Math Meets Money

The intersection of combinatorics and finance for portfolio optimization and risk management

Blake Marterella

Monday 22nd April, 2024

Abstract

In this study we will explore the intersection of combinatorics and finance, specifically how graph theory can be used to optimize and assess risk in a stock portfolio. Using various theorem's and definitions in graph theory, we will analyze the composition of a portfolio to determine low risk, medium risk, and high risk holdings along with how the correlation between various stocks. Appliying mathematics to finance allows individuals to make more informed trades and mitigate risks by gaining insight to the mathematical signifigance of a stock price on any given day. A sample portfolio is introduced in this study, along with 4 years of historical stock data, but the concepts explored extend beyond this sample. The goal of this paper is to provide a theoretical framework for understanding portfolio optimization and risk assessment using advanced mathematical tools that can be applicable to any portfolio.

Contents

1 Introduction

1.1 Brief History

Mathematics has always been a powerful tool for humans to discover and describe seemingly complex patterns in the natural world. From the ancient days of arithemetic used to facilitate trade to the complex algorithms governing modern financial markets, mathematics has been an indispensable tool. The pivotal moment in the incorporation of mathematics into finance occured in 1654 when two great mathematicians, Blaise Pascal and Pierre de Fermat, developed probability theory to help predict the outcome of gambling [KD2010]. The ideas introduced in 1654 would evolve and help create the first automated trading system in 1949. Richard Donchian founded a commodity fund that used rule-based trading to execute trades based on moving averages in the market [LS2023]. These early pioneers set the stage for a methematical framework that would evolve into critical financial theories such as portfolio opimization, diversification, and risk management. In 2023, over 50% of all trades in the US stock market were executed by algorithms [LS2023]. A critical tool in financal analysis is graph theory which can be used to represent the relationships between various stocks in a portfolio and help investors make more informed decisions.

1.2 Interest

The integration of combinatorics and finance, especially by the use of graph theory, is interesting for two main reasons. First, it provides a more comprehensive view to the correlations and diversification within a financial portfolio. Graph theory translates complex market dynamics into comprehensible, manageable models, allowing investors to visualize portfolio holdings in a new perspective and make informed decisions. Secondly, with the strategic application of graph theory, any individual's holdings can be optimized in such a way that investments remain not only sound but also resilient against market volatities. With the proper mathematical tools, any investor can have the power to make informed decisions just as large investment firms do.

1.3 Motivation

This study aims to democratize the complex mathematical strategies that find their place in financial markets so that more accessible knowledge and strategies are found for a more open audience. Though such advanced models have long been in use by large investment firms or hedge funds, it certainly would be of great value to put this kind of information in the hands of small investors. Through the examination of foundational concepts of graph theory and finance, this paper seeks to prepare the reader with the tools of understanding and engaging actively with their investment strategies.

The field of quantitative graph theory is rapidly expanding and encapsulating fields such as machine learning, graph algorithms, and quantitative analysis [MSFEYS2017]. This study serves as foundational and prerequisite knowledge to apply more complex analysis on financial data. By exploring the basics and providing access to historical stock data, individuals can become powerful and successful independent traders.

2 Background

2.1 Data Collection

For the purpose of this paper, I took the time to develop a custom API that allows me to quickly export historical data for a given stock [BM2024]. The API contains an endpoint that allows users to generate a CSV file for any given stock ticker and date range, providing 20 years of historical data. The csv file is loaded into a Pandas dataframe with the following fiels:

- **Date - The date of the stock price.**
- **Open - The opening price of the stock on that date.**
- **High - The highest price the stock reached on that date.**
- **Low - The lowest price the stock reached on that date.**
- **Close - The closing price of the stock on that date.**
- **Volume - The number of shares traded on that date.**

To enforce the concepts introduced in the paper, we will simulate a sample stock portfolio that contains 30 stocks from the DOW 30[1]. The API endpoint is used to pull a CSV file, hereby referred to as a dataset, for each

[1]The Dow Jones Industrial Average was selected because it is a relaitvely small list yet it is one of the most popular metrics to benchmark the stock exchange[DC2018]

stock in the portfolio that contains data from the past 4 years[2], along with the fields mentioned above.

2.2 Data Processing

Before continuing, it is important to understand how a graph can be constructed to represent using the datasets. Each stock within our portfolio will be represented as a vertex and the relationship between each stock will be represented as an edge. This edges can be drawn using many different relationships depending on what your graph is attempting to convey, such as: edges connecting companies in the same sector, edges connecting companies with similar market capitalization, etc. For the purpose of this study, each edge will be the correlation factor between stocks.

Definition 2.1 (Pearson Correlation Coefficent)

$$r = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}}$$

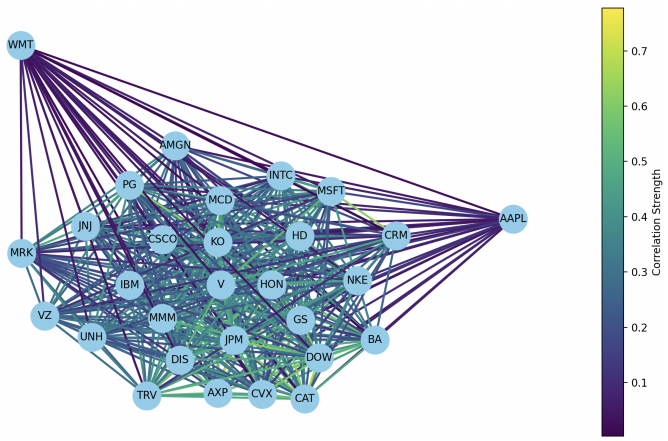
The Pearson Corelation Coefficient, denoted as r, is a measure of the linear relationship between two variables. In the context of this study, r is the correlation between 2 stocks' closing price over a 4 year period. X_i and Y_i is the closing price of stock x and stock y on day i. $\overline{X}$ and $\overline{Y}$ is the average closing price of stock x and stock y over the 4 year period. In the formula above, the numerator calculates the covariance between 2 stocks and the deonominator uses product of stock x and stock y's standard deviation. The result is a number between -1 and 1, where -1 is a perfect negative linear relationship (as X increases, Y decreases), 1 is a perfect positive linear relationship (as X increases, Y increases), and 0 represents no linear relationship (there is no way X can predict Y)[ST2024].

Using the datasets in our portfolio, r is calculated for each pair of stocks. The result is a correlation matrix. The table below shows a subsection of our correlation matrix using 4 stocks. Note that, the correlation for a stock with itself is always 1.0.

	AXP	**AMGN**	**DIS**	**DOW**
AXP	1	0.160404	0.553888	0.586814
AMGN	0.160404	1	0.141811	0.192764
DIS	0.553888	0.141811	1	0.463608
DOW	0.586814	0.192764	0.463608	1

Figure 1 depicts a graph constructed with the correlation matrix that allows us to visualize the relationships between stocks in the portfolio using a color scale. Although the information presented is not usable, it provides a foundation for the application of graph theory in portfolio management.

Figure 1: Correlation Matrix Graph showing the relationships between 30 stocks in our sample portfolio



The figure has 30 vertices and 435 total edges. The average number of degrees for each vertex is 29, meaning that the average stock is correlated with 29 other stocks in the portfolio. The maximum number of edges possible

[2]The number, 4 years of DOW 30 historical data, was selected because it accounts for various financial markets

for an undirected graph without loops and 30 verties is $\binom{30}{2} = \frac{30(30-1)}{2} = 435$ edges. Therefore the graph is dense because the number of edges is equal to the maximum number of edges possible. This is a sign that the portfolio is not well diversified and that there are many stocks that are highly correlated with each other. Additionally, the average correlation strength is 0.303. A diversified portfolio should have a low average correlation strength and a low number of edges. In the next section, we will explore how graph theory can be used to optimize and diversify a portfolio.

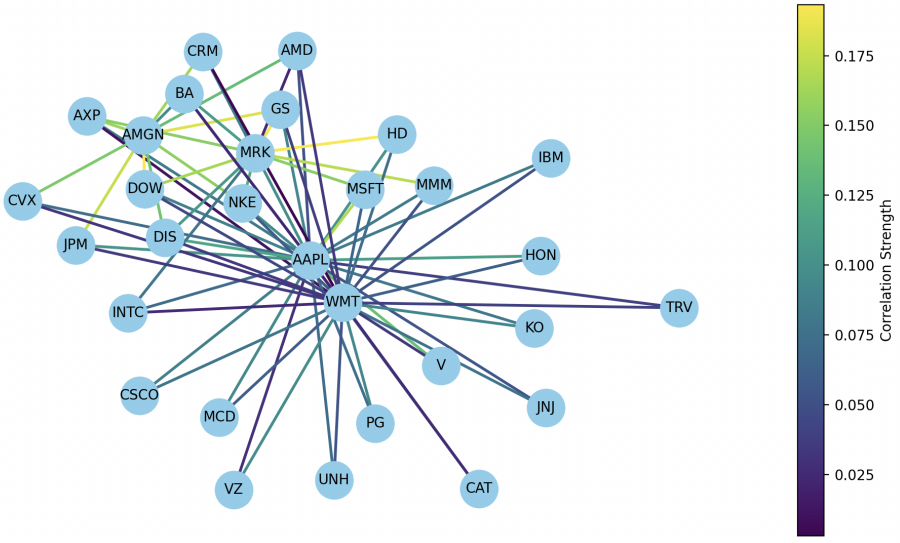
3 Portfolio Optimization

Definition 3.1 (Extremal Graph Theorem) *Let G be a graph with n vertices and m edges. Then, if G does not contain a subgraph isomorphic to K_{r+1}, the complete graph on $r+1$ vertices, then $m \leq \frac{r}{2}(n-1)$.*

Extremal Graph Thoerem is a powerful tool in portfolio optimiziation. The theorem can be particularly useful in preventing over-concentration of correlated assets, thus enhancing the robustness of the portfolio against market volatility. The theorem states that if a graph does not contain a subgraph isomorphic to K_{r+1}, then the number of edges in the graph is bounded by $\frac{r}{2}(n-1)$. In the context of a stock portfolio, avoiding a subgraph K_{r+1} means ensuring that no group of $r+1$ stocks are all heavily correlated with each other. This helps in spreading out the investment risk by limiting the number of direct correlations any single stock has within the portfolio, thus achieving a higher degree of diversification. For example, a conservative portfolio may want to exclude any subset of 3 stocks from being mutually interconnected. In this case, $r = 3$ and $n = 30$ (since the portfolio contains 30 stocks). By applying the theorem, we can calculate the maximum number of edges allowed in the graph to maintain a diversified portfolio.

For further optimization, I only allowed an edge to be drawn in the graph if the correlation between two stocks was less than .2. This number was selected as it is less than the average correlations strength of the original graph show in Figure 1. Figure 2 shows a graph that does not contain cliques of size 3 or more and has a correlation threshold of .5. The resulting graph, shown in Figure 2, still has 30 nodes but the number of edges has been reduced to only 74. The average correlation strength for the graph is 0.08.

Figure 2: Extremal Graph Theory applied to the correlation matrix graph



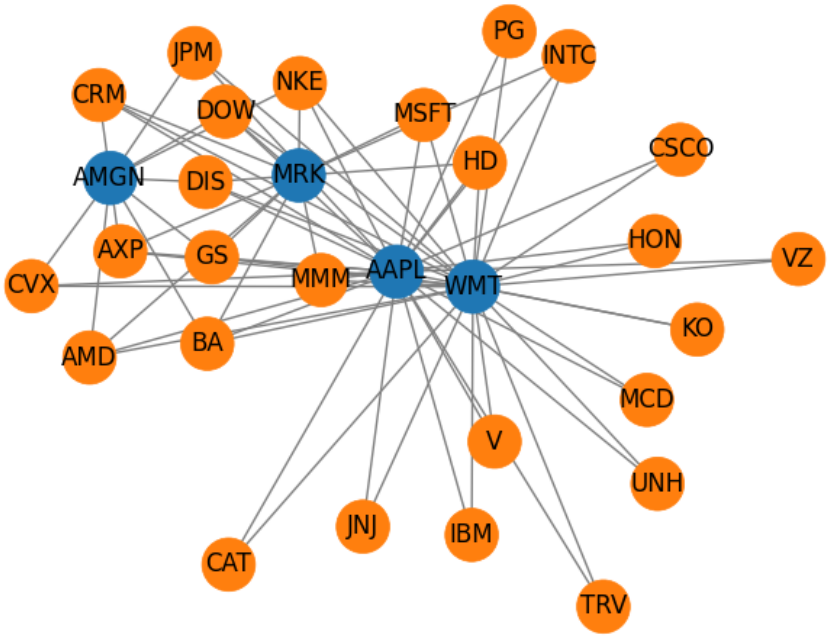
4 Risk Management

Now that we have optimized our portfolio, we can use graph coloring to represent different levels of risks and create groupings of stocks that are less correlated with each other. By partinitioning the graph using a coloring algorithm, we can determine the most optimial holding strategy for our portfolio.

Definition 4.1 (Welsh-Powell Coloring Algorithm) *A graph coloring algorithm that assigns colors to vertices, such that no two adjacent vertices share the same color. The algorithm starts with the vertex with the highest degree and assigns the first color. It then finds all adjacent vertices and assigns a different color. The algorithm continues to color the remaining vertices in descending order of degree until all have been colored.*

In the context of the optimized graph (constructed using extremal graph theory in the section above), the vertex with the highest degree is "AAPL" which has 26 total edges. AAPL is colored blue and the 26 adjacent vertices are colored orange. From here, there are only 3 remaining vertices which all do not happen to be neighbors. Therefore, our graph only needs 2 colors to represent the entire portfolio. The graph is shown in Figure 3.

Figure 3: Welsh-Powell coloring algorithm applied to the optimized graph



After coloring the graph, we can see that the portfolio is divided into two groups: blue and orange. Each of these groups represents a diversified portfolio that contains stocks less correlated with each other, thereby reducing the overall risk of the portfolio. The blue group contains 26 stocks and the orange group contains 4 stocks. The orange group is less risky than the orange group because it contains more stocks and is more diversified. The blue group is more risky because it contains fewer stocks and is less diversified. By using graph coloring, we can easily visualize the risk of our portfolio and make informed decisions on how to adjust our holdings.

5 Conclusion

Using the orange group of stocks and the unmodified sample portfolio, we can quantify the effectiveness of the graph theory concepts explored in this paper. To test, say you invested $100 into each stock in the original portfolio on April 24, 2020. After 4 years, there would be a 32.87% return on investment. However, using the filtered orange group of stocks with the same investment stragey, the return on investment would be 67.91%. This is over a 100% increase in return on investment by using the graph theory concepts explored in this paper. In summary, the concepts of graph theory can be used to optimize and diversify a stock portfolio, thereby reducing risk and increasing return on investment.

5.1 Future Development

Althought the Welsh-Powell coloring algorithm is a powerful tool for risk management, it is not the only tool available. There are many other graph coloring algorithms that can be used to partition a graph into different groups, some do it more optimally than others. One example is a backtracking algorithm which will correct coloring

if a conflict arises, thereby helping ensure that the graph is colored with the minimum number of colors [AB2023]. This comes at the cost of complexity.

For further optimization, there are more complex graph theory concepts can be applied to not only manage a portfolio but execute trades at the perfect time.

References

[KD2010] K. Devlin, *The Pascal-Fermat Correspondance*, Mathematics Teacher, Vol. 103 **No. 8**, National Council of Teachers of Mathematics (2010), 580-582.

[LS2023] L. Smigel, *Algorithmic Trading History: A Breif Summary*, Analyzing Alpha (2023).

[MSFEYS2017] M. Dehmer, S. Emmert-Streib, Y. Shi, *Quantitative Graph Theory*, Information Sciences, Vol. 418 **Issue C**, Association for Computing Machinery Digital Library (2017).

[ST2024] S. Turney, *Pearson Correlation Coefficent*, Scribbr (2024).

[DC2018] D. Caplinger, *How Often Do Dow Stocks Get Replaced*, The Motley Fool (2018).

[BM2024] B. Marterella *Stock Market API*, marterella.com (2024).

[AB2023] A. Bhuyan *Understanding Graph Coloring*, Medium, FAUN - Developer Community (2023).

Appendices

A Code

Note that the code below was modified from its original form to only show the relavant information. For example, function docstrings, input validation, etc. have been removed for brevity.

A.1 Import Historical Stock Data

```
def get_stock_data(ticker, start=None, end=None):
    url = f"{BASE_URL}historical-stock-data"
    params = {
        "symbol": ticker,
        "start": start,
        "end": end,
        "datatype": "json"
    }
    response = requests.get(url, params=params)

    if response.status_code == 200:
        data = response.json()
        json_data = json.loads(data['data'])

        # Load the JSON data into a Pandas DataFrame
        df = pd.DataFrame(json_data)
        df['date'] = pd.to_datetime(df['date'])

        # Ensure the directory exists and save CSV
        os.makedirs(os.path.dirname(path), exist_ok=True)
        df.to_csv(path, index=False)

        return df
    else:
        print("Failed to fetch data. Status code:", response.status_code)
        return None
```

A.2 Import Historical Stock Data for Portfolio

```
def get_portfolio_data(tickers):
    portfolio_data = {}

    for ticker in tickers:
        start = datetime(year=2020, month=4, day=25).strftime('%Y-%m-%d')
        end = datetime(year=2024, month=4, day=24).strftime('%Y-%m-%d')
        ticker_df = get_stock_data(ticker, start=start, end=end)

        if ticker_df.empty:
            print(f"Failed to fetch data for {ticker}!")
            break
        else:
            portfolio_data[ticker] = ticker_df

    # Return dictionary of stock dat
    return portfolio_data
```

A.3 Get Pearson Correlation Coefficient

```
def get_pearson_correlation_matrix(portfolio_data):
    returns = {name: df['close'].pct_change() for name, df in portfolio_data.items()}
    returns_df = pd.DataFrame(returns)

    return returns_df.corr(method='pearson')
```

A.4 Display Graph

This helper function makes displaying a graph using color mapped edges easier.

```
def draw_graph(G):
    pos = nx.spring_layout(G, k=0.2, scale=1)  # positions for all nodes

    # Generate edge colors based on weight
    weights = [G[u][v]['weight'] for u,v in G.edges()]
    edges = G.edges()
    edge_colors = plt.cm.viridis((np.array(weights) - min(weights)) /
                                 (max(weights) - min(weights)))
    fig, ax = plt.subplots()

    # Drawing Nodes, Edges, and Labels
    nx.draw_networkx_nodes(G, pos, node_color='skyblue', node_size=700, ax=ax)
    nx.draw_networkx_edges(G, pos, edgelist=edges, edge_color=edge_colors, ax=ax)
    nx.draw_networkx_labels(G, pos, font_size=10, font_family='sans-serif', ax=ax)

    # Color bar settings
    sm = plt.cm.ScalarMappable(cmap=plt.cm.viridis,
            norm=plt.Normalize(vmin=min(weights), vmax=max(weights)))
    sm.set_array([])
    cbar = plt.colorbar(sm, ax=ax, orientation='vertical')
    cbar.set_label('Correlation Strength')

    plt.axis('off')
    plt.show()
```

A.5 Draw Correlation Matrix Graph

```
    def draw_correlation_matrix_graph(portfolio_data, corr_threshold=-1):
    correlation_matrix = get_pearson_correlation_matrix(portfolio_data)

    G = nx.Graph()
    for stock1 in correlation_matrix.columns:
        for stock2 in correlation_matrix.index:
            if stock1 != stock2 and correlation_matrix.loc[stock1, stock2] > corr_threshold:
                G.add_edge(stock1, stock2, weight=correlation_matrix.loc[stock1, stock2])

    draw_graph(G, image_name)
```

A.6 Apply Extremal Graph Theory

```
def apply_extremal_graph_theory(portfolio_data, corr_threshold=-1, max_clique_size=3):
    correlation_matrix = get_pearson_correlation_matrix(portfolio_data)
    G = nx.Graph()

    # Add edges based on the correlation threshold and avoiding cliques
    for stock1 in correlation_matrix.columns:
        for stock2 in correlation_matrix.index:
            if stock1 != stock2 and correlation_matrix.loc[stock1, stock2] < corr_threshold:
                # Tentatively add edge
                G.add_edge(stock1, stock2, weight=correlation_matrix.loc[stock1, stock2])

                # Check for cliques and remove the edge if it forms a forbidden clique
                if any(len(clique) >= max_clique_size for clique in nx.find_cliques(G)):
                    G.remove_edge(stock1, stock2)

    draw_graph(G, image_name)
```

A.7 Apply Welsh-Powell Coloring Algorithm

```
def visualize_welsh_powell_coloring(portfolio_data, threshold=0.5):
    G = apply_extremal_graph_theory(portfolio_data)

    # Sort nodes by descending degree
    sorted_nodes = sorted(G.nodes(), key=lambda x: G.degree(x), reverse=True)

    custom_colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b']

    # Apply Welsh-Powell coloring algorithm
    color_map = {}
    available_colors = set(range(len(custom_colors)))  # Use index of custom colors
    for node in sorted_nodes:
        adjacent_colors = {color_map.get(neighbor) for neighbor in G.neighbors(node)}
        color_map[node] = min(available_colors - adjacent_colors)

    # Map colors from indices to custom colors
    node_colors = [custom_colors[color_map[node]] for node in G.nodes()]

    # Prepare the graph layout
    pos = nx.spring_layout(G)

    # Draw the graph
    nx.draw(G, pos, node_color=node_colors, with_labels=True, node_size=700, edge_color='gray
    plt.axis('off')
    plt.show()
```