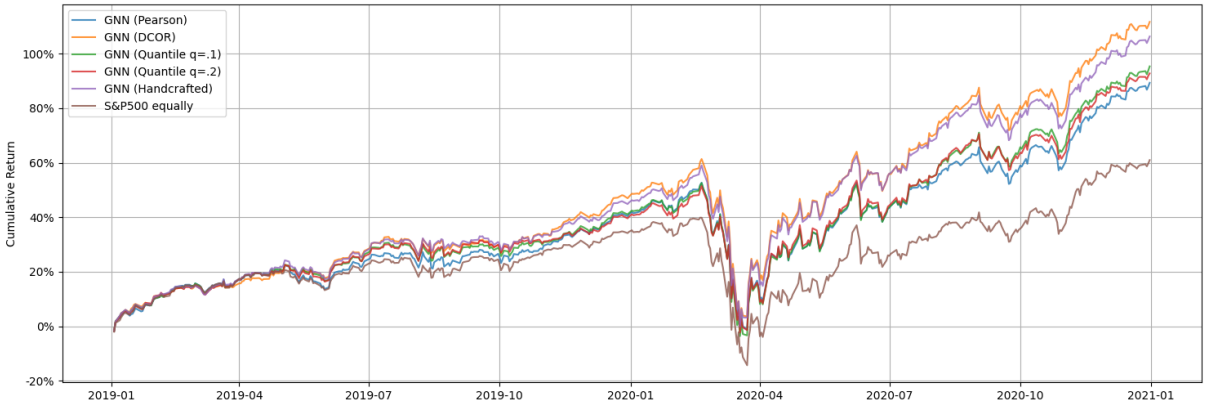


Figure 10: Cumulative portfolio returns over the first test period.

First, we notice that the overall market segment during the first test period is bullish. Hence, we have the same overall market segment as during the training period. Interestingly, the uncorrelated portfolios continue to outperform the benchmark in terms of returns over the first test period. During the Corona crisis market crash in early 2020, the uncorrelated portfolios experienced roughly the same drawdowns as the benchmark portfolio, as shown in Figure 11. However, the portfolio based on the quantile correlation (q=.2) experienced slightly smaller drawdowns during this crisis.

Figure 11: Portfolio Drawdowns over the first test period.

These results do not support the hypothesis that uncorrelated portfolios are less volatile. However, this must be taken with a grain of salt because we have a very limited set of assets in our experiment. First, the asset universe contains only one asset class – namely stocks. Second, the S&P500 index includes only large-cap U.S. stocks. Therefore, the asset universe that was chosen for this project is highly correlated by default and it may be very difficult to create a low volatility portfolio with these assets.

Finally, a second test period (2022-2023) has been chosen because the overall market segment changed from a clearly bullish market in the training and the first test period to a more stagnant and slightly bearish market.

Figure 12 shows the cumulative returns for each portfolio over the second test period:

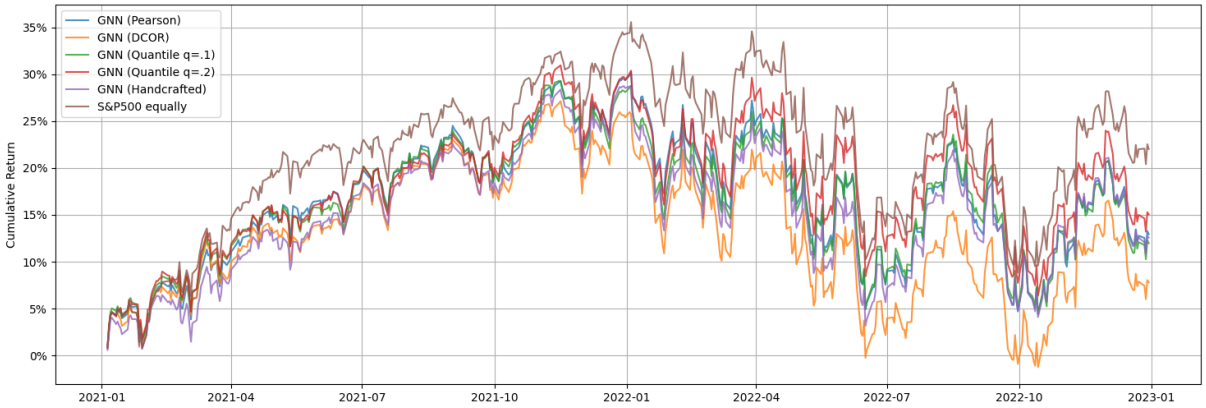


Figure 12: Cumulative portfolio returns over the second test period.

As we can see, the benchmark outperforms the uncorrelated portfolios. One assumption is that correlations are not static. Therefore, it would be interesting to see how the uncorrelated portfolios would have performed over this period if the model would have been retrained with data up to the start of the second test period.

Figure 13 shows the maximum drawdown of the portfolios during the second test period:

Figure 13: Portfolio Drawdowns over the second test period.

It shows the same behaviour as in the first test period – no significant reduction in drawdowns, except for the portfolio based on the quantile correlation (q=.2), which could slightly reduce drawdowns during this period.

Finally, four different performance metrics are calculated by year over the whole period (covering the training, first and second test period). The performance metrics and its definitions are as follows:

R_p = Actual or expected portfolio return
r_f = Risk – free rate
σ_p = Standard deviation of the portfolio's excess return
σ_d = Standard deviation of the downside

- The **Sharpe ratio** compares the return of an investment with the risk taken. The Sharpe ratio's numerator is the difference over time between realized, or expected, returns and a benchmark such as the risk-free rate of return or the performance of a particular investment category. Its denominator is the standard deviation of returns over the same period of time, which is a measure of volatility and risk.

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p}$$

- The **Sortino ratio** is a variation of the Sharpe ratio that differentiates harmful volatility from total overall volatility by using the asset's standard deviation of negative portfolio returns—also refered to as downside deviation—instead of the total standard deviation of portfolio returns.

$$Sortino\ Ratio = \frac{R_p - R_f}{\sigma_d}$$

- **Maximum Drawdown** is the maximum observed loss from a peak to a trough of a portfolio, before a new peak is attained. Maximum drawdown is an indicator of downside risk over a specified time period.

$$MDD = \frac{TroughValue - PeakValue}{PeakValue}$$

- The **Calmar Ratio** indicates the relationship between risk and return. It is a function of the expected annual rate of return and the maximum drawdown over the previous three years.

$$CalmarRatio = \frac{R_p - R_f}{MDD}$$

Table 5 shows the performance indicators. The table is divided into green, yellow and grey sections. These sections correspond to the training, first and second test period. The last column of these sections shows the average over the period. The best scores are highlighted in bold.

Metric	Portfolio	2014	2015	2016	2017	2018	Avg	2019	2020	Avg	2021	2022	Avg
Sharpe	GNN (Pearson)	1.59	0.58	0.92	2.77	-0.22	1.13	2.65	1.00	1.82	1.96	-0.50	0.73
	GNN (DCOR)	1.47	0.45	1.15	3.29	**0.44**	1.36	**3.13**	**1.16**	**2.15**	1.70	-0.53	0.58
	GNN (Quantile q=.1)	1.58	0.41	1.12	3.25	0.16	1.30	2.77	1.05	1.91	1.82	-0.48	0.67
	GNN (Quantile q=.2)	1.56	0.39	1.05	3.30	0.09	1.28	2.69	1.06	1.87	1.87	-0.40	0.73
	GNN (Handcrafted)	**1.95**	**0.66**	0.96	**3.47**	0.41	**1.49**	2.92	1.14	2.03	1.87	-0.47	0.70
	S&P500 equally	1.53	0.24	**1.18**	3.06	-0.24	1.15	2.37	0.63	1.50	**2.17**	**-0.35**	**0.91**
Sortino	GNN (Pearson)	2.09	0.84	1.24	4.00	-0.29	1.58	3.52	1.09	2.30	2.87	-0.82	1.02
	GNN (DCOR)	2.01	0.66	1.52	4.75	**0.59**	1.91	**4.15**	**1.25**	**2.70**	2.53	-0.89	0.82
	GNN (Quantile q=.1)	2.09	0.59	**1.55**	4.83	0.22	1.85	3.77	1.13	2.45	2.69	-0.79	0.95
	GNN (Quantile q=.2)	2.09	0.55	1.39	4.70	0.12	1.77	3.68	1.15	2.42	2.76	-0.66	1.05
	GNN (Handcrafted)	**2.58**	**0.97**	1.33	**5.19**	0.54	**2.12**	4.09	1.23	2.66	2.76	-0.78	0.99
	S&P500 equally	2.02	0.34	1.52	4.34	-0.30	1.58	3.01	0.73	1.87	**3.15**	**-0.57**	**1.29**
MDD	GNN (Pearson)	**-0.08**	**-0.14**	-0.19	-0.05	-0.40	**-0.17**	-0.09	**-0.54**	-0.31	-0.09	-0.26	**-0.17**
	GNN (DCOR)	-0.09	-0.16	-0.23	-0.07	**-0.36**	-0.18	**-0.07**	-0.61	-0.34	-0.10	-0.28	-0.19
	GNN (Quantile q=.1)	**-0.08**	**-0.14**	-0.20	-0.05	-0.41	-0.18	**-0.07**	-0.56	-0.31	-0.09	**-0.25**	**-0.17**
	GNN (Quantile q=.2)	-0.10	-0.16	-0.22	**-0.04**	-0.38	-0.18	**-0.07**	**-0.54**	**-0.30**	-0.10	**-0.25**	**-0.17**
	GNN (Handcrafted)	-0.09	-0.16	-0.22	-0.07	-0.44	-0.20	**-0.07**	-0.59	-0.33	-0.09	-0.26	**-0.17**
	S&P500 equally	-0.09	**-0.14**	**-0.18**	**-0.04**	-0.39	**-0.17**	**-0.07**	**-0.54**	-0.31	**-0.08**	-0.27	**-0.17**
Calmar	GNN (Pearson)	2.31	**0.62**	0.69	4.68	-0.09	1.64	3.95	0.64	2.29	2.95	-0.46	1.25
	GNN (DCOR)	2.09	0.42	0.75	4.20	**0.19**	1.53	**5.60**	0.68	3.14	2.28	-0.46	0.91
	GNN (Quantile q=.1)	2.39	0.44	0.82	5.25	0.06	1.79	5.19	0.67	2.93	2.75	-0.46	1.14
	GNN (Quantile q=.2)	1.98	0.38	0.73	6.05	0.04	**1.84**	4.93	**0.69**	2.81	2.59	-0.40	1.09
	GNN (Handcrafted)	**2.67**	0.61	0.63	4.60	0.15	1.73	5.41	0.68	**3.04**	2.68	-0.44	1.12
	S&P500 equally	2.01	0.26	**0.95**	**5.22**	-0.10	1.67	3.97	0.45	2.21	**3.58**	**-0.30**	**1.64**

Table 5: Performance Indicators.

As we can see from the metrics table, the *Handcrafted GNN* achieved the highest Sharpe and Sortino ratio during the training period, which is not surprising, but shows that a problem specific Q matrix improves the results. However, the GNN portfolios do not reduce the maximum drawdowns, which contradicts the hypothesis, that uncorrelated assets reduce the volatility of a portfolio.

Interestingly, during the first test period the portfolio based on the distance correlation outperformed the other portfolios. This may indicate that the distance correlation is a more appropriate correlation measure for portfolio optimization because it is better at capturing nonlinear correlations.

Conclusion & Outlook

Overall, the results are very promising. We were able to apply the approach of Schuetz et al. to a portfolio optimization problem with real data. The biggest achievement is, that we found a robust GNN model that finds good MIS on different graph structures, even if the graph has a large average node degree.

The main advantage of the Schuetz et al. approach over traditional solvers, namely the scalability, could not be tested, because graphs in this project were rather small. It would be interesting to apply my SAGE model to a larger asset universe and compare its runtime and the approximated MIS with a traditional solver.

Although the backtesting of the uncorrelated GNN portfolio showed good results, the main goal of reducing the portfolio's volatility was only partially achieved. It would be interesting to see if this holds true in a larger and more diversified asset universe.

Finally, a more sophisticated back test and asset allocation method should be applied. Since this was beyond the scope of this project, very basic methods have been used in these areas. Regarding the backtesting, a walk forward back test with periodic retraining of the GNN could be applied to get a more realistic results over a longer period of time.

Attachement

Run time	Best Loss	MIS Size	# Violations	dropout-rate	learning-rate	Model / Architecture
1’074.99	-38.98	39	0	0.05	0.0001	SAGE_2L_Model
1’151.59	-37.00	37	0	0.05	0.0010	SAGE_2L_Model
2’064.49	-36.00	36	0	0.05	0.0010	GAT_1L_2H_Model
1’159.50	-35.00	35	0	0.10	0.0010	SAGE_2L_Model
648.04	-33.00	33	0	0.10	0.0010	SAGE_1L_Model
1’975.92	-33.02	33	0	0.10	0.0001	GAT_1L_2H_Model
2’124.38	-33.00	33	0	0.05	0.0010	GAT_1L_4H_Model
2’286.62	-33.00	33	0	0.05	0.0001	GAT_1L_4H_Model
923.91	-32.22	32	0	0.05	0.0001	SAGE_1L_Model
1’844.99	-32.00	32	0	0.10	0.0010	GAT_1L_4H_Model
3’374.62	-30.00	30	0	0.10	0.0010	GAT_2L_2H_Model
811.02	-30.00	30	0	0.05	0.0010	SAGE_1L_Model
1’638.54	-30.00	30	0	0.10	0.0010	GAT_1L_1H_Model
2’247.61	-29.02	29	0	0.10	0.0001	GAT_1L_4H_Model
2’027.04	-27.64	28	0	0.10	0.0010	GAT_1L_2H_Model
1’170.61	-27.98	28	0	0.00	0.0001	GAT_1L_4H_Model
665.02	-26.00	26	0	0.10	0.0010	GCN_1L_Model
927.90	-25.63	26	0	0.10	0.0001	SAGE_1L_Model
681.80	-23.00	23	0	0.05	0.0001	GCN_1L_Model
65.16	-21.98	22	0	0.00	0.0001	SAGE_2L_Model
712.94	-21.60	22	0	0.10	0.0001	GCN_1L_Model
385.79	-21.99	22	0	0.00	0.0010	GAT_1L_2H_Model
2’015.45	-22.24	22	0	0.05	0.0001	GAT_1L_2H_Model
197.74	-20.98	21	0	0.00	0.0001	SAGE_1L_Model
754.22	-20.97	21	0	0.00	0.0001	GAT_1L_2H_Model
832.98	-19.00	19	0	0.10	0.0010	GCN_2L_Model
1’567.30	-18.75	19	0	0.10	0.0001	GAT_1L_1H_Model
817.96	-18.00	18	0	0.05	0.0010	GCN_2L_Model
659.02	-18.00	18	0	0.05	0.0010	GCN_1L_Model
849.52	-16.99	17	0	0.05	0.0001	GCN_2L_Model
788.98	-16.98	17	0	0.10	0.0001	GCN_2L_Model
4’759.51	-16.98	17	0	0.05	0.0001	GAT_2L_4H_Model
1’431.20	-17.01	17	0	0.05	0.0001	GAT_1L_1H_Model
3’367.03	-14.36	15	0	0.10	0.0001	GAT_2L_2H_Model
1’208.14	-15.18	15	0	0.05	0.0010	GAT_1L_1H_Model
2’625.36	-13.99	14	0	0.10	0.0010	GAT_2L_1H_Model
238.72	-13.92	14	0	0.00	0.0010	GAT_1L_1H_Model
2’605.04	-11.75	12	0	0.05	0.0001	GAT_2L_1H_Model
4’563.15	-12.00	12	0	0.10	0.0001	GAT_2L_4H_Model
2’343.58	-10.92	11	0	0.10	0.0001	GAT_2L_1H_Model
3’402.64	-11.01	11	0	0.05	0.0001	GAT_2L_2H_Model
3’706.61	-9.83	10	0	0.05	0.0010	GAT_2L_2H_Model

86.78	-7.96	8	0	0.00	0.0010	SAGE_1L_Model
46.96	-2.99	3	0	0.00	0.0010	SAGE_2L_Model
1’861.33	-1.00	3	0	0.05	0.0010	GAT_2L_1H_Model
56.58	0.00	0	0	0.00	0.0010	GCN_2L_Model
166.25	0.00	0	0	0.00	0.0001	GCN_2L_Model
18.03	0.00	0	0	0.10	0.0001	SAGE_2L_Model
176.96	0.00	0	0	0.00	0.0010	GAT_2L_1H_Model
500.55	0.00	0	0	0.00	0.0001	GAT_2L_1H_Model
134.57	0.00	0	0	0.00	0.0010	GAT_2L_2H_Model
725.27	0.00	0	0	0.00	0.0001	GAT_2L_2H_Model
351.60	0.00	0	0	0.00	0.0010	GAT_2L_4H_Model
2’204.79	0.00	0	0	0.05	0.0010	GAT_2L_4H_Model
2’186.76	-0.01	0	0	0.10	0.0010	GAT_2L_4H_Model
946.30	0.00	0	0	0.00	0.0001	GAT_2L_4H_Model
84.30	0.00	0	0	0.00	0.0010	GCN_1L_Model
168.93	-0.14	0	0	0.00	0.0001	GCN_1L_Model
301.90	-0.37	0	0	0.00	0.0001	GAT_1L_1H_Model
249.28	0.00	0	0	0.00	0.0010	GAT_1L_4H_Model