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

# 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 goal is to reduce the volatility of portfolio returns. At a high level, we consider a (potentially very large) universe of assets. Given a correlation matrix that measures the volatility among the asset universe, we construct a graph. With the assets as vertices and the edges indicating that the asset returns are correlated.

Finding the largest possible set of uncorrelated assets is a major challenge. The set of possible solution candidates grows exponentially with the number of nodes. This means a brute force approach is infeasible for sets with greater than 100. 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 network based approach claims to be very fast and accurate. The promising solution runtime of 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

The main objective of this project is to apply the approach of Schuetz et al. to the S&P500 assets to construct a portfolio of uncorrelated assets and compare its volatility and performance with benchmarks.

# Basic Theory

In this section, I cover the most relevant basic theories. First, I summarise the main points of the paper “Combinatorial Optimization with Physics-Inspired Graph Neural Networks” by Schuetz et al. 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”

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:

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

A picture containing text, diagram, line, origami

Description automatically generated

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:

Where is a vector of binary decision variables and the QUBO matrix is a square matrix encoding the actual problem. For undirected graphs the matrix can be assumed to be symmetric and in the upper diagonal form. To make things clear, let’s look at the simplest possible example. A graph with three nodes given by :

With a penalty term of 2, the matrix for the above graph is as follows:

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

If we select nodes 0 and 1, which violates the MIS, the loss is 0.

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

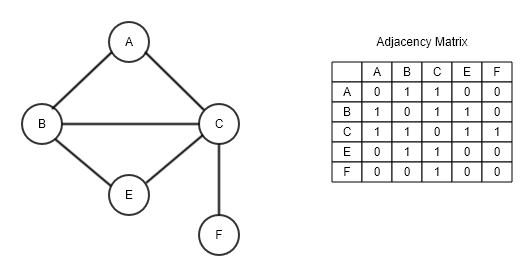
Figure 2 shows a simple graph with nodes A to F. These nodes are connected by undirected edges. Let’s stick with the example of people who know each other. Person A knows person B and C, person B and C know person A and others. These relationships can be converted into an **adjacency matrix**, where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

Figure 2: Simple graph with adjacency matrix.  
Source: [https://www.oreilly.com/](https://www.oreilly.com/library/view/hands-on-data-structures/9781788995573/b7b11326-d032-420c-a8c0-aad6181a8f2c.xhtml)

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 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.

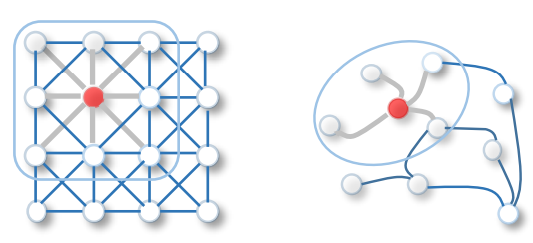
Figure 3 shows on the left-hand side an image convolution and a graph convolution on the right-hand side. In the case of images we can slide a learnable kernel over the grid structure of an image, which then extracts the most important information. This can also be seen as combining the information from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

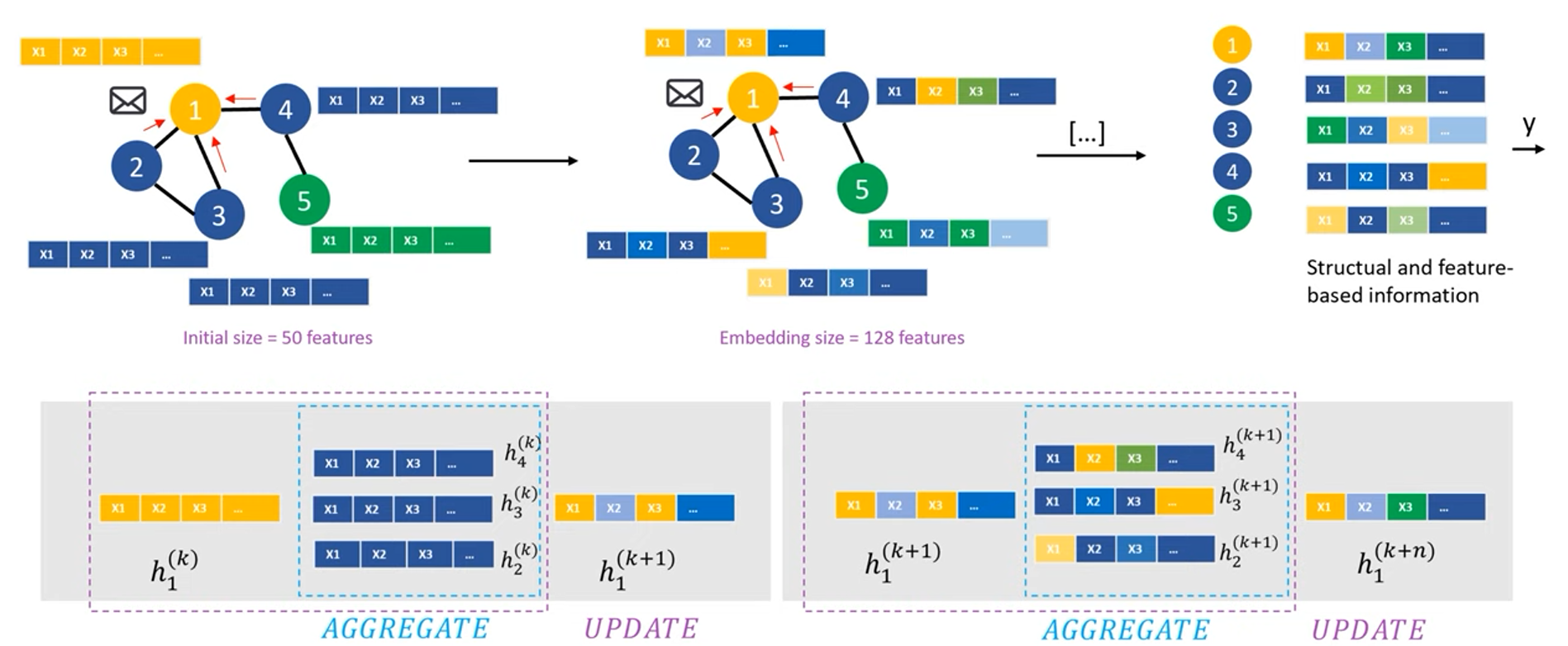


Figure 4: Illustration of message passing.  
Source: [DeepFinr, Understanding Graph Neural Networks | Part 2/3, Youtube](https://www.youtube.com/watch?v=ABCGCf8cJOE&list=PLV8yxwGOxvvoNkzPfCx2i8an--Tkt7O8Z&index=2)

# 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 maximal independent set of stocks among the all S&P500 stocks. The hypothesis is that the volatility of a portfolio of independent stocks will be 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, I will go through the portfolio optimization use case we have chosen for this project. 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](https://en.wikipedia.org/wiki/List_of_S%26P_500_companies).

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.

I used the daily adjusted closing prices from 2014 to 2023. 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, I decided to drop stocks from the dataset that did not have the full history of price data, which was about 5% of the stocks.

## Data-Processing

In this section, I highlight the transformation and computational steps 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 set):** I converted theactual prices 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 trainings 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.

A picture containing text, screenshot, plot, line

Description automatically generated

Testing period 2

Testing period 1

Training 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. (Gleeson, 2018, www.freecodecamp.org)
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. (Gleeson, 2018, www.freecodecamp.org)
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. (Ji-Eun & Dong Wan, 2022, Quantile correlation coefficient: a new tail dependence measure)

In particular,I have computed four different correlation matrices 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, I used the Python library *dcor* which provides a quite efficient method to compute the distance correlation between two arrays. The quantile correlation was the most difficult and least efficient. I used the *statsmodels.formula.api.quantreg* method to compute the quantile slopes and implemented the rest of the formula using *numpy*. However, the implementation does not scale to large datasets. It took about 2.5 hours to compute the quantile correlation matrix for our 470 stocks. To put this into relation, the Pearson’s correlation matrix was computed sub second.

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 : 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, I defined a threshold and assigned 1 to all correlations above or equal to the threshold and 0 to the rest. The result is an adjacency matrix where highly correlated stocks are connected. In general, the higher the threshold, the lower the degree of the graph and the larger the MIS. The optimal value for the threshold should be evaluated by back-testing. However, this is beyond the scope of this project, so I decided to use a **threshold of 0.35**.

**Step 4 (Generate the QUBO matrix and the graph):**

I used the *NetworkX* library to create graphs from the adjacency matrices and just removed the self-loops. Figure 5 shows the node degree distribution of the four different graphs.

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated with low confidence

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. Apart from the regularity, the average degree in my graphs is about 30-50 times higher.

Figure 6 shows the correlation graph. The low degree nodes are on the edge and the high degree nodes are 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, I have not plotted the fourth graph, which is based on the quantile correlation (q=.1).

A picture containing sketch, drawing, text

Description automatically generated

Figure 7: Correlation Graphs

Before the GNN can be trained, the problem must be encoded in a **QUBO matrix** . For the graphs shown above, I used the MIS problem encoding proposed by Schuetz et al. I used a reward value of -1 and a penalty value of 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. Uncorrelation may not be the only criterion for choosing an optimal portfolio. For investors, the returns of the portfolio are also important. Therefore, I modified the matrix and instead of assigning static rewards of -1 I used the negative average Sortino ratio of the training period as a reward.

The Sortino ratio is a well-known portfolio evaluation metric that measures the ratio of return to downside deviation. 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 and . If the graph is large () they suggest setting else and .

I tried to apply the architecture of Schuetz et al. with different hyperparameter sets to my 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 I worked with are quite different as they are not regular and have much higher average degrees.

I stared to search for better suited architectures and hyperparameter sets with a grid-search. I used Ray Tune, an industry standard tool for distributed hyperparameter tuning. I mainly experimented with different layer units, architectures, learning rates and dropout rates. In total, I tested 60 different combinations of architectures and hyperparameter sets. 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.

I have found that the SAGE unit significantly outperforms the GraphConv unit as well as the Graph-Attention unit. Furthermore, small dropout rates between 5-10% help to stabilize the training and increase the overall results. Learning rates between and showed consistently good results.

Table 3 shows the best performing model and the hyperparameter set:

|  |  |
| --- | --- |
| Model architecture | Hyperparameters |
| SAGE\_2L\_Model(  (conv1): SAGEConv(  (feat\_drop): Dropout(p=0.0, 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=0.0, 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': 50000,  'tolerance': 0.0001,  'patience': 1000,  'model': 'SAGE\_2L\_Model'} |

Table 3: Top performing model and hyperparameters.

# Evaluation

The evaluation consists of two parts. First, we evaluate the GNN approximated independent set. Second, I test the hypothesis that a portfolio of uncorrelated S&P500 stocks is less volatile than the S&P500 index.

## MIS Evaluation

Finally, I trained the above model on my input graphs for 25’000 epochs on my CPU laptop, which took about 230 seconds on average. For each input graph the GNN found a MIS with zero violations. To evaluate whether the approximated MIS are good, I compared them with the approximated MIS of 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 : 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 the GNN found a smaller MIS. It can happen that the GNN gets stock in a local minimum. To mitigate this problem Schuetz et al. suggest running the unsupervised learning several times and keeping track of the results. However, this was beyond scope of this project. Overall, Schuetz et al. approach worked well for this project.

## Portfolio Backtest

Finally, I tested how the uncorrelated portfolios performed 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 I have used only Benchmark 1 in the following figures. Figure 7 shows the cumulative returns for each portfolio over the training period:

A picture containing text, line, plot, diagram

Description automatically generated

Figure 8: Cumulative portfolio returns over the trainings period.

Interestingly, the uncorrelated portfolios outperform the benchmark in terms of returns, which is in a sense the opposite of the hypothesis. I 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, because I have encoded the (true) returns in the matrix. Thus, the GNN was able to select a portfolio that is not only uncorrelated but also contains the best stocks with in terms of their Sortino ratio.

Figure 8 shows the maximum drawdown of the portfolios during the training period:

A picture containing text, line, plot, font

Description automatically generated

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, I suspect that it is nearly impossible to reduce maximum drawdowns on an asset universe consisting only stocks in the S&P500 index.

Figure 9 shows the cumulative returns for each portfolio over the validation period:

A picture containing text, line, plot, diagram

Description automatically generated

Figure 10: Cumulative portfolio returns over the validation period.

Interestingly, the uncorrelated portfolios continue to outperform the benchmark in terms of returns over the validation 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 10. However, the portfolio based on the quantile correlation (q=.2) experienced slightly smaller drawdowns during this crisis.

A picture containing text, line, plot, diagram

Description automatically generated

Figure 11: Portfolio Drawdowns over the validation period.

These results do not support the hypothesis that uncorrelated portfolios are less volatile. However, this has to 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 I chose for this project is highly correlated by default and it may be very difficult to create a low volatility portfolio.

Finally, I chose a second test period (2022-2023) 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 11 shows the cumulative returns for each portfolio over the second test period:

A picture containing text, line, plot, screenshot

Description automatically generated

Figure 12: Cumulative portfolio returns over the 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 I had retrained the model with data up to the start of the second test period.

Figure 12 shows the maximum drawdown of the portfolios during the second test period:

A graph with red and blue lines

Description automatically generated with low confidence

Figure 13: Portfolio Drawdowns over the 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, I calculated the following four performance metrics for each year:

* The **Sharpe ratio** compares the return of an investment with its risk. 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, a measure of volatility and risk.
* 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—downside deviation—instead of the total standard deviation of portfolio returns.
* **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.
* **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.

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



Table 5: Performance Indicators.

# Conclusion & Outlook

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

To see the full potential of Schuetz et al. approach the asset universe should be larger. It would be interesting to see

# 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 |