ANALYSIS OF THE STOCK MARKET USING COMPLEX NETWORKS AND MACHINE LEARNING

A DISSERTATION SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF MASTER OF SCIENCE
IN THE FACULTY OF SCIENCE AND ENGINEERING

2018

By
Lingjie Zhang
School of Computer Science

Contents

Word Count: 1626

List of Tables

List of Figures

Abstract

ANALYSIS OF THE STOCK MARKET
USING COMPLEX NETWORKS
AND MACHINE LEARNING
Lingjie Zhang
A dissertation submitted to the University of Manchester
for the degree of Master of Science, 2018

Due to the complexity of financial market and the interconnectedness and interdependencies of industry sectors in the economy, the price returns of each coupling stocks might have certain underlying economic link. Such behaviours can hardly be explained by traditional financial models and theories. This project combines machine learning techniques, individual stock features, and empirical data of Industry Economic Accounts (IEAs) from Bureau of Economic Analysis (BEA) in the US to predict Granger causality of coupling US stocks. Limited Granger causalities are calculated as a small sample set compared to the target date set. A directed weighted complex network (DWCN) is constructed by considering companies as nodes, correlations of abnormal stock returns (alpha) as weights of links, and predicted Granger causalities indicate directions of links. The generated DWCN is visualised and its topological properties, stability, and effects on individual stocks and industries are researched in this paper. Suggestions towards financial market investment are provided based on the results of this research.

Declaration

No portion of the work referred to in this dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

I would like to thank...

Contents

Chapter 1

Introduction

1.1 Motivation

Financial markets are complex systems, the interconnectedness and interdependencies of industry sectors in the economy are highly inter-coupled with strong correlations with stock price fluctuations, i.e., the price returns of each coupling stocks underlying certain economic link, e.g. two companies that manufacture similar products, or both in one supply chain. Such behaviours can hardly be explained by traditional financial models and theories.

During recent times, weighted but undirected complex network models have been applied to study the correlations of stock prices. Prevailing approach is to use companies as nodes, and correlations between each pair of stock price time series, return time series, or fluctuation patterns as links. As a result, much of the previous researches have proved the represented complex networks of worldwide stock markets are scale-free and small-world [LLH07, CLL10]. However in theory, directed complex networks for the stock market can be achieved hence more potential information can be produced which is helpful for investment decisions and financial market supervisions.

For building an investment portfolio of stocks, one should consider not only about the irrelevancy on price or price return time series, but also about the irrelevancy on economical activities between firms and industries. Naturally and instinctively, we can depict the interactions of economical activities between stock-pairs as the directions of edges in the stock complex network, and the weights of the edges are still determined by the correlations of stock price return time series as many previous researches practised. Therefore, some guidance may be offered for investors to build more effective portfolio and market supervisors to avoid potential crisis according to the study of directed stock complex network.

The goal of this thesis is to reveal the interactions of economical activities between companies and utilise them into the topological analysis and visualisation of constructed directed complex networks as so far no previous work has attempted to construct a directed network about stock markets. In addition, suggestions for stock market are provided according to the results and findings.

In addition to this approach, there was a goal of using machine learning techniques to predict the directions of stock complex network. However, these results were not as successful and conclusive as we expected. They are reported in the last chapter of this thesis as additional work complementing the core analysis of this thesis based on complex network theory.

1.2 Objectives and deliverables

The goal of this project is to construct a directed complex network using economical industrial transaction data and stock price data to depict the US stock market by means of topological properties analysis, community detection and visualisation. Same-sized directed Watt-Strogats small-world network and random networks are generated for the purpose of comparison. This paper will explore whether the conclusions are consistent with the undirected complex network researches.

Objectives produced:

- To normalise the Economic Input-Output (EIO) tables into matrices for setting thresholds.
- To generate a matrix of correlation coefficients for all stock-pairs.
- To set appropriate thresholds of correlation coefficient and normalised economical transactions.
- To construct directed-unweighted and directed-weighted stock price return networks.
- To study the topological properties of directed stock price return networks.
- To detect communities in the directed stock price return networks.

- To visualise the directed stock price return networks.
- To study the relationships between price return and betweenness centrality.

Deliverables produced:

- Stock counts by industries.
- Matrix of EIO transaction flows.
- Heatmap of combinations of thresholds of directed demands and directed requirements flows and correlation coefficients.
- Two benchmarking networks: directed Watts-Strogatz (WS) small-world network and directed Erdős–Rényi (ER) random network.
- Directed-unweighted and directed-weighted stock network.
- Topological properties of studied networks.
- Community partition of directed-unweighted stock network.
- We plan to produce a research paper to submit to a journal. We are thinking of journals like: *Physica A, Journal of Mathematical Finance, Journal of Applied Mathematical Finance, Applied Network Science*.

1.3 Proposed methodology

In this thesis we proposed a method to generate directed-unweighted complex network and directed-weighted complex network for stock market, especially for the US stock market in 2016. Analysis of topological properties and comparison with benchmarking networks provide unique insights towards its continuity or uncontinuity features to conventional undirected stock networks in previous researches and its compositional structure.

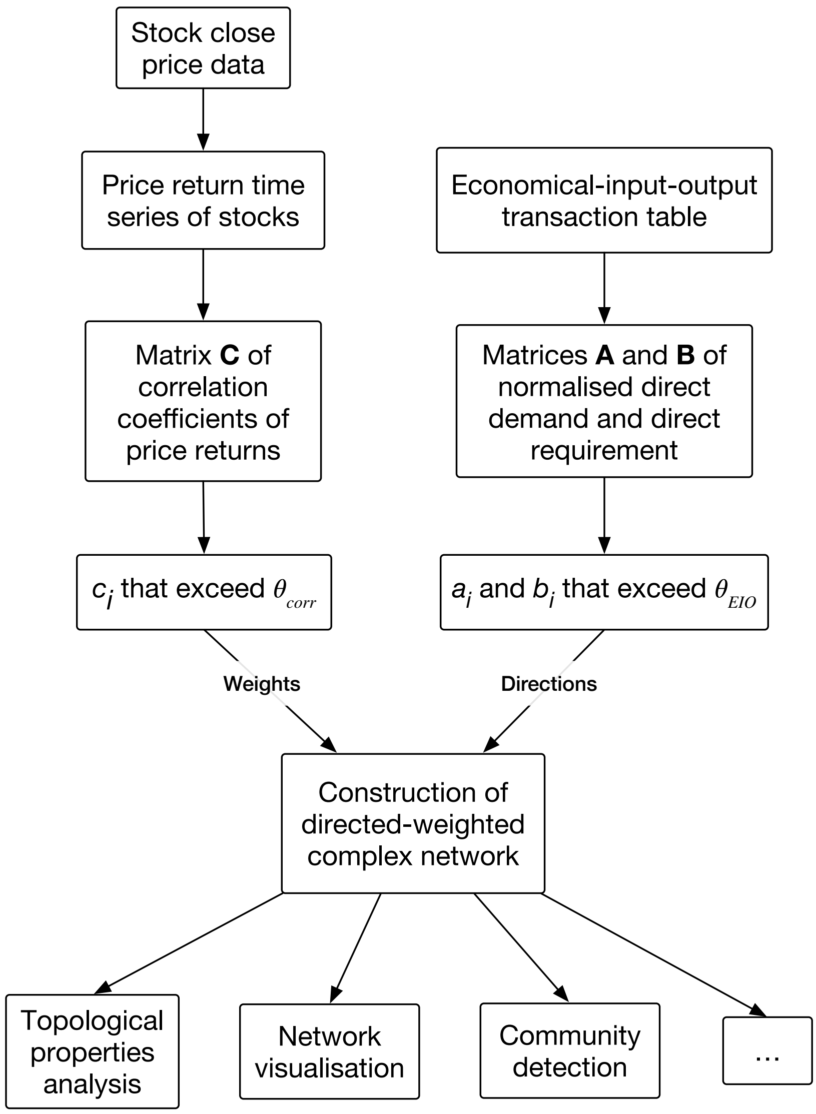


Figure 1.1: The methodology

1.4 Summary of results

While working on this thesis, the following achievements have been made that will be described in detail in this thesis:

- To calculate topological property values to study the the properties of stock complex networks.
- To compare the directed-unweighted stock complex network with generated directed WS small-world network and directed ER random network.
- To apply community detection algorithm to find possible communities of the directed-unweighted stock complex network.
- To generate bivariate distributions between betweenness centralities and functions of stock daily return to find the potential relationships between them.

1.5 Outline of the thesis

The rest of this thesis is organised as follows. Chapter 2 discusses the development of quantified financial analysis for stock markets and the application of complex theories of complex networks towards stock markets. The subsequent chapters of methodologies introduce the critical analytical methods implemented in the research by this paper. Detailed outcomes are then illustrated in Chapter 7. Finally, the findings and conclusions are discussed in Chapter 9.

Chapter 2

Background and state of the art

2.1 Introduction

This chapter will introduce the development of quantitative finance, stock market investment, and the application of complex networks theory into the financial markets, especially the stock markets.

2.2 Background

2.2.1 Historical review of quantitative finance and stock market investment

Quantitative finance combines mathematics, statistics, econometrics, machine learning and empirical finance to provide a solid analytical foundation for the analysis of financial issues, especially for the investments in financial markets such as stock, future, option and forex. The theoretical foundations of quantitative finance including efficient market theory, behavioural finance, asset allocation theory and Chaos theory.

The revolutionary who pioneered the theory of asset allocation, Markowitz (1952) proposed the idea of efficient frontier and the mean-variance model, which established a clear mathematical definition of the two previously vague concepts of risk and return [Mar52]. Markowitz (1956, 1959) then presents the properties and formulas for efficient frontier [Mar56] and generalised it by allowing an arbitrary, possibly singular, covariance matrix [Mar59]. Tobin (1958) developed Markowitz's theory and added the concept of risk free assets [Tob58]. Sharp (1964) and Linter (1965) added two key assumptions based on the mean-variance model to enable the portfolio mean-variance valid, forming a capital asset pricing model (CAPM) with the support of economic theories [Sha64, Lin65]. The CAPM believes that only non-dispersible systemic risks can be compensated off, while non-systematic risks can be eliminated by effectively decentralised investments. Investors could only assume systemic risks through decentralised investment. The systematic risk of a single security or portfolio can be characterised by beta, which represents the extent to which a single security or portfolio is affected by the overall market volatility. And later, Richard Michaud and Robert Michaud (1998) addressed the information uncertainty in risk-return estimates by the invention of the Re-sampled Efficient portfolio [MM98].

Since the invention of the CAPM, worldwide scholars had been actively conducting empirical tests towards the practicality of the CAPM, while early results show that *beta* is able to explain the return movements of stock prices. However, in the late 1970s, some empirical studies on the CAPM began to show that a large part of the changes in the stock returns cannot be explained by *beta*, as anomalies about market return were increasingly found.

Researchers had proposed models and theories considering the individual stock features to explain such anomalies. For instance, Fama and French (1993, 1996) proposed a three-factor model based on the inter-temporal capital asset model (ICAPM) [Mer73] proposed by Merton (1973) and the arbitrage pricing theory (APT) [CR76] proposed by Cox et al.(1976), which reveals a large part of the cross-section of the average return of stocks that cannot be explained by CAPM, can be explained using firm size, book-to-market equity ratio, and overall market return [FF93, FF96] as regression factors.

2.2.2 Theory of complex networks

Complex networks are the networks that have some or all of the properties from self-organisation, self-similarity, attractors, small-world and scale-free. A large number of complex systems that exist in nature or society can be described by a variety of networks. Since complex networks are the topological basis for the existence of a large number of complex systems, the research on complex networks is believed to help understand the critical problem of ”why complex systems are complex”.

A typical network consists of a number of nodes and edges which connect these nodes, where nodes are used to represent different individuals in the real system and edges are used to represent the relationships between individuals. In certain cases, a particular relationship is described as an edge between two nodes, otherwise, it is not connected. Two nodes with edge are considered as adjacent among the network.

For instance, the nervous system can be regarded as a network formed by a large number of nerve cells interconnected by nerve fibres [WS98]. The computer networks can be regarded as a network in which autonomously working computers are connected to each other through communication media such as optical cables, twisted pairs, coaxial cables, etc [WS98]. In addition, there are more complex networks like electric power networks [FFF99], social networks [WS98, HSW17, EMB02], etc.

The research of complex networks involves the knowledge and theoretical basis of many disciplines due to its cross-disciplinary and complex characteristics, especially for system science, statistical physics, mathematics, computer and information science. Commonly used analytical methods and tools for complex networks research include graph theory, combinatorial mathematics, matrix theory, probability theory, stochastic process, optimisation theory and genetic algorithms. The main research methods for complex networks are based on the theories and methods of graph theory. However, in recent years, many concepts and methods of statistical physics have been successfully applied in the modelling and calculating of complex networks, such as statistical mechanics, self-organisation theory, critical and phase transition theory, seepage theory, etc. [AB02], such as the concept of network structure entropy, and its application of quantitative measure of the ”order” of complex networks. The models of complex networks are widely used in numerous scientific areas.

Below are some frequently used topological properties and statistical features of complex networks:

- Average path length
- Clustering coefficient
- Degree (strength) and degree (strength) distribution
- Centrality
- Small-world
- Scale-free

Below are some common complex networks models:

- Regular network
- Random network
- Small-world network
- Scale-free network
- Self-similar network

The details of major topology and community features of complex networks will be presented in chapter 4 and the introduction of small-world and random networks will be presented in chapter 5.

2.2.3 Complex networks theory applied in Finance

While traditional stock pricing models still capture limited forms of financial behaviour, according to Johnson et al. (2003), the premises of standard financial theory contradict the modern notion of financial markets are complex systems [JJH$^+$03], by which many statistical niceties such as stationarity no longer can be taken for granted.

Since the property of small-world [WS98] and scale-free [BA99] are respectively revealed by the research upon complex networks by Watts (1998) and Barabási (1999), the application of complex networks has been greatly promoted to each field including finance. Therefore, recent researches have implemented the complex networks theory to reveal the underlying factors of price movements in financial markets. Huang et al. (2009) implemented the threshold method to construct correlation network in China's A-Share stock market and studied the topological properties and topological stability of the stock correlation networks [HZY09]. Their statistical analysis of the degree distribution has revealed the power-law property of financial networks, and the networks display a topological robustness against random node failures, while they are also fragile under intentional attacks. Namaki et al. (2011) utilised Random Matrix Theory (RMT) to specify the biggest eigenvector in the complex networks of price correlations [NSRJ11], which reveals that the Tehran Stock Exchange correlation network is scale-free in a specific time interval. Yu (2013) studied the evolution of gold price from a network perspective using the visibility network approach [Lon13] and shows that the series of gold price and gold price return are long-term correlated, fractal series with a power-law degree distribution of visibility graph network. Chopra and Khanna (2015) developed a framework which associates the economic input–output model with techniques for understanding interdependencies and interconnectedness in the economy of US, based on complex networks theory [CK15]. Its topological analysis for two networks suggests that the unweighted network exhibits small world properties, and the weighted network follows a power-law with an exponential cut-off. Boginski et al. (2005) identified cliques and independent groups among stock networks [BBP05], which invents a new alternative data mining method to the classification of stocks. Chen et al. (2015) studied the inter-stock and inter-industry effects towards stock returns based on APT and the topological properties of a complex network of correlations [CLSW15]. They have found that the average centrality of the top 100 stocks tracks the GDP growth rate of China, hence, the degrees of connection between stocks in the stock market reflect the development of the real economy to some extent. Another finding is that stocks with smaller market capitalisation tend to be located in more central positions in networks. Brida (2002) proposed the approach by using symbolic-network model based on coarse graining and symbolisation for calculating the distance of stocks [Bri02], which is illustrated to be effective to transform stock price time series into complex networks as well, and it does provide an advantageous attempt for analysing time series from the network perspective.

2.3 Summary

In this chapter a brief literature review and discussion about quantitative finance, stock market investment and applied complex networks theory have been presented. It can be seen that because of the less regulated price float and tremendously available data, the study of financial market complexity is mainly concentrated on stock data. However, it can also be seen that prevailing complex network approaches to analyse stock markets are almost all about investigating weighted or unweight but undirected networks. To our best knowledge, there is no previous work has attempted to construct a directed network so far.

Chapter 3

Pre-processing of stock market and industrial data

3.1 Introduction

This chapter will introduce the sources of economy data and stock data and the methods to pre-process these data.

3.2 Data source

This paper considers 1,418 stocks of listing US companies that were traded consecutively in the NYSE and NASDAQ stock market of US on the trading days from January 4, 2016 to December 30, 2016 and uses daily closing price during this period and the economical use table data from the Industry Economic Accounts (IEAs) of year 2016 in a summary-level of industrial sectors are collected from the official website of Bureau of Economic Analysis, US Department of Commerce [oEA18].

3.3 Economic Input-Output table

The Bureau of Economic Analysis (BEA) in the US publishes Economic Input-Output (EIO) tables each year, which are the transaction matrices of all purchases and sales between sectors in a certain industry group level of a year, i.e. depict how industries provide input to, and use output from, each other to produce Gross Domestic Product (GDP).

This paper uses the use table of 2016. Among the transaction matrix **Z** there are Total Industry Input row **I** at the bottom and the Total Industry Output column **O** at the right are the statistics of total purchase by each sectors and total sales from each sectors respectively. The elements of the normalised direct demand matrix **A** and the direct requirement matrix **B** are:

$$a_{i,j} = -log_{10} \frac{z_{i,j}}{I_j N_j}^{-1} \tag{3.1}$$

and

$$b_{i,j} = -log_{10} \frac{z_{i,j}}{O_i N_i}^{-1} \tag{3.2}$$

respectively. N_i is the number of stocks in the industrial sector which stock i belongs to. Moreover, certain threshold values θ_{DD} and θ_{DR} are specified and a directed edge can be added between stock i and stock j if the value of $a_{i,j}$ is greater than θ_{DD} or the value of $c_{i,j}$ is greater than θ_{DR}.

3.4 Logarithmic return of stock prices

Logarithmic return of a stock in this paper is calculated as the log of the close price of one day divided by the close price of the previous day, which is obtained from the following formula:

$$r_i(\tau) = lnP_i(\tau) - lnP_i(\tau - \Delta t) \tag{3.3}$$

As a proxy for the percentage change in the price, logarithmic return is symmetric and has mathematical conveniences for adding up or subtracting values on the log scale, which are useful for mathematical finance. Therefore, logarithmic return is the measure of price changes in this paper.

3.5 Correlation coefficient

The correlation coefficient between two stocks is considered in terms of the matrix **C**, as the following equation shows:

$$c_{i,j} = \frac{\langle r_i r_j \rangle - \langle r_i \rangle \langle r_j \rangle}{\sqrt{(\langle r_i^2 \rangle - \langle r_i \rangle^2)(\langle r_j^2 \rangle - \langle r_j \rangle^2)}} \tag{3.4}$$

where r denotes the return and the bracket indicates a temporal average over the period. Additionally, a certain threshold value θ_{corr}, $0 \leq \theta_{corr} \leq 1$ is specified, and a directed edge is qualified to be linked between stock i and stock j if the value of $c_{i,j}$ is greater than or equal to θ_{corr}.

3.6 Summary

In this chapter the data sources together with the pre-processing methods have been introduced. Hence, the table of economical data are converted into matrix and the stock price data are transformed to logarithmic return data and correlation coefficients for further analysis.

Chapter 4

Topological properties of directed complex networks

4.1 Introduction

Topological properties of stock complex network are the structural organisations of the interconnections of the system components, e.g., nodes, edges, and the directions and weights of edges, which are also referred to as ”network architecture”.

In the networks of real stock markets, there are numerous common recurring patterns of connections which have profound effects towards the way the complex financial systems behave. This chapter will introduce the critical topological properties implemented in this thesis.

4.2 Degree centrality and strength centrality

The degree of a node k represents the number of its neighbours. In directed network, out-degree k_{out} is the number of edges which start from the given node and end at others, while in-degree k_{in} is the number of edges which end at the given node and start from others. Thus, there is relationship between k_{in} and k_{out}:

$$k = k_{in} + k_{out}. \tag{4.1}$$

As one of the most widespread measures to calculate network centrality, degree centrality of a node can be described as the number of direct links that relate to a specific node [Fre78]. In terms of the directed stock price return network, this paper mainly focuses on the out-degree analysis on the nodes. Moreover, the strength centrality has generally been accumulated to the sum of weights of out-degrees to form the weighted networks. The equation of this measure is shown as bellow:

$$C_D^W(i) = \sum_j^N w_{ij} \tag{4.2}$$

where W represents the matrix of weighed adjacencies, and w_{ij} represents the weight of the link between node i and j.

4.3 Degree distribution and strength distribution

The degree distribution of stock price return network $p(k)$ can be defined as:

$$p_d(k) = \frac{N_k}{N}, \tag{4.3}$$

while N_k represents the number of nodes whose out-degree value is k. The distribution of strength has a similar definition:

$$p_s(w) = \frac{N_w}{N}, \tag{4.4}$$

while N_w represents the number of nodes whose strength value is w.

4.4 Average shortest path length

In a network, the distance between two nodes is the number of edges contained on the shortest path connecting the two nodes. The average path length of the network refers to the average distance of all pairs of nodes in the network. It indicates the degree of separation between nodes in the network and reflects the global characteristics of the network.

The average shortest path length of a directed network G is defined as the following equation:

$$l_G = \frac{1}{n(n-1)} \sum_{i,j \in V} d(i,j) \tag{4.5}$$

where V is the set of nodes of G.

4.5 Betweenness centrality

Other than strength, betweenness centrality [Fre77] can be used to determine the critical nodes among the entire network and to recognise the most associated firms in the chosen stock market. When it comes to weighted networks, betweenness centrality of a node is the sum of the weights in the fraction of all-pairs shortest paths that pass through this node, which can be described as the following equation:

$$C_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)} \tag{4.6}$$

where V is the set of nodes, $\sigma(s,t)$ is the sum of weights of all-pairs shortest (s,t)-paths, and $\sigma(s,t|v)$ is the sum of weights of those paths passing through some node v other than s,t. If $s = t$, $\sigma(s,t) = 1$, and if $v \in s,t$, $\sigma(s,t|v) = 0$.

4.6 Clustering coefficient

The clustering coefficient of a node refers to the ratio of the number of connected edges between all of adjacent nodes of the node to the maximum number of possible edges between these adjacent nodes. The clustering coefficient of the network refers to the average of the clustering coefficients of all nodes in the network, which indicates the clustering of nodes in the network, i.e., the clustering characteristic of the network. It also means that how large is the possibility of two nodes are adjacent nodes if they are both adjacent nodes of a same node, which reflects the local characteristics of the network.

In a nutshell, clustering coefficient is a measure of the degree to which nodes in a network tend to cluster together. Concerning the clustering coefficient of the complex networks, it is defined as:

$$C_i = \frac{2E_i}{(k_i(k_i-1))}, \tag{4.7}$$

where k_i is the degree of a given node v_i, E_i is the real existing edges among the nearest neighbour nodes of the given node v_i, and $k_i(k_i-1)/2$ means the maximum possible edges existing between its nearest neighbours of the node v_i. Besides, the clustering coefficient of a node accounts for the extent to which the transmission relationship between the given node and its neighbours also exists between its neighbours, and the clustering coefficient may be given by:

$$C = \frac{3 \times number\ of\ triangles\ in\ the\ networks}{number\ of\ connected\ triples\ of\ nodes}. \tag{4.8}$$

This measure gives an indication of the clustering in the whole network, and can be applied to both undirected and directed networks.

4.7 Efficiency

Network efficiency measures how efficient for information being conducted and exchanged in the network, which can help to determine whether the objective network shows small-world property. There are global and local efficiencies that on the different scale sizes [LM01].

4.7.1 Global efficiency

Global efficiency quantifies the conduction and exchange of information through out the entire network. The global efficiency of network **G** is defined as:

$$E_{glob}(\mathbf{G}) = \frac{\sum_{i \neq j \in \mathbf{G}} \varepsilon_{ij}}{N(N-1)} = \frac{1}{N(N-1)} \sum_{i \neq j \in \mathbf{G}} \frac{1}{d_{ij}} \tag{4.9}$$

In other words the global efficiency is calculated as the average of each inverse shortest path length among the entire network, hence it is inversely related to the average shortest path length.

4.7.2 Local efficiency

The local efficiency evaluates the resistance of a network towards node i and quantifies the conduction and exchange of information among its neighbours. The local efficiency of node i in network **G** is defined as:

$$E_{loc}(G,i) = \frac{1}{N} \sum_{i \in \mathbf{G}} E_{glob}(\mathbf{G}_i) \tag{4.10}$$

Therefore it can be seen that the local efficiency is related to the clustering coefficient.

4.8 Assortativity and degree correlations

The phenomenon of assortative [New02] mixing can be quantified by means of an assortative coefficient. Let E_{ij} be the number of edges in the network that connect a vertex of type i to one of type j, with $i, j = 1, ..., n$, then similar in spirit to the adjacency matrix for vertices, these edges can be represented in the form of an edge incidence matrix $\mathbf{E}$, with elements E_{ij}. A normalized mixing matrix is defined as follows:

$$\mathbf{e} = \frac{\mathbf{E}}{||\mathbf{E}||}, \tag{4.11}$$

where $||\mathbf{E}||$ refers to the sum of the elements of the matrix $\mathbf{E}$. The entries e_{ij} in the normalized matrix represent the fraction of edges that connect vertices of types i and j, and satisfies the normalization condition,

$$\sum_{ij} e_{ij} = 1. \tag{4.12}$$

The assortativity coefficient r is then defined thus,

$$r = \frac{Tr(\mathbf{e}) - ||\mathbf{e}||^2}{1 - ||\mathbf{e}||^2}, \tag{4.13}$$

where $Tr(\mathbf{e})$ is the standard matrix trace—the sum of the diagonal elements e_{ii}. The value of the coefficient r lies in the range $-1 \leq r \leq 1$, where 1 represents a perfectly assortative network, 0 a randomly mixed one and -1 a perfectly disassortative network.

Since the degree is an important topological measure, degree correlations assume a significant amount of relevance as they can give rise to complicated network structural effects. The degree correlation can be computed using Eqn. 4.13, where the elements e_{ij} represent the fraction of edges that connect a vertex of degree i to that with degree j.

4.9 Modularity

Modularity stands for the difference between fraction of links that fall within communities and the expected fraction if links are randomly distributed [NG04]. This project introduces modularity as a measure to evaluate the connection strength between node pair within a group. Regarding to the industry where the stocks belong to, these stocks are divided into different groups hence modularity is used to measure the closeness of intra- and inter-group.

Two groups are combined to generate the modularity value while computing the closeness of two groups, as formula below shows:

$$Q = \frac{1}{2m} \sum_{j} \left[w_{ij} - \frac{k_i k_j}{2m} \right] \delta\left(c_i, c_j\right) \tag{4.14}$$

where c_i is the community to which node i is assigned, and k_i represents the degree of node i. The δ-function $\delta(u,v)$ is 1 if $u = v$ and 0 otherwise and $m = 0.5 \sum_{ij} w_{ij}$ is the sum of weights in the whole network.

4.10 Summary

In this chapter the critical topological properties for this thesis have been introduced. Corresponding properties will be calculated if they are applicable for the directed-unweighted or directed-weighted network during the empirical practice.

Chapter 5

Community detection for directed networks

5.1 Introduction

In this chapter, the basic theory and algorithm of community detection for directed networks will be introduced. It starts with the concepts of modularity score and modularity optimisation method. Then, it will be discussed the eigenvector and eigenvalue in a matrix for finding the optimal nodes assignments of communities. Finally, an practical algorithm will be illustrated.

5.2 Community detection algorithm

This thesis considers a theoretic concept of modularity-based community detection method for directed graphs to recognise natural faults occur in the stock network along which it partitions. Community detection is applied for further understanding to the overall pattern of economical and stock price relations of listed companies.

While there are many methods to identify communities in undirected graphs, the community detection method used in this thesis is for the directed networks proposed by Leicht and Newman [LN08], which based on modularity optimisation method. Modularity optimisation method identifies communities by maximizing the modularity Q, which is defined as:

$$Q = (\textit{fraction of intra-community edges}) - (\textit{expected fraction of such edges}) \quad (5.1)$$

It signifies that a community is figured when the number of edges inside the community is more than the expected number on the basis of chance. As a result, modularity-based community detection maximised intra-community density and minimised inter-community density. While the complexity of modularity optimisation is NP-complete problem, this paper uses the spectral optimisation methodology, which finds the best partition of the directed US stock network by the following expression of Q:

$$Q = \frac{1}{m}\sum_{ij}\left[A_{ij} - \frac{k_i^{\mathrm{in}} k_j^{out}}{m}\right]\delta_{c_i,c_j} \tag{5.2}$$

where A_{ij} is defined to be 1 if there is an edge from j to i and 0 otherwise. k_i is the in-degree for node i, k_j is the out-degree for node j, m is the total number of edges in the network. δ is the Kronecker delta symbol that is 1 if nodes i and j are in the same community, i.e., $C_i = C_j$, and 0 otherwise. Spectral optimisation technique for modularity maximisation assigns nodes to different communities based on the sign of the eigenvector, corresponding to the largest positive eigenvalue of the modularity matrix **B**, whose elements are:

$$B_{ij} = A_{ij} - \frac{k_i^{in} k_j^{out}}{m} \tag{5.3}$$

This thesis applies the repeated bisection graph-partitioning algorithm in the cause of community detection according to Leicht and Newman [LN08]. This approach begins with partition the network in two and then repeating it while optimising for the maximum modularity score of the communities. A preferred partition of a network results in a higher modularity score, therefore the modularity Q is maximised over all possible partitions of the stock network to detect communities of listed companies.

The following algorithm describes the details about the partitioning process and maximising modularity score in community detection. The functions of calculating modularity and subdividing node group are called repeatedly over iterations until no further increment of the overall modularity score.

Algorithm 1 Community detection

procedure COMMUNITY(*G, nNode, nEdge, EntireModMat, EntireNodeSpace*)
  procedure CALDELTAQ(**s**, **B**)
    return $Q \leftarrow 1\,/\,(4 * nNode) * \boldsymbol{s}^T(\boldsymbol{B} + \boldsymbol{B}^T)\boldsymbol{s}$
  procedure UPDCOMMUNITYASSIGNMENT(*NodeSpace*, *UpdAssign*)
    $Mark1, Mark2 \leftarrow \max(Assignment) + 1, \max(Assignment) + 2$
    for each $node \in NodeSpace$ **do**
      if $node \in UpdAssign > 0$ **then** node of $Assignment \leftarrow Mark1$
      if $node \in UpdAssign < 0$ **then** node of $Assignment \leftarrow Mark2$
    return *Assignment*
  procedure SUBDIVIDECOMMUNITY(**B**)
    $SymmetricMatrix \leftarrow \mathbf{B} + \mathbf{B}^T$
    $eigv \leftarrow$ eigenvector as $\max(eigenvalues)$ in *SymmetricMatrix*
    return $\operatorname{sign}(eigv)$
  procedure CALMODULARITY(*assignment*)
    for each $node1 \in Nodes\ of\ G$ **do**
      for each $node2 \in Nodes\ of\ G$ **do**
        if *assignment* of $node1 \leftarrow$ *assignment* of *node2* **then**
          $Q \leftarrow Q + HasEdge - (\mathrm{nIn}(node1)) * (\mathrm{nOut}(node2))/(\mathrm{nEdge})$
    return $Q/(nEdge)$
  procedure GENMODULARITYMATRIX(*NodeSpace*, *ModMat*)
    for each $node1 \in NodeSpace$ **do**
      for each $node2 \in NodeSpace$ **do**
        $B \leftarrow HasEdge - (\mathrm{nIn}(node1)) * (\mathrm{nOut}(node2)/\mathrm{nEdge})$
        if Assignment of *node1* = Assignment of *node2* **then**
          for each $node \in NodeSpace$ **do** $C \leftarrow C + HasEdge1 + HasEdge2 - (\mathrm{nIn}(node1) * \mathrm{nOut}(node) + \mathrm{nIn}(node) * \mathrm{nOut}(node1))/\mathrm{nEdge}$
  procedure INTERATEBISECTION(*ModMat, NodeSpace*)
    $UpdAssign \leftarrow \mathrm{SubdivideCommunity}(ModMat)$
    $DeltaQ \leftarrow \mathrm{CalDeltaQ}(UpdAssign, ModMat)$
    if $DeltaQ > 0$ **then**
      $Assignment \leftarrow \mathrm{UpdCommunityAssignment}(NodeSpace, UpdAssign)$
      for each $side \in UpdCommunityAssignment$ **do**
        $ModMat \leftarrow \mathrm{GenModularityMatrix}(NodeSpace)$
        $\mathrm{InterateBisection}(ModMat, NodeSpace)$
  $\mathrm{InterateBisection}(EntireModMat, EntireNodeSpace)$
  return *Assignment*

5.3 Summary

In this chapter, the basic theory and algorithm of community detection for directed networks have been introduced. This modularity-based community detection method allows us to produce more accurate partitioning of the directed network than the methods only for undirected networks.

Chapter 6

Benchmarking networks generation

6.1 Introduction

Numerous literatures support the idea that undirected stock networks have small-world features. It can be helpful to examine whether the directed one is a small-world network or not by comparing it with conventional and acknowledged random network and small-world networks. This chapter will introduce the directed forms of such two conventional and basic networks: Erdős–Rényi random network and Watts-Strogatz small-world network.

6.2 Directed Erdős–Rényi random network

To some extent, the regular network and the random network are two extremes while the complex network is between them. In random network, nodes are connected in a purely random manner, hence the resulting network is called a random network. If the nodes are wired in a self-organising manner, it will then evolve into a variety of different complex networks.

The Erdös–Rényi (ER) model [ER59] generates a graph that winded randomly between N nodes in the network with probability p. The degrees of nodes comply with a Poisson distribution, indicating that most nodes have approximately same number of edges.

Erdős and Rényi has found that as the number of edges M gradually increases from a small value, the random graph will evolve from a fragmented graph with many independent components to a fully connected one [Str01].

Algorithm 2 ErdosRenyiRandomNetwork

procedure GENERATERANDOMNETWORK(*nNodes, p0, edges*)
  Initialize:
  $G \leftarrow$ Directed graph with nodes in *nNodes*
  for each $e \in edges$ **do**
    if random number between 0 and $1 < p0$ **then** add edge e to G
  return G

6.3 Directed Watts-Strogatz small-world network

The nearest neighbour coupled regular network is highly clustered, but it is not a small-world network. On the other hand, the ER random network has a small average path length but without high clustering characteristics. Therefore, neither of these two types of network models can reproduce some important features of the real network, for most of the actual networks are neither completely regular nor completely random. As a transition from a fully regular network to a completely random network, Watts and Strogtz introduced a small-world network model called the WS small-world model [WS98].

In the complex networks theory, a network with both a small average path length and a large average clustering coefficient feature is called a small-world network. In the WS model, when the random reconnection probability p of the connected nodes is gradually increased from 0 to 1, it can be observed that the initial regular network will go through the following three phases: regular network, small-world network, and eventually random network.

This paper uses an alternative method based on WS model [SW14]. Specify the number of nodes N, the mean degree K (assumed to be an even integer), and a special parameter β, satisfying $0 \leq \beta \leq 1$ and $N \gg K \gg \ln N \gg 1$, the model constructs an undirected graph with N nodes and $NK/2$ edges as the following algorithm depicts:

Algorithm 3 WattsStrogatzSmallWroldNetwork

procedure GENERATESMALLWORLDNETWORK(*nNodes, p0, beta*)
    $Dmax \leftarrow nNodes$ % 2
    $R \leftarrow range from 1 to Dmax$
    $D \leftarrow$ circulant matrix of $R/Dmax$
    $p \leftarrow beta * p0 + ((D \leq p0) * (1 - beta))$
    $A \leftarrow 1 *$ (randomised matrix $p < p$)
    fill diagonal of matrix A
    fill diagonal of matrix A
    $G \leftarrow$ Directed graph corresponds to A
    return G

6.4 Summary

In this chapter, the directed forms of ER random network and WS small-world network have been introduced. In practice these two kinds of networks will be generated by the same number of nodes and edges, and their topological properties can be deemed as the benchmarks under certain circumstances to compare with the stock price return networks.

Chapter 7

Empirical study and results

7.1 Stock market description

There are totally 261 trading days in 2016 of US stock market, this thesis selects 1,418 stocks of listing US companies that were traded in all trading days in 2016. Table 7.1 lists the titles of 55 industrial sectors corresponding to the summary level of BEA industry codes as well as the number of stocks in each of them.

Industrial Sector Title	**Stock Count**
Banks, credit intermediation, and related activities	214
Computer and electronic products	173
Funds, trusts, and other financial vehicles	132
Insurance carriers and related activities	85
Chemical products	76
Utilities	64
Food and beverage and tobacco products	56
Fabricated metal products	52
Securities, commodity contracts, and investments	42
Broadcasting and telecommunications	42
Other retail	38
Machinery	36
Wholesale trade	33
Motor vehicles, bodies and trailers, and parts	32
Construction	30
Computer systems design and related services	27

Performing arts, spectator sports, museums, and related activities	27
Miscellaneous professional, scientific, and technical services	23
Petroleum and coal products	16
Paper products	15
Air transportation	14
Data processing, internet publishing, and other information services	14
Ambulatory health care services	12
Plastics and rubber products	12
Accommodation	12
Administrative and support services	11
Truck transportation	11
Rental and leasing services and lessors of intangible assets	10
Other transportation and support activities	9
Publishing industries, except internet (includes software)	8
Other transportation equipment	8
Support activities for mining	7
Other real estate	7
Miscellaneous manufacturing	7
Oil and gas extraction	6
Furniture and related products	6
Electrical equipment, appliances, and components	6
Rail transportation	5
Textile mills and textile product mills	5
Nonmetallic mineral products	5
Transit and ground passenger transportation	4
Waste management and remediation services	3
Hospitals	3
Wood products	3
Printing and related support activities	2
Motion picture and sound recording industries	2
Nursing and residential care facilities	2
Pipeline transportation	2
Primary metals	2
Apparel and leather and allied products	2
Other services, except government	1

Water transportation	1
Legal services	1
Social assistance	1
Mining, except oil and gas	1
Total	**1,418**

Table 7.1: Part of counts for US stocks by industry [SEC18]

It is not hard to see the composition of stock market are mainly dominated by the finance-related industry (”Banks, credit intermediation, and related activities”, ”Funds, trusts, and other financial vehicles”, ”Insurance carriers and related activities”, ”Securities, commodity contracts, and investments”, etc.) and computer-related industry (”Computer and electronic products”, ”Computer systems design and related services”, ”Data processing, internet publishing, and other information services”, ”Electrical equipment, appliances, and components”, etc.). The total numbers of finance-related industry and computer-related industry are over 473 and 220 respectively, which jointly take almost half of the number of total stocks. Therefore, it is necessary to use the formalised formula which divides the value by the number of stocks in its belonging industrial sector, punishing the connection from or to a node by the industry size, i.e., if a stock is in a large industry, it will need higher transaction flows to connect to other nodes in the network.

7.2 Networks construction

This thesis first generated matrices of normalised direct demand **A**, normalised direct requirement **B**, correlation coefficient **C** using formula 3.1, 3.2, and 3.4.

Figure 7.1 from normalised direct demand **A** and normalised direct requirement **B** illustrates that the transaction densities decrease as threshold of normalised direct requirement and normalised direct demand increase, and their patterns are very similar with the same inflection point at around $threshold = 0.136$ where the densities begin to decline. Therefore, the values of thresholds for normalised direct requirement and normalised direct demand are set to be equal, i.e., $\theta_{EIO} = \theta_{DD} = \theta_{DR}$, to filter the directed edges among the stock network.

Figure 7.2 shows the distribution of stock price correlation coefficients has a shape complies to the normal distribution. Most correlation coefficients are vary from -0.2 to 0.85 with the mean of 0.265. Figure 7.3 also shows that the edge density drops

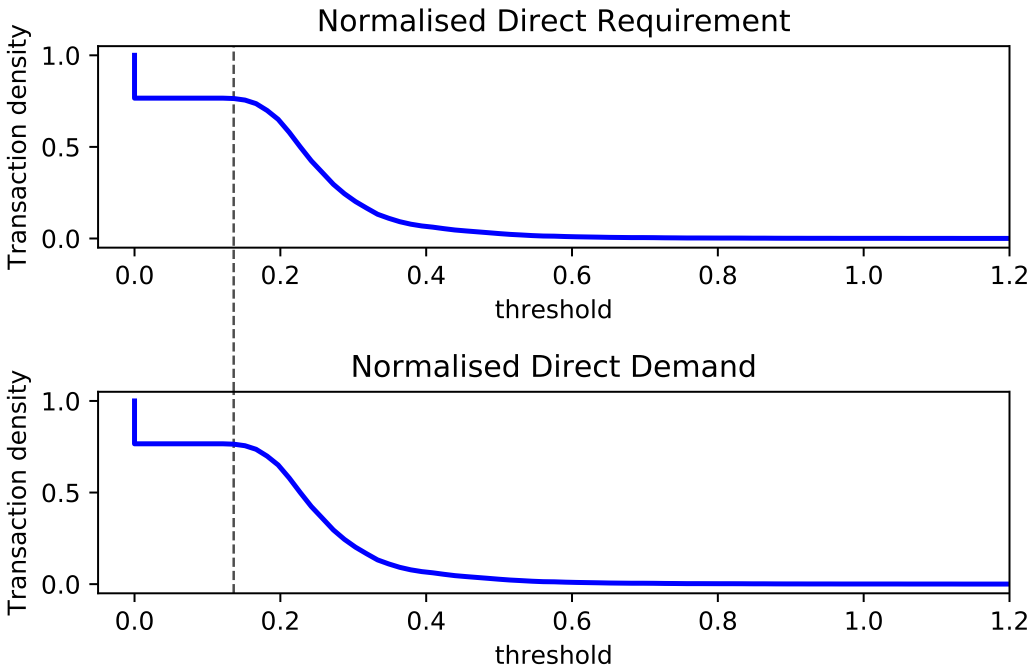


Figure 7.1: Transaction densities in EIO. The density of transactions drops vertically at the threshold of 0, which means nearly a quarter of values in the normalised direct demand matrix **A** and normalised direct requirement **B** are 0. Then the two densities both decrease from the point around 0.136, and overall they follow a same pattern.

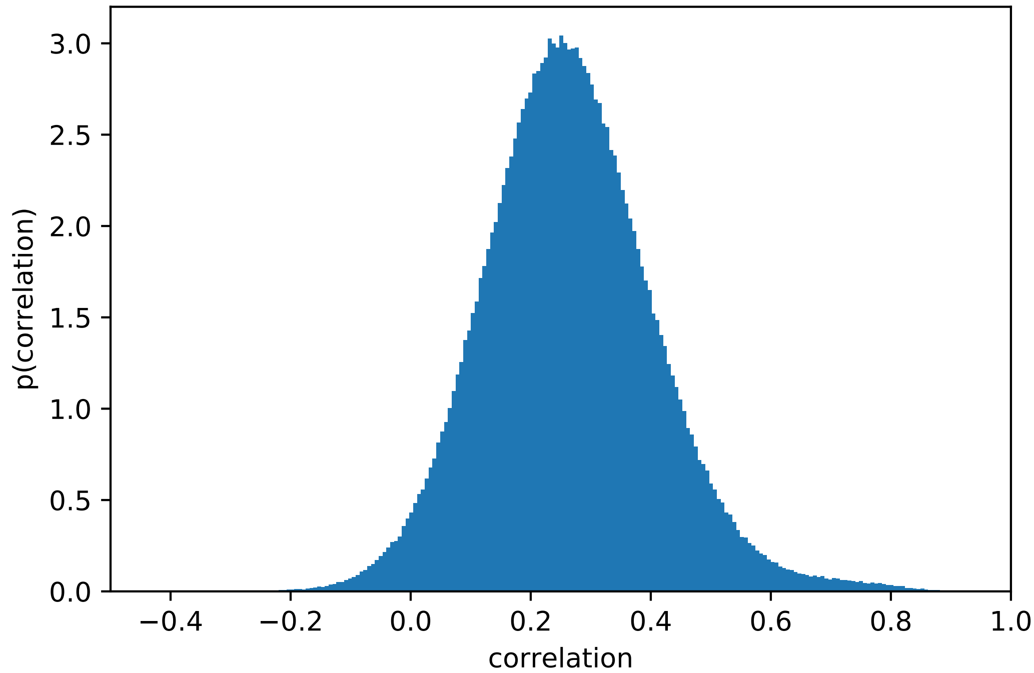


Figure 7.2: Correlation coefficient distribution of stock price return. The minimum and maximum are -0.687 and 0.977 and the mean is 0.265. The distribution follows normal distribution.

dramatically as the correlation coefficient increases from 0 to 0.50. It implies that the prices of most stocks traded in NYSE and NASDAQ often fluctuate to the same direction, but the patterns are less similar to each other.

Figure 7.4 shows the number of directed edges remain at the conditions of different value combinations of $\{\theta_{EIO}, \theta_{corr}\}$. When both of the thresholds set to be minimal at their own value range, i.e., $\theta_{EIO} = 0$ and $\theta_{corr} = -1$, the number of directed edges is $N \times (N-1) = 2{,}009{,}306$, while N indicates the total number of nodes, which is 1418. According to the figure 7.4, the number of edges will be less than $100{,}000$, in which case the network has a density of lower than 5%, if $\theta_{EIO} \geq 0.3545$ or $\theta_{corr} \geq 0.5020$.

It is obvious that the larger values assigned to θ_{EIO} and θ_{corr}, the more significant will be for the weights and directions of the remaining edges. But if the network becomes too sparse, it can not be strongly or even weakly connected and there would be many independent cliques, hence the network becomes too inefficient to be a senseful network. As a result, this paper selects the threshold-value-pair $\{\theta_{EIO} = 0.292, \theta_{corr} = 0.379\}$ to construct a directed-unweighted network and a directed-weighted network

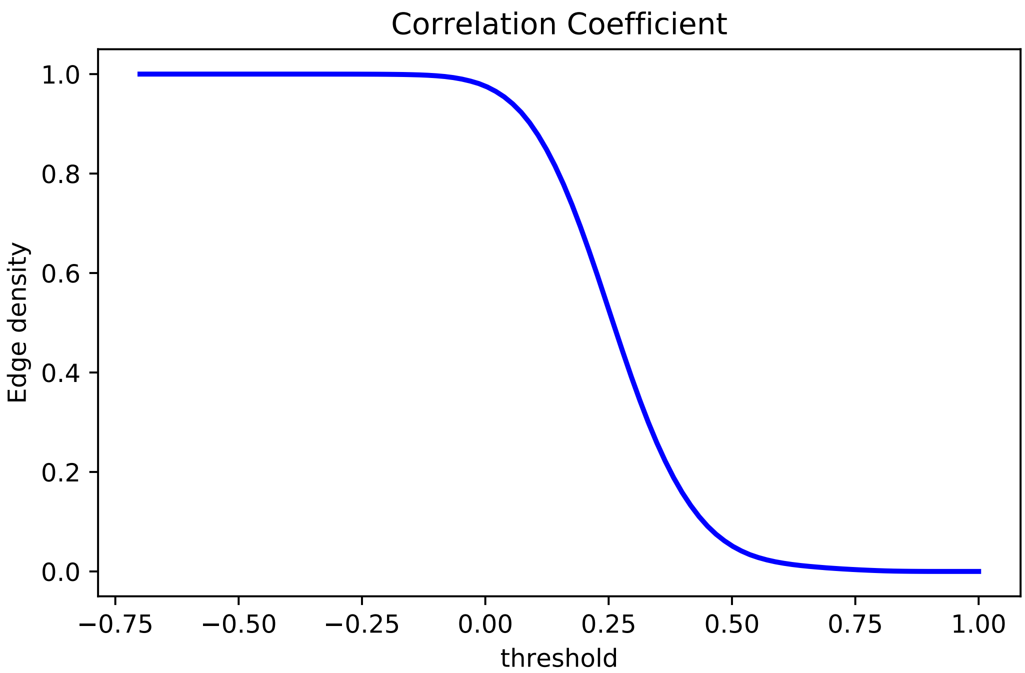


Figure 7.3: Edge density with correlation coefficient. The edge density drops dramatically as the correlation coefficient increases from 0 to 0.50.

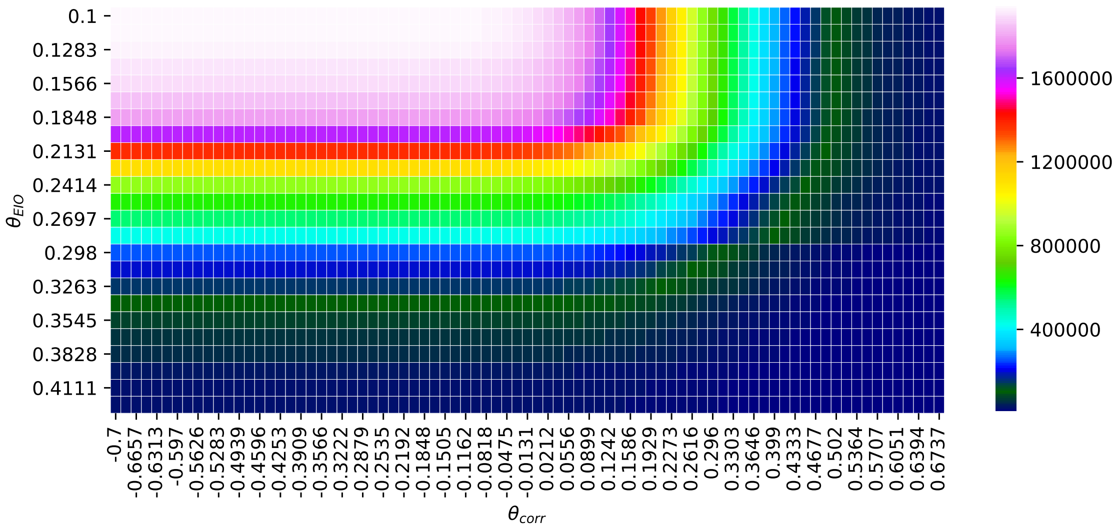


Figure 7.4: Heatmap of the numbers of directed edges per EIO-threshold and correlation-coefficient-threshold. The number of directed edges changes from the maximum to 0 as θ_{EIO} decreases from 0.10 to 0.45 and θ_{corr} increases from -0.70 to 0.68.

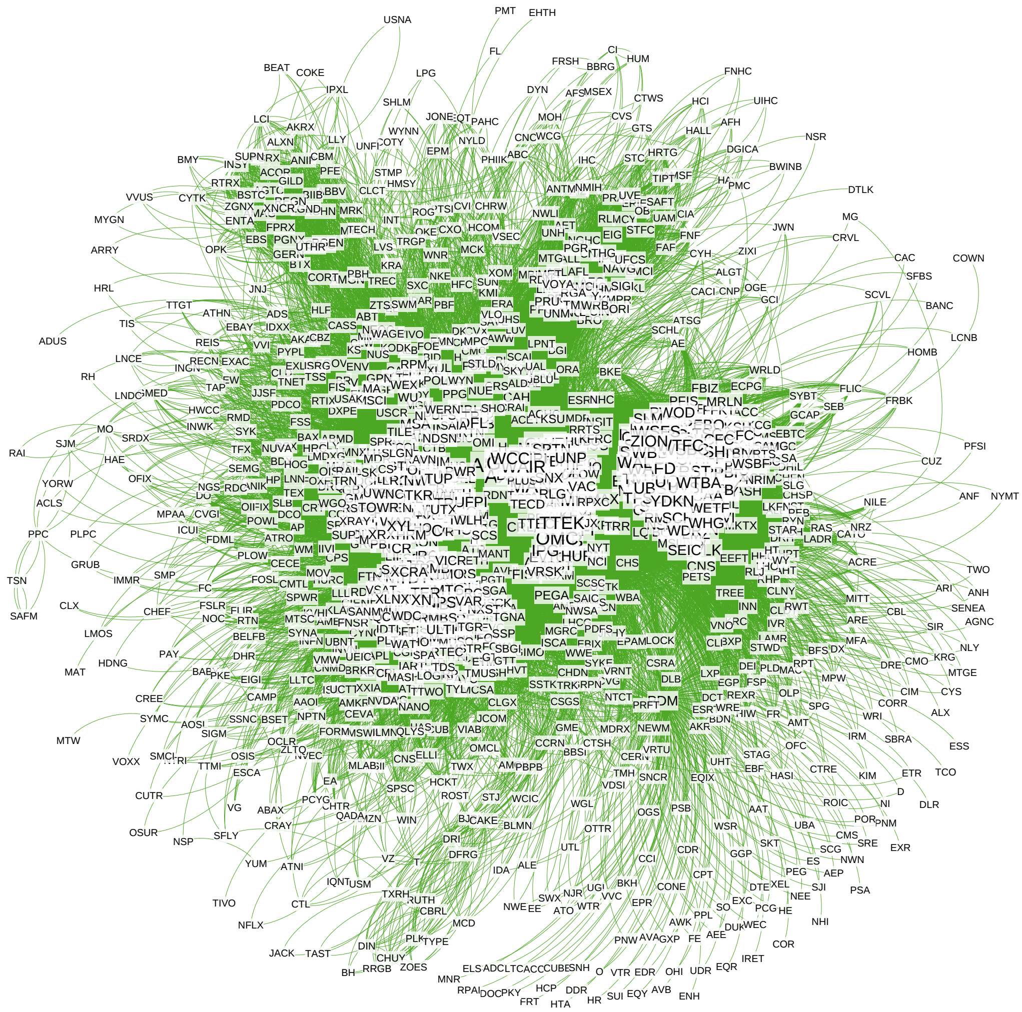


Figure 7.5: Visualisation of the directed-unweighted stock price return network. The stock codes are nodes and the clockwise rotations of edges are directions. Graph is generated by Gephi.

Directed networks	Stock price return	WS small-world	ER random
Number of nodes	1418	1418	1418
Number of edges	102051	102051	102051
Out-degree distribution	Power-law	Normal	Normal
Average out-degrees	143.94	143.94	143.94
Average path length	2.775	2.005	1.973
Clustering coefficient	0.4675	0.1367	0.05105
Global efficiency	0.2563	0.5161	0.5216
Local efficiency	0.6276	0.5027	0.4456
Assortativity	0.02004	-0.002180	0.001452

Table 7.2: Main properties of stock network, small-world network, and random network

for the 2016 US stock market. Figure 7.5 shows the visualisation of the directed-unweighted stock network.

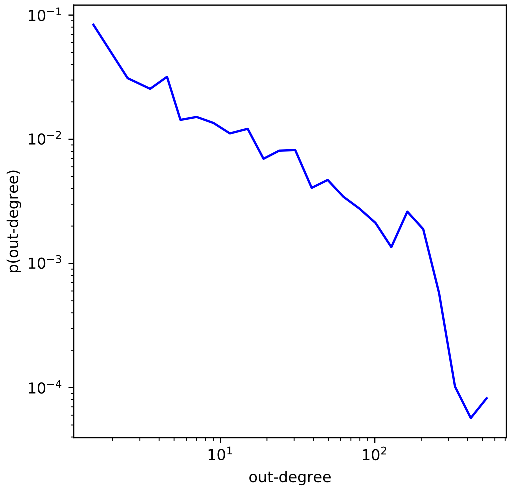
7.3 Analysis of the directed-unweighted stock network

A directed WS small-world network and a directed ER random network with the same number of nodes and edges with the stock directed-unweighted network are generated according to algorithms 3 and 2. Table 7.2 compares the main topological properties of the three networks, which will be discussed together with some other measures in the following sections.

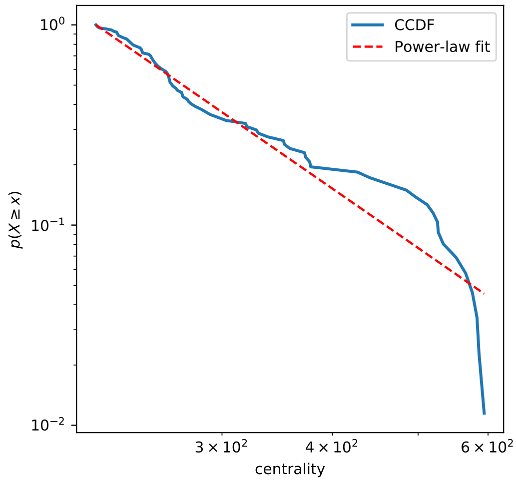
7.3.1 Power-law distribution

According to the table 7.2, the values of average out-degrees of directed stock network, WS small-world network and ER random network are exactly the same due to the identical numbers of nodes and edges, but in terms of the distributions of out-degrees, stock network is totally different from the others.

The distribution and P-P plots in figures 7.7 and 7.8 show clearly that the out-degree distributions of WS small-world network and ER random network fitted nicely to the normal distribution, because most degrees of nodes fall in the middle range, especially in the P-P plots, the sample data points are basically on the diagonal representing the theoretical normal distribution for both WS small-world network and ER random network. Nonetheless, figure 7.6a illustrates that for the stock price return network, only



(a) Empirical out-degree distribution



(b) CCDF and power-law fit

Figure 7.6: Out-degree distribution of directed stock price return network

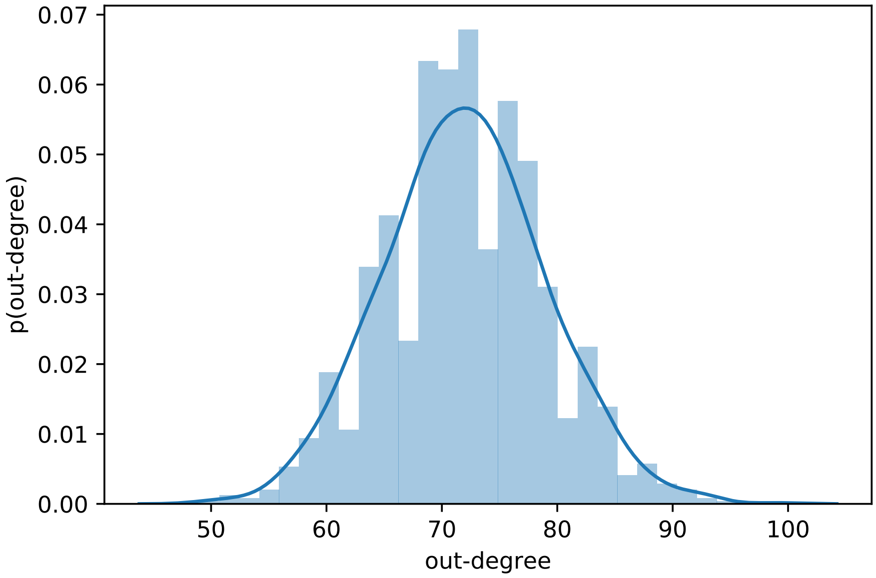
a few number of nodes show higher out-degree, while most nodes are at the positions of low out-degree level. Statistical result shows the distribution of the directed stock price return network follows power-law distribution with the exponent of 4.057.

The discovered power-law distribution property reveals that in the aspect of degree the directed stock network shows the continuity with conventional undirected stock networks in previous studies. Therefore in general, most of the nodes have a small degree while a few modes have a higher degree for both directed and undirected stock networks.

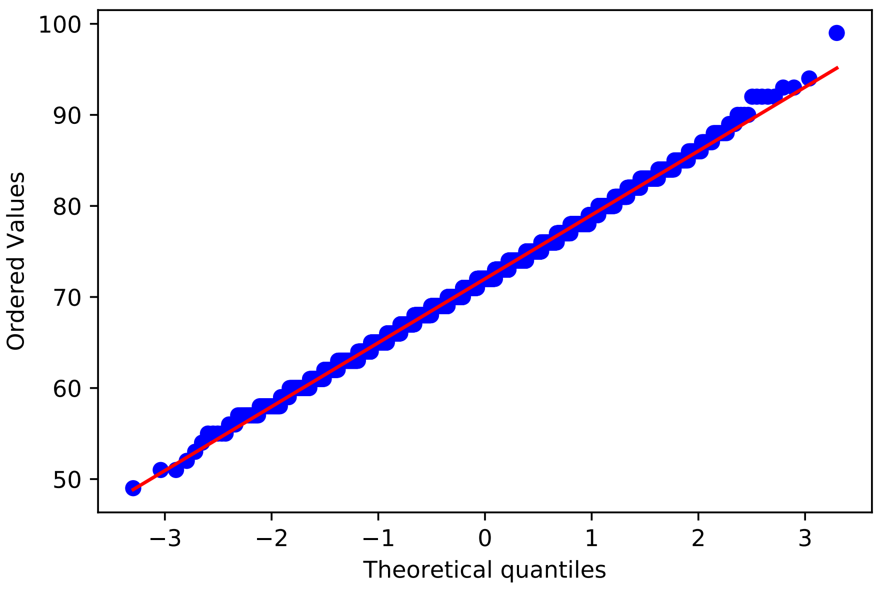
7.3.2 Small-world property

Previous researches upon undirected stock networks have argued that they have small-world topologies. Such feature is also applied to the directed WS small-world network and ER random network, for the average path lengths of around 2, indicating that if we take any node in the network, it can be expected to reach any other nodes just through one node as the medium. For the directed stock network, the expectation number of medium nodes is 1.775, which can be also treated as a small number for network connectedness.

On the other hand, the global efficiency of the benchmarking networks are slightly higher than $1/l_G$ which are both around 0.5. It means that in physical terms the flow

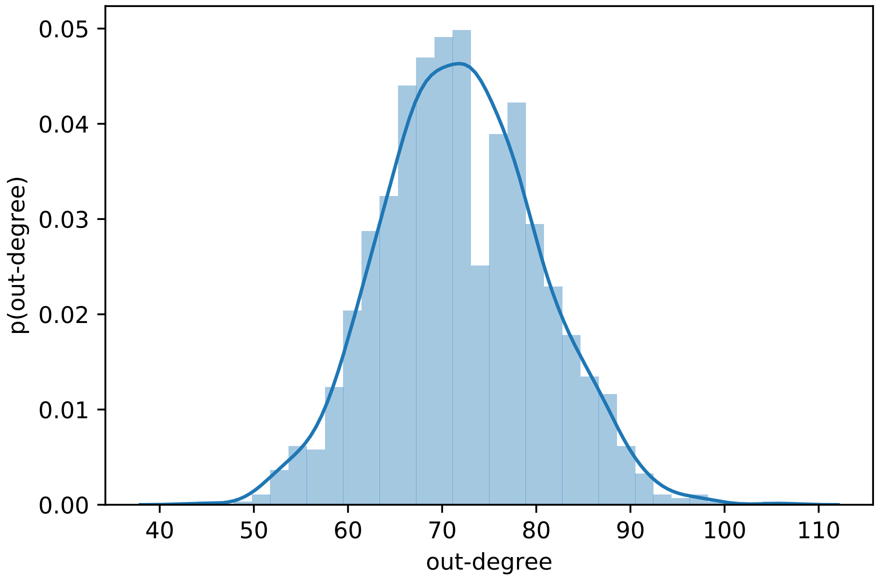


(a) Empirical out-degree distribution

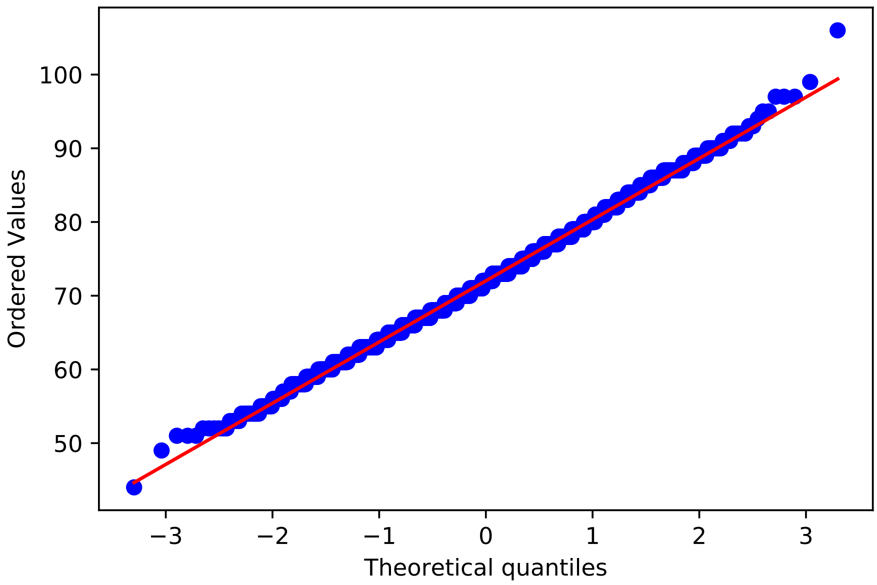


(b) P-P plot

Figure 7.7: Out-degree distribution and P-P plot of small-world network



(a) Empirical out-degree distribution



(b) P-P plot

Figure 7.8: Out-degree distribution and P-P plot of random network

of information in these two networks is efficient, which is the behaviour of small-world networks. While as for the directed stock network, its global efficiency is less than $1/l_G$ which is 0.36. In essence, like directed WS small-world and ER random networks, directed stock network also has the small-world properties to some degree, while it is not efficient to exchange information across the network in the global scale.

7.3.3 Clustering feature

The directed ER random network shows no clustering feature because its clustering coefficient is close to zero (0.05105). For the directed WS small-world network, its clustering coefficient of 0.1367 shows slight clustering feature. While for the directed stock network, its clustering coefficient of 0.4675 is much higher, which shows an significant clustering feature. The indicator of local efficiency also support this conclusion because it reveals the neighbours of a node in the directed stock network are more efficient when conduct information than the other two benchmarking networks, therefore the nodes in the stock network tend to cluster together in higher degree.

The assortativity values for the three networks are all non-significant, for the two benchmarking networks this is corresponding to the aforementioned non-clustering feature and the normal distribution of degrees. However, for the stock network, the incredibly low assortativity together with the aforementioned power-law distribution of degrees indicate that the nodes in this network tend to connect to other nodes with high degrees.

7.3.4 Community structure of the directed-unweighted stock network

The larger value of clustering coefficient for stock network than the other two networks indicating that the nodes in stock network tend to cluster together. Therefore, communities of stock network will be identified implementing the *algorithm 1* for directed networks in this section. According to the composition of industrial sectors of each community, as figure 7.11 shows, the following five communities are identified: (1) Production (2) Finance (3) Livelihood (4) Insurance and chemical products (5) Utilities and financial vehicles.

The communities of production (purple) and livelihood (blue) are sparsely distributed while there are some large-sized nodes acting as hubs of the overall network. The hubs not only connect to the nodes of same communities, but also the externals.

These two communities are partially intertwined due to the high relevancy of production industry and livelihood industry.

Unlike the above two communities, it can be seen from figure 7.9 that the community of finance (green) is decentralised, i.e., there is no obvious hubs and the degrees of each node distribute evenly. It also has a very dense structure, connected closely inside and completely exclusive from other nodes or communities. This means the co-movements among financial stocks are incredibly strong and economically they rely tightly to each other.

The other two communities are more interesting because of their peculiar structural features. Every industrial sectors of individual stocks in community are identified to investigate the properties of the community of insurance and chemical products (yellow). As figure 7.10d illustrates and through the investigation, almost all firms in the upper and lower clusters are in the sectors of ”chemical products” and ”insurance carriers and related activities” respectively, while firms between the two big clusters, like ”MCK” (McKesson) and ”CAH” (Cardinal Health), are large medical supplier, pharmaceutical and heathcare service companies with high out-degrees to both of the two clusters. Apart from that, there are also a considerable number of links from the nodes in upper cluster to the hubs of chemical companies. Thus, it is reasonable to infer that the prices of medicines have significant influence to medical insurance industry, additionally the purchases of chemical products of pharmaceutical firms and the sales of chemical products have made pharmaceutical and chemical companies influence to each other.

Another investigation towards the community of utilities and financial vehicles (orange) is conducted by the same measure. As figure 7.10e illustrates, there is only one huge hub (PDM) which is the company ”Piedmont Office Realty Trust” among the whole community while all the others are one-degree nodes located remotely. There are more links from the hub to the rest than the opposite direction, and also the weights of the former links are generally higher. The hub, ”Piedmont Office Realty Trust”, is a real estate investment trust company, and the rest in the community contains 59 ”funds, trusts, and other financial vehicles” firms and 44 ”utilities” firms. For a big realty trust enterprise, demand for financial trust business is extremely high, and its successes of investments upon real estates will promote the development of utilities companies. It depicts that the major realty trust enterprise alone has significant influence to all of these financial trust and utilities companies.

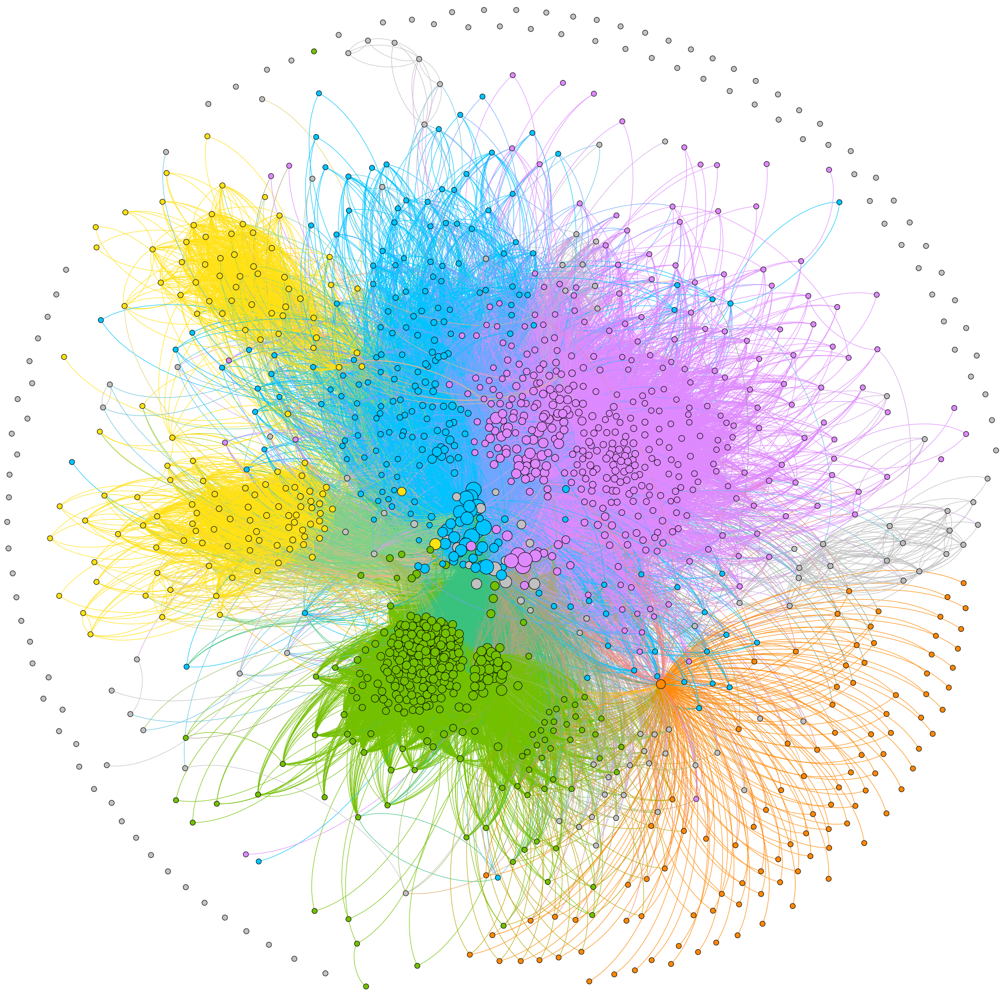
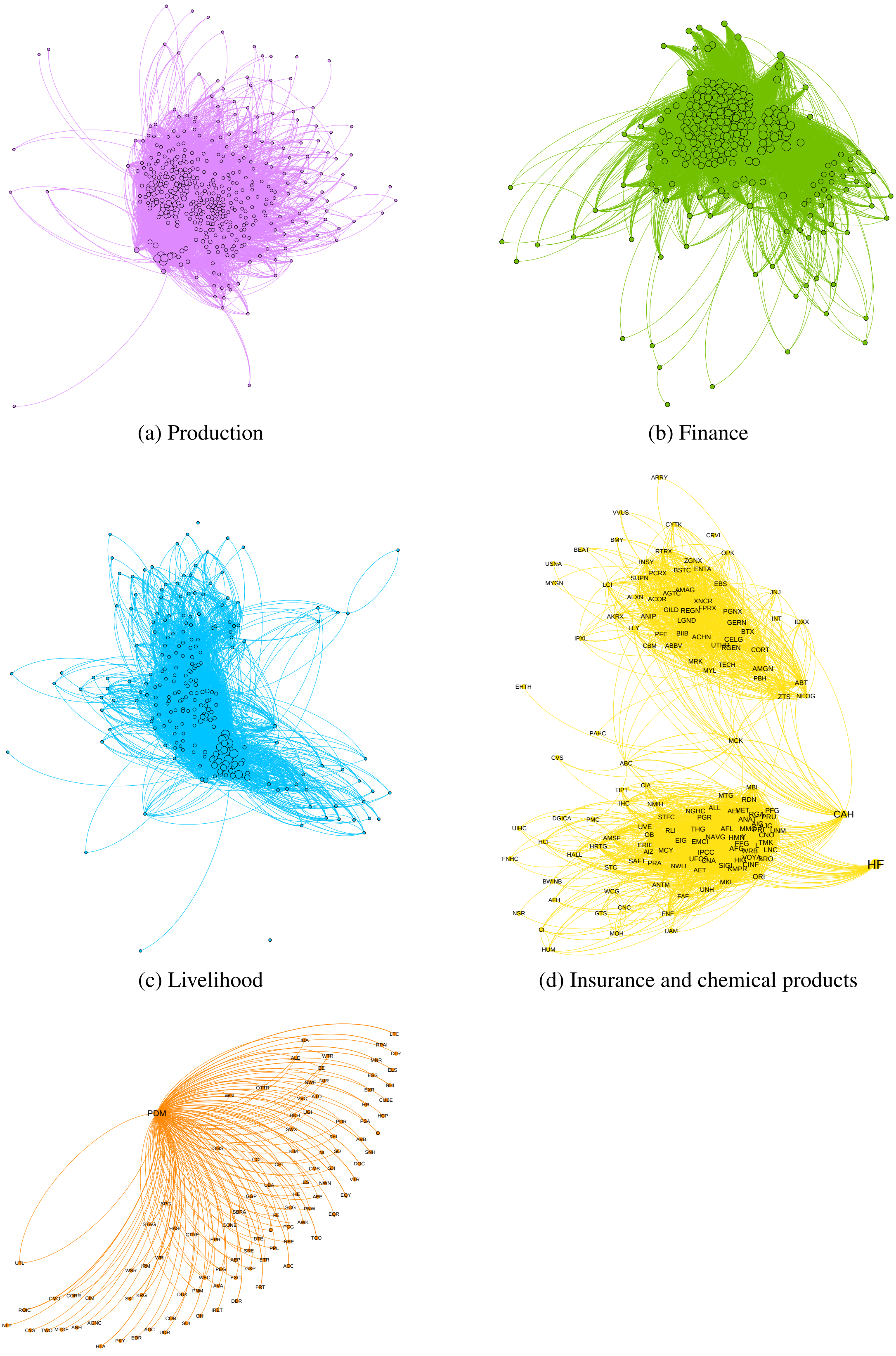


Figure 7.9: Community structure of the 2016 US stock price return network. Five distinct communities are detected represented by different colours of nodes. The direction of edge is clockwise. The size of nodes and thickness of edges are related to the value of degrees and weights. The grey nodes do not belong to any communities and most of them have zero degree. Graph is generated by Gephi.



(a) Production

(b) Finance

(c) Livelihood

(d) Insurance and chemical products

(e) Utilities and financial vehicles

Figure 7.10: Community sole views of the directed stock network. Stock tickers are displayed for the sparsely distributed communities. Graphs are generated by Gephi.

Weighted stock network	Directed	Undirected
Number of nodes	1418	1418
Number of edges	102051	51037
Strength distribution	Power-law	Power-law
Average strength	40.89	42.31
Average betweenness centrality	0.0007440	0.0007464
Weighted assortativity	0.1244	0.06138

Table 7.3: Main topologies of weighted stock networks

According to the stacked bar chart in figure 7.11, almost all firms in the industrial sectors of ”Computer and electronic products”, ”Construction”, ”Data processing, internet publishing, and other information services”, ”Electrical equipment, appliances, and components”, ”Machinery”, ”Motor vehicles, bodies and trailers, and parts”, ”Nonmetallic mineral products”, ”Other transportation equipment”, ”Waste management and remediation services”, ”Wood products” are partitioned into the community of production.

7.4 Analysis of the directed-weighted stock network

A directed-weighted stock network is generated by adding the correlation coefficients in the matrix **C** as weights to each existing directed edges from the directed-unweighted stock network. As the weighted form of the directed stock network that discussed in previous sections, this section will only focus on its features about the weights.

7.4.1 Topological properties on weighted networks

A conventional undirected-weight network is constructed independently and through the threshold of correlation coefficient we can adjust the number of undirected edges remain. When converting an undirected network to directed network, the number of edges are doubled because each undirected links are generated into two directed links with opposite directions for remaining original network topological features unchanged. Same reason is applied for constructing the conventional undirected-weighted stock network with equivalent topologies with the directed stock network. Hence, table 7.3 compares the topological properties between them.

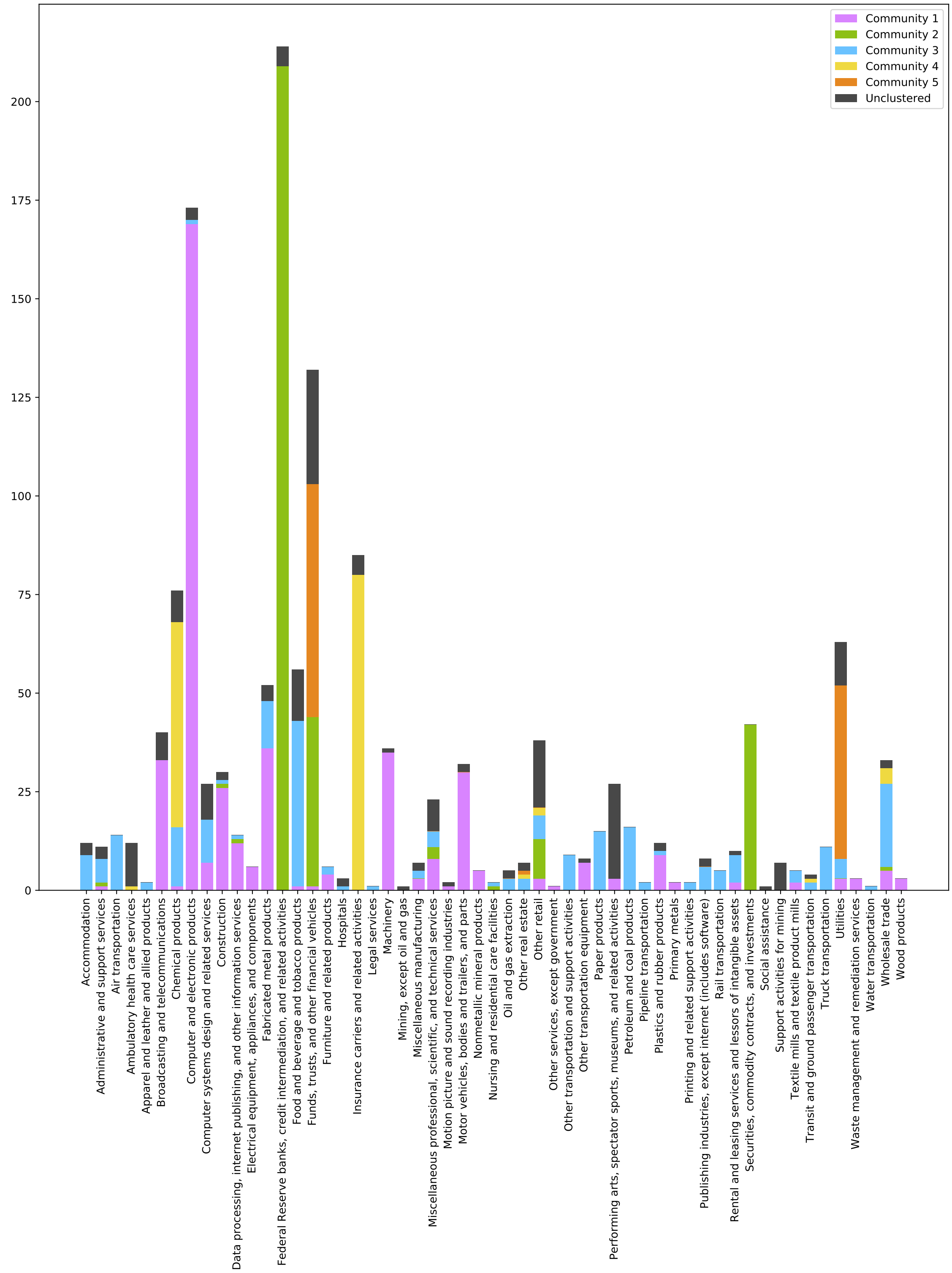
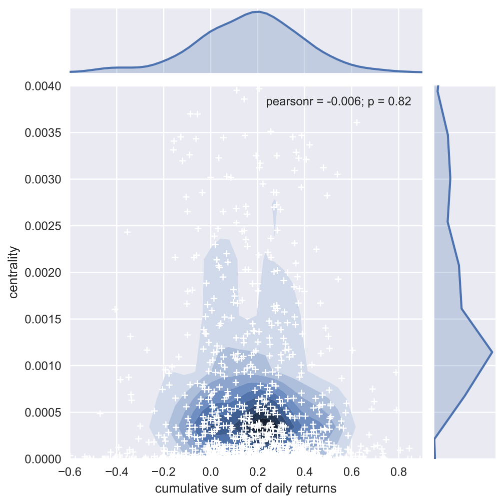
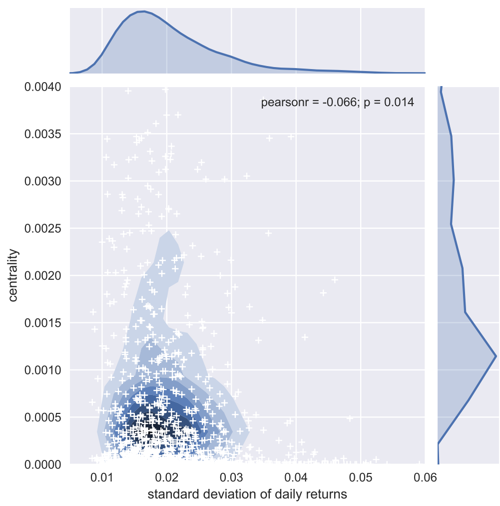


Figure 7.11: Stacked bar chart about the distribution of communities upon industrial sectors. Colours of stacks correspond to the colours of communities in figure 7.9 and figure 7.10, except the black stack indicating the nodes not belong to any communities. Sectors are arranged alphabetically.



(a) Bivariate distribution beween betweenness centralities of nodes and cumulative sums of stock daily return. The returns are actually logarithmic returns therefore the accumulation of all daily logarithmic returns in an entire year equals to a corresponding yearly logarithmic return.



(b) Bivariate distribution beween betweenness centralities of nodes and standard deviations of stock daily return.

Figure 7.12: Bivariate distributions with betweenness centrality

Like directed-unweighted stock network, and also conventional weighted stock networks, the strength distribution of directed-weighted stock network follows power-law distribution. The average strength and betweenness centrality of the two weighted networks are close to each other. But interestingly, the directed-weighted stock network has significantly high assortativity other than directed-unweighted and conventional weighted networks. It reveals that in the researched stock networks, nodes tend to be connected with other nodes with similar strength values rather than degree values, which means stocks tend to have relationships with other stocks with similar fluctuation in stock price return. Therefore, correlation coefficient is an important factor for the behaviour of node connections.

7.4.2 Analysis on the relationships between price return and betweenness centrality

Figures 7.12 reveal the relationships between betweenness centralities and price returns of stocks. First, in figure 7.12a as much more nodes have low betweenness centrality values (lower than 0.001), and the cumulative sum of returns fall intensively in the range of $(-0.1, 0.6)$, while that of the nodes with high betweenness centrality values (higher than 0.001) also fall evenly in the same range Therefore, there is no significant difference between the expected return for stocks with different betweenness centrality values.

Second, according to the figure 7.12b, in spite of several outliers, as the betweenness centrality of nodes becomes higher, there will be a higher possibility of nodes tend to have low standard deviation of stock daily return. This indicates the hubs in the network have considerably stable return during the specified researched year among the whole stock market. The average standard deviation for all values of betweenness centrality remain similar because of the more frequent occurrences of outliers with higher betweenness centralities from the figure.

As a result, although choosing stocks with high centralities possibly will not bring a higher expected return for a portfolio, they have the functionality of decreasing the overall risks and generating more stable returns, which is also a vital feature for stock investment.

Chapter 8

Other results

8.1 Introduction

Due to the complexity of financial market and the interconnectedness and interdependencies of industry sectors in the economy, the price returns of each coupling stocks might have certain underlying economic link. Such behaviours can hardly be explained by traditional financial models and theories. This project combines machine learning techniques, individual stock features, and empirical data of Industry Economic Accounts (IEAs) from Bureau of Economic Analysis (BEA) in the US to predict Granger causality of coupling US stocks. Limited Granger causalities are calculated as a small sample set compared to the target date set. A directed weighted complex network (DWCN) is constructed by considering companies as nodes, correlations of abnormal stock returns (alpha) as weights of links, and predicted Granger causalities indicate directions of links. The generated DWCN is visualised and its topological properties, stability, and effects on individual stocks and industries are researched in this paper. Suggestions towards financial market investment are provided based on the results of this research.

8.1.1 Motivation

Conducting Granger causality test between stock pairs is straightforward but not feasible due to the heavy-precondition and time-complexity in programme. Compared to the large order of magnitude of total stock pairs, manually calculated Granger causalities are too less to be used as training samples. Enlightened by the recent work by Jean et al. [3] which uses transfer learning and noisy proxy information performed very well at predicting poverties, demonstrating that machine learning techniques is powerful to be applied in a setting with limited training data, so an exploration towards the directed network of stock market is motivated in this project, combines machine learning techniques, transfer learning, individual stock features, and empirical data of Industry Economic Accounts (IEAs) from Bureau of Economic Analysis (BEA) in the US to predict Granger causality of coupling US stocks. Therefore, a directed weighted complex network is constructed by considering companies as nodes, correlations of abnormal stock returns (alpha) as weights of links, and predicted Granger causalities as the indications of directions of links.

8.1.2 Objectives

The goal of the attempt in this chapter is to reveal the Granger causality of price return series and utilise them into the topological analysis and visualisation of directed complex networks as so far no previous work has attempted to construct a directed network about stock markets. In addition, suggestions for stock market can be provided according to the results and findings.

8.1.3 Methodology

8.1.4 Technical challenges in the pre-processing for network construction

A preliminary correlation matrix is generated for the co-integration test of two price return time series. Bivariate Granger causality test is prerequisite-heavy and its time-complexity in programme is much more than that of linear correlation. This is the most important point to be solved, since the amount of all coupling stock pair is $n(n-1)/2$, while n is the number of stocks. There are over 3,000 company listings in NASDAQ, hence millions of times for Granger causality tests and pre-processes should run in programme in order to construct the directed network.

However, in this paper, machine learning technique is applied to predict the precedence relations i.e. predicted Granger causalities of every possible US stock pairs, based on a limited amount of actual Granger causalities calculated as training set. Fundamental indicators such as market capitalisation, P/B ratio, etc., plus public empirical data of Industry Economic Accounts (IEAs) from Bureau of Economic Analysis (BEA) [oEA18] in the US, are applied into the whole machine learning processes.

Stocks are divided into industry groups according to the summary level defined in the IEAs. Transfer learning is also applied and will be mentioned at Chapter 3.3. In addition, learning performance of each models are compared based on their predicting performance according to the ROC analysis.

8.1.5 Granger causality test

Granger causality test [Gra69] provides an asymmetrical measure for testing precedence relationship between two time series. The leitmotiv inside is that a time series can be described and analysed through a time-delayed auto-regressive model. Granger causality test tests whether the difference of a prediction to the time series from another time series through a multi-variate auto-regressive model is able to improve the prediction of the current behaviour of the time series, as the following forms illustrates:

$$x_t = \sum_{i=1}^{\infty} a_i x_{t-i} + c + \varepsilon_t \tag{8.1}$$

$$x_t = \sum_{i=1}^{\infty} a_i x_{t-i} + \sum_{j=1}^{\infty} b_j y_{t-j} + c' + \varepsilon'_t \tag{8.2}$$

Calculate the f-statistic using the following equation, the Granger causality is not significant if f-statistic is greater than the f-value:

$$F = \frac{(ESS_R - ESS_{UR})/q}{ESS_{UR}/(n-k)} \tag{8.3}$$

8.1.6 Transfer learning

Financial data of listed companies and fundamental economical data are both available in each stock market and government websites. Efforts have been taken upon the researches such as the work of Patel et al. [PSTK15], which applied machine learning techniques to predict stock price movement, but most of them use correlations between stock price or return series, such measures are unable to provide direction information for building a directed graph of stock market. Granger causality test is one suitable measure but the computation is overwhelmingly complex so that no researchers have ever implemented this.

This project has probed into the feasibility of applying machine learning techniques helping to predict Granger causality based on samples of Granger causalities that have been manually calculated. Here the word “predict” means estimation of some property that is not directly observed, rather than its common meaning of inferring something about the future. Unfortunately, over 3,000 listed companies yielding couples many orders of magnitude larger than the amount of sample data for human-beings can ever calculate. The scarcity of training data on these outcomes makes the application of machine learning techniques challenging.

This project overcome this challenge through a multi-step “transfer learning” approach [PY$^+$10], whereby a noisy but easily obtained proxy for sectoral association, the correlations of stock pairs, and fundamental indicators of listed companies are used to train a deep learning model. The model is then used to estimate Granger causalities based on very limited samples through a transferring process.

8.2 Brief results

In terms of future work, stock complex networks during a longer range of years can be generated and compared in together, the periods correspond to bull, bear, and stable market can be recognised and analysed separately and accordingly. New methods for determining the directions of edges to generate directed complex networks are expected to be proposed.

8.3 Considerations

Chapter 9

Conclusions and future work

9.1 Summary of results

This thesis studied the directed complex networks of US stock market in 2016. The directions and weights of edges are determined by the economical transaction relations and stock price correlation coefficients respectively. Overall, the characteristics of topology properties have not changed significantly from undirected weighted stock networks from other literatures which used correlation coefficients of stock prices as the weights of edges. However, from the new horizon, this paper is able to analyse on a higher dimensionality – topological property research and community detection with methods for directed networks which utilised the feature of edge directions. The resulting features of power-law and small-world for directed stock complex networks show continuity with the results in undirected stock complex networks researches. The study on community detection suggests ”livelihood” and ”production” are the most dominant and influential sectors in the stock market and the ”finance” sector has extremely strong internal connections. The partitioned communities are highly related with the economical activities among industries and indicate the potential cascading impact from a collapse of a specific firm or sector. The theoretical and practical contributions of aforementioned findings have been discussed.

9.2 Future work

In terms of future work, stock complex networks during a longer range of years can be generated and compared in together, the periods correspond to bull, bear, and stable market can be recognised and analysed separately and accordingly. New methods for determining the directions of edges to generate directed complex networks are expected to be proposed.

Bibliography

[AB02] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[BA99] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[BBP05] Vladimir Boginski, Sergiy Butenko, and Panos M Pardalos. Statistical analysis of financial networks. *Computational statistics & data analysis*, 48(2):431–443, 2005.

[Bri02] G Brida. High resolution frequency stability measurement system. *Review of scientific instruments*, 73(5):2171–2174, 2002.

[CK15] Shauhrat S Chopra and Vikas Khanna. Interconnectedness and interdependencies of critical infrastructures in the us economy: Implications for resilience. *Physica A: Statistical Mechanics and its Applications*, 436:865–877, 2015.

[CLL10] K Tse Chi, Jing Liu, and Francis CM Lau. A network perspective of the stock market. *Journal of Empirical Finance*, 17(4):659–667, 2010.

[CLSW15] Kun Chen, Peng Luo, Bianxia Sun, and Huaiqing Wang. Which stocks are profitable? a network method to investigate the effects of network structure on stock returns. *Physica A: Statistical Mechanics and its Applications*, 436:224 – 235, 2015.

[CR76] John C Cox and Stephen A Ross. The valuation of options for alternative stochastic processes. *Journal of financial economics*, 3(1-2):145–166, 1976.

[EMB02] Holger Ebel, Lutz-Ingo Mielsch, and Stefan Bornholdt. Scale-free topology of e-mail networks. *Physical review E*, 66(3):035103, 2002.

[ER59] Paul Erdős and Alfréd Rényi. On random graphs i. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[FF93] Eugene F Fama and Kenneth R French. Common risk factors in the returns on stocks and bonds. *Journal of financial economics*, 33(1):3–56, 1993.

[FF96] Eugene F Fama and Kenneth R French. Multifactor explanations of asset pricing anomalies. *The journal of finance*, 51(1):55–84, 1996.

[FFF99] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM computer communication review*, volume 29, pages 251–262. ACM, 1999.

[Fre77] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[Fre78] Linton C Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1978.

[Gra69] Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, pages 424–438, 1969.

[HSW17] Jake M Hofman, Amit Sharma, and Duncan J Watts. Prediction and explanation in social systems. *Science*, 355(6324):486–488, 2017.

[HZY09] Wei-Qiang Huang, Xin-Tian Zhuang, and Shuang Yao. A network analysis of the chinese stock market. *Physica A: Statistical Mechanics and its Applications*, 388(14):2956–2964, 2009.

[JJH$^+$03] Neil F Johnson, Paul Jefferies, Pak Ming Hui, et al. Financial market complexity. *OUP Catalogue*, 2003.

[Lin65] John Lintner. Security prices, risk, and maximal gains from diversification. *The journal of finance*, 20(4):587–615, 1965.

[LLH07] Kyoung Eun Lee, Jae Woo Lee, and Byoung Hee Hong. Complex networks in a stock market. *Computer Physics Communications*, 177(1-2):186, 2007.

[LM01] Vito Latora and Massimo Marchiori. Efficient behavior of small-world networks. *Physical review letters*, 87(19):198701, 2001.

[LN08] E. A. Leicht and M. E. J. Newman. Community structure in directed networks. *Phys. Rev. Lett.*, 100:118703, Mar 2008.

[Lon13] Yu Long. Visibility graph network analysis of gold price time series. *Physica A: Statistical Mechanics and its Applications*, 392(16):3374–3384, 2013.

[Mar52] Harry Markowitz. Portfolio selection. *The journal of finance*, 7(1):77–91, 1952.

[Mar56] Harry Markowitz. The optimization of a quadratic function subject to linear constraints. *Naval research logistics Quarterly*, 3(1-2):111–133, 1956.

[Mar59] Harry M Markowitz. Portfolio selection: Efficient diversification of investment. 344 p, 1959.

[Mer73] Robert C Merton. An intertemporal capital asset pricing model. *Econometrica: Journal of the Econometric Society*, pages 867–887, 1973.

[MM98] Richard O. Michaud and Robert O. Michaud. Efficient asset management: A practical guide to stock portfolio optimization and asset allocation. *Harvard Business School Press*, 1998.

[New02] Mark EJ Newman. Assortative mixing in networks. *Physical review letters*, 89(20):208701, 2002.

[NG04] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[NSRJ11] A Namaki, AH Shirazi, R Raei, and GR Jafari. Network analysis of a financial market based on genuine correlation and threshold method. *Physica A: Statistical Mechanics and its Applications*, 390(21-22):3835–3841, 2011.

[oEA18] U.S. Bureau of Economic Analysis. Use tables/before redefinitions/producer value. `https://www.bea.gov/industry/io_annual.htm`, 2018.

[PSTK15] Jigar Patel, Sahil Shah, Priyank Thakkar, and K Kotecha. Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, 42(1):259–268, 2015.

[PY$^+$10] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[SEC18] U.S. Securities and 2018. Exchange Commission, accessed April 12. Edgar — company filings. `http://www.sec.gov/edgar/searchedgar/companysearch.html`, 2018.

[Sha64] William F Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk. *The journal of finance*, 19(3):425–442, 1964.

[Str01] Steven H Strogatz. Exploring complex networks. *nature*, 410(6825):268, 2001.

[SW14] H Francis Song and Xiao-Jing Wang. Simple, distance-dependent formulation of the watts-strogatz model for directed and undirected small-world networks. *Physical Review E*, 90(6):062801, 2014.

[Tob58] James Tobin. Liquidity preference as behavior towards risk. *The review of economic studies*, 25(2):65–86, 1958.

[WS98] Duncan J Watts and Steven H Strogatz. Collective dynamics of ‘small-world’networks. *nature*, 393(6684):440, 1998.

Appendix A

Example of operation

The analytical code for this thesis on GitHub is available for readers to preview.

A.1 Example input and output

A.1.1 Input

```
from __future__ import division
import logging
import logging.config
import sys, csv, time, requests, statsmodels, math
from sklearn.linear_model import LinearRegression
from statsmodels.tsa.stattools import coint, adfuller, grangercausalitytests
import statsmodels.api as sm
from scipy import stats
import scipy.special
from scipy.stats import describe
from scipy.linalg import circulant
from contextlib import contextmanager
from datetime import datetime, timedelta
from dateutil.parser import parse
import collections
import random
import scipy.stats as ss
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
import matplotlib as mpl
import pylab
import powerlaw
import networkx as nx
from networkx.algorithms import community
import pandas as pd
import numpy as np
from core import SendEmail, ROOTPATH, sri_SP500_log_return, sri_SP500_close
from preprocess_stocks import lst_tickers_stp, LENTCKR, LENTRGL, df_codes_and_title, DICT_STP, getIndustryCodeByStockCode

sns.set(color_codes=True)
data_dir = ROOTPATH + r'/Codes'

# function for generating PGD
def genPureDedirectedGraph(theta_1, theta_2, mat_1, mat_2):
    G = nx.DiGraph()
    G.add_nodes_from(lst_tickers_stp)
    for i in lst_tickers_stp:
        #print(i, end=' ')
        matidx_i = EIO_industry_BEA_code_list.index(
            df_codes_and_title.loc[i, 'BEA'])
        for j in lst_tickers_stp:
            if i != j:
                matidx_j = EIO_industry_BEA_code_list.index(
```

```
                df_codes_and_title.loc[j, 'BEA'])
            a = mat_1[matidx_i, matidx_j]
            b = mat_2[matidx_i, matidx_j]
            if a > 0.0 and a > theta_1:
                G.add_edge(i, j)
            if b > 0.0 and b > theta_2:
                G.add_edge(j, i)
    return G

def genWDGraphFromPureDirectedGraph(PGD, return_list, threshold=-1):
    G = nx.DiGraph()
    G.add_nodes_from(lst_tickers_stp)
    for n, nbrs in PGD.adj.items():
        S1 = return_list[n]
        for nbr, eattr in nbrs.items():
            S2 = return_list[nbr]
            corr = S1.corr(S2)
            if corr >= threshold: G.add_edge(n, nbr, corr=corr)
    return G

# function for generating PCGD
# generate partial correlation directed graph
def genPartCorrGraph(UGD):
    G = nx.DiGraph()
    G.add_nodes_from(lst_tickers_stp)
    for n, nbrs in UGD.adj.items():
        for nbr, eattr in nbrs.items():
            G.add_edge(n, nbr, weight=eattr['corr'])
    return G

# function for generating ECGU
# generate entire correlation undirected graph
def genEntCorrGraph():
    G = nx.Graph()
    G.add_nodes_from(lst_tickers_stp)
    for i in range(LENTCKR):
        S1 = df_stock_abr[lst_tickers_stp[i]]
        for j in range(i+1, LENTCKR):
            S2 = df_stock_abr[lst_tickers_stp[j]]
            G.add_edge(n, nbr, weight=S1.corr(S2))
    return G

def rmvEdgeAttrOfGraph(WG):
    G = WG.copy()
    for n, nbrs in G.adj.items():
        for nbr, eattr in nbrs.items():
            if 'corr' in eattr: del eattr['corr']
    return G

def rmvIndepNodesFromGraph(WholeG):
    G = WholeG.copy()
    for n in WholeG.nodes():
        if WholeG.degree(n) == 0: G.remove_node(n)
    return G

def _distance_matrix(L):
    Dmax = L//2

    D = list(range(Dmax+1))
    D += D[-2+(L%2):0:-1]

    return circulant(D)/Dmax

def _pd(d, p0, beta):
    return beta*p0 + (d <= p0)*(1-beta)

def watts_strogatz(L, p0, beta, directed=False, rngseed=1):
    """
    Watts-Strogatz model of a small-world network

    This generates the full adjacency matrix, which is not a good way to store
    things if the network is sparse.

    Parameters
    ----------
    L        : int
               Number of nodes.

    p0       : float
               Edge density. If K is the average degree then p0 = K/(L-1).
               For directed networks "degree" means out- or in-degree.

    beta     : float
               "Rewiring probability."
```

```
    directed : bool
               Whether the network is directed or undirected.

    rngseed  : int
               Seed for the random number generator.

    Returns
    -------
    A        : (L, L) array
               Adjacency matrix of a WS (potentially) small-world network.

    """
    rng = np.random.RandomState(rngseed)

    d = _distance_matrix(L)
    p = _pd(d, p0, beta)

    if directed:
        A = 1*(rng.random_sample(p.shape) < p)
        np.fill_diagonal(A, 0)
    else:
        upper = np.triu_indices(L, 1)

        A          = np.zeros_like(p, dtype=int)
        A[upper]   = 1*(rng.rand(len(upper[0])) < p[upper])
        A.T[upper] = A[upper]

    return A

dct_title_amt = dict(df_codes_and_title['Title'].value_counts())
dct_BEA_amt = dict(df_codes_and_title['BEA'].value_counts())

def genEIODirectMatrix(directType, tradeoff = True, to_log = True):
    global dct_title_amt
    global EIO_industry_title_list
    industry_list = EIO_industry_title_list
    EIO_direct_matrix = np.matrix(
        np.zeros(
            (len(industry_list),
             len(industry_list)),
            dtype=np.float64))
    if directType is 'requirements':
        for i in range(len(industry_list)):
            for j in range(len(industry_list)):
                if EIO_matrix[i, j] > 0:
                    m = np.float64(EIO_matrix[i, j]) / EIO_matrix[-1, j]
                    if tradeoff and industry_list[i] in dct_title_amt:
                        m /= dct_title_amt[industry_list[i]]
                    EIO_direct_matrix[i, j] = logarise(m) if to_log else m
    elif directType is 'demands':
        for i in range(len(industry_list)):
            for j in range(len(industry_list)):
                if EIO_matrix[i, j] > 0:
                    m = np.float64(EIO_matrix[i, j]) / EIO_matrix[i, -1]
                    if tradeoff and industry_list[j] in dct_title_amt:
                        m /= dct_title_amt[industry_list[j]]
                    EIO_direct_matrix[i, j] = logarise(m) if to_log else m
    else: return None
    return EIO_direct_matrix

def getAllMatrixContent(mat):
    arr = []
    for m in mat:
        arr = np.append(arr, np.array(m)[0])
    return arr

def getNonzeroMatrixContent(mat):
    arr = []
    for m in mat:
        ar_m = np.array(m)[0]
        ar_m = ar_m[ar_m!=0]
        arr = np.append(arr, ar_m)
    return arr

def genEdgeDensity(lst, bins=100):
    theta_thresholds = np.linspace(np.floor(min(lst)*10.0)/10.0, np.ceil(max(lst)*10.0)/10.0, bins)
    edge_densities = []
    n = 0
    LENLST_FLOAT = np.float(len(lst))
    for theta in theta_thresholds:
        n += 1
        #print(n, end=' ')
        edge_densities.append(sum(corr >= theta for corr in lst)/LENLST_FLOAT)
```

```
    return theta_thresholds, edge_densities

def logarise(n): return 0.0 if n == 0 else -1.0/np.log10(n)

def combineThresholds(thresholds_1, thresholds_2, mat_1, mat_2):
    LENMAT = mat_1.shape[0]
    df = pd.DataFrame(
        index=range(len(thresholds_1)*len(thresholds_2)),
        columns=['theta_DR', 'theta_DD', 'no_directions'])
    idx = 0
    print(len(thresholds_1), end='')
    for t1 in thresholds_1:
        print('.', end='')
        exceeded = False
        for t2 in thresholds_2:
            cnt = 0
            if not exceeded:
                for i in range(LENMAT):
                    for j in range(LENMAT):
                        a = mat_1[i, j]
                        b = mat_2[i, j]
                        if (a > 0.0 and a > t1) or (b > 0.0 and b > t2):
                            cnt += 1
                if cnt == 0: exceeded = True
            df.iloc[idx,:] = [t1, t2, cnt]
            idx += 1
    return df

def combineThresholdsOfEIOAndCorrForAmtOfEdges(thresholds_eio, thresholds_corr, FG):
    global lst_tickers_stp
    df = pd.DataFrame(
        index=range(len(thresholds_eio)*len(thresholds_corr)),
        columns=['theta_EIO', 'theta_corr', 'no_edges'])
    idx = 0
    numrow = 0
    for t1 in thresholds_eio:
        print('[%s]' % numrow)
        exceeded = False
        for t2 in thresholds_corr:
            cnt = 0
            if not exceeded:
                for n, nbrs in FG.adj.items():
                    for nbr, eattr in nbrs.items():
                        if ('direct_requirement' in eattr and eattr['direct_requirement'] > t1) or ('direct_demand' in eattr and (eattr['direct_demand'] > t1)
                            if eattr['corr'] > t2: cnt += 1
                if cnt == 0: exceeded = True
            print(cnt, end=' ')
            df.iloc[idx,:] = [t1, t2, cnt]
            idx += 1
        numrow += 1
    return df

def combineThresholdsOfEIOAndCorrForIsWeaklyConnected(thresholds_eio, thresholds_corr, FG):
    global lst_tickers_stp
    df = pd.DataFrame(
        index=range(len(thresholds_eio)*len(thresholds_corr)),
        columns=['EIO', 'corr', 'is_weakly_connected'])
    idx = 0
    for t1 in thresholds_eio:
        for t2 in thresholds_corr:
            G = nx.DiGraph()
            G.add_nodes_from(lst_tickers_stp)
            for n, nbrs in FG.adj.items():
                for nbr, eattr in nbrs.items():
                    if ('direct_requirement' in eattr and eattr['direct_requirement'] > t1) or ('direct_demand' in eattr and (eattr['direct_demand'] > t1)):
                        if eattr['corr'] > t2: G.add_edge(n, nbr)
            G = rmvIndepNodesFromGraph(G)
            print(G.number_of_nodes(), end=' ')
            if G.number_of_nodes() > 0:
                is_weakly_c = nx.is_weakly_connected(G)
                print(is_weakly_c, end=' ')
                df.iloc[idx,:] = [t1, t2, is_weakly_c]
            else:
                print('Nill', end=' ')
                df.iloc[idx,:] = [t1, t2, False]
            idx += 1
    return df

def continueCombineThresholdsOfEIOAndCorrForIsWeaklyConnected(thresholds_eio, thresholds_corr, FG, start_point):
    global df
    i = 0
    idx = start_point
    cnt = 0
    for t1 in thresholds_eio:
```

```
        print('[%s]' % cnt)
        for t2 in thresholds_corr:
            if i < start_point:
                i += 1
                continue
            G = nx.DiGraph()
            G.add_nodes_from(lst_tickers_stp)
            for n, nbrs in FG.adj.items():
                for nbr, eattr in nbrs.items():
                    if ('direct_requirement' in eattr and eattr['direct_requirement'] > t1) or ('direct_demand' in eattr and (eattr['direct_demand'] > t1)):
                        if eattr['corr'] > t2: G.add_edge(n, nbr)
            G = rmvIndepNodesFromGraph(G)
            print(G.number_of_nodes(), end=' ')
            if G.number_of_nodes() > 0:
                is_weakly_c = nx.is_weakly_connected(G)
                print(is_weakly_c, end=' ')
                df.iloc[idx,:] = [t1, t2, is_weakly_c]
            else:
                print('Nill', end=' ')
                df.iloc[idx,:] = [t1, t2, False]
            idx += 1
        cnt += 1

FILE_EIO_2016 = ROOTPATH + '/Source/lxl/EIO_2016.csv'
EIO_matrix = np.matrix(np.genfromtxt(open(FILE_EIO_2016, 'rb'), delimiter=',', skip_header=2))
EIO_industry_BEA_code_list = list(pd.read_csv(FILE_EIO_2016, nrows=0).columns)[:-2]
EIO_industry_title_list = list(pd.read_csv(FILE_EIO_2016, skiprows=1).columns)[:-2]
FILE_STOCK_ABR = ROOTPATH + r'/Source/DF_STOCK_ABR.csv'
df_stock_abr = pd.read_csv(FILE_STOCK_ABR).set_index('Date')
df_stock_normal_return = pd.DataFrame(index=df_stock_abr.index, columns=df_stock_abr.columns)
for i in DICT_STP: df_stock_normal_return[i] = DICT_STP[i]['log_return']

EIO_direct_requirements_matrix = genEIODirectMatrix('requirements', tradeoff=True, to_log=True)
EIO_direct_demands_matrix = genEIODirectMatrix('demands', tradeoff=True, to_log=True)

ar_all_DR_Mat = getAllMatrixContent(EIO_direct_requirements_matrix)
ar_all_DD_Mat = getAllMatrixContent(EIO_direct_demands_matrix)

ar_all_DR_trans = [i for i in ar_all_DR_Mat]
ar_all_DD_trans = [i for i in ar_all_DD_Mat]

theta_thresholds_DR_all, edge_densities_DR_all = genEdgeDensity(ar_all_DR_trans, 100)
theta_thresholds_DD_all, edge_densities_DD_all = genEdgeDensity(ar_all_DD_trans, 100)

ar_nonzero_DR_Mat = getNonzeroMatrixContent(EIO_direct_requirements_matrix)
ar_nonzero_DD_Mat = getNonzeroMatrixContent(EIO_direct_demands_matrix)

ar_nonzero_DR_trans = [i for i in ar_nonzero_DR_Mat]
ar_nonzero_DD_trans = [i for i in ar_nonzero_DD_Mat]

theta_thresholds_DR, edge_densities_DR = genEdgeDensity(ar_nonzero_DR_trans, 100)
theta_thresholds_DD, edge_densities_DD = genEdgeDensity(ar_nonzero_DD_trans, 100)

b1 = np.append(1.0, edge_densities_DR_all)
b1[1] = b1[2]
b2 = np.append(0, theta_thresholds_DR_all)

x_dashline = 0.136
fig = plt.figure()
ax = fig.add_subplot(2,1,1)
ax.set_xlabel('threshold')
ax.set_ylabel('Transaction density')
ax.set_xlim(left=-0.05, right=1.2)
ax.set_title('Normalised Direct Requirement', fontsize='large')
dashed_line = Line2D([x_dashline, x_dashline], [-1.05, 1.05], linestyle = '--', linewidth = 1, color = [0.3,0.3,0.3], zorder = 1, transform = ax.transData)
ax.lines.append(dashed_line)
ax.plot(b2, b1, color='blue', lw=2)
ax = fig.add_subplot(2,1,2)
ax.set_xlabel('threshold')
ax.set_ylabel('Transaction density')
ax.set_title('Normalised Direct Demand', fontsize='large')
dashed_line = Line2D([x_dashline, x_dashline], [-0.05, 1.05], linestyle = '--', linewidth = 1, color = [0.3,0.3,0.3], zorder = 1, transform = ax.transData)
ax.lines.append(dashed_line)
ax.set_xlim(left=-0.05, right=1.2)
ax.plot(b2, b1, color='blue', lw=2)
fig.tight_layout()

df_combined_thresholds = combineThresholds(
    theta_thresholds_DR,
    theta_thresholds_DD,
    EIO_direct_requirements_matrix,
    EIO_direct_demands_matrix)

pt = df_combined_thresholds.pivot_table(index='theta_DR', columns='theta_DD', values='no_directions', aggfunc=np.sum)
```

```
f, ax = plt.subplots(figsize = (10, 4))
sns.heatmap(pt.iloc[:30,:20], cmap='rainbow', linewidths = 0.05, ax = ax)
ax.set_title('Amounts of directions per DR-threshold and DD-threshold')
ax.set_xlabel('theta_DD')
ax.set_ylabel('theta_DR')

def genFullGraph(stock_return_df):
    global lst_tickers_stp
    global EIO_industry_BEA_code_list
    global EIO_direct_requirements_matrix
    G = nx.DiGraph()
    G.add_nodes_from(lst_tickers_stp)
    for i in lst_tickers_stp:
        #print(i, end=' ')
        matidx_i = EIO_industry_BEA_code_list.index(df_codes_and_title.loc[i, 'BEA'])
        S1 = stock_return_df[i]
        for j in lst_tickers_stp:
            if i != j:
                matidx_j = EIO_industry_BEA_code_list.index(df_codes_and_title.loc[j, 'BEA'])
                if EIO_direct_requirements_matrix[matidx_i, matidx_j] > 0:
                    G.add_edge(i, j, direct_requirement = EIO_direct_requirements_matrix[matidx_i, matidx_j])
                if EIO_direct_demands_matrix[matidx_j, matidx_i] > 0:
                    G.add_edge(i, j, direct_demand = EIO_direct_demands_matrix[matidx_j, matidx_i])
                if G.has_edge(i, j): G.add_edge(i, j, corr=S1.corr(stock_return_df[j]))
    return G

FullG = genFullGraph(df_stock_normal_return)
direct_requirements_CN = []
direct_demands_CN = []
for n, nbrs in FullG.adj.items():
    for nbr, eattr in nbrs.items():
        if 'direct_requirement' in eattr.keys():
            direct_requirements_CN.append(eattr['direct_requirement'])
        if 'direct_demand' in eattr.keys():
            direct_demands_CN.append(eattr['direct_demand'])

corr_coef_CN = []
for n, nbrs in FullG.adj.items():
    for nbr, eattr in nbrs.items():
        corr_coef_CN.append(eattr['corr'])

plt.hist(corr_coef_CN, density=1, bins=260, histtype='bar')
plt.axis([-0.5, 1, 0, 3.2])
plt.xlabel('correlation')
plt.ylabel('p(correlation)')

describe(corr_coef_CN)

corr_mean = np.mean(corr_coef_CN)
corr_std = np.std(corr_coef_CN)
corr_coef_CN_00 = [(i - corr_mean)/corr_std for i in corr_coef_CN]

stats.probplot(corr_coef_CN, dist='norm', plot=pylab)
pylab.show()

sm.qqplot(np.array(corr_coef_CN_00), line='45',)
pylab.show()

ss.kstest(corr_coef_CN_00, 'norm')

theta_thresholds_corr, edge_densities_corr = genEdgeDensity(corr_coef_CN, 100)

fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('threshold')
ax.set_ylabel('Edge density')
ax.set_title('Correlation Coefficient', fontsize='large')
ax.plot(theta_thresholds_corr, edge_densities_corr, color='blue', lw=2)
fig.tight_layout()

recalc = False
npzfile_name = data_dir + '/pt_cteac_0719.npz'
pt_cteac = None
if recalc == True:
    df = pd.DataFrame(
        index=range(len(theta_thresholds_DR)*len(theta_thresholds_corr)),
        columns=['EIO', 'corr', 'is_weakly_connected'])
    continueCombineThresholdsOfEIOAndCorrForIsWeaklyConnected(
        theta_thresholds_DR, theta_thresholds_corr, FullG, 0)
    df_cteac = df.copy()
    pt_cteac = df_cteac.pivot_table(
        index = 'EIO', columns='corr',
        values = 'is_weakly_connected', aggfunc=np.sum)
    outfile = open(npzfile_name, 'wb')
```

```
    np.savez(outfile, ar_cteac=pt_cteac, col=pt_cteac.columns, ind=pt_cteac.index)
    outfile.close()
else:
    infile = open(npzfile_name, 'rb')
    npzfile = np.load(infile)
    infile.close()
    ar_cteac = npzfile['ar_cteac']
    pt_cteac = pd.DataFrame(ar_cteac)
    pt_cteac.columns = npzfile['col']
    pt_cteac.index = npzfile['ind']

f, ax = plt.subplots(figsize = (10, 4))
sns.heatmap(pt_cteac.iloc[5:50,20:90], cmap='rainbow', linewidths = 0.05, ax = ax)
ax.set_title('Amounts of directions per DR-threshold and DD-threshold')
ax.set_xlabel('corr')
ax.set_ylabel('EIO');

recalc = True
npzfile_name = data_dir + '/pt_toeacfaoe_0719.npz'
pt_toeacfaoe = None
if recalc == True:
    df_toeacfaoe = combineThresholdsOfEIOAndCorrForAmtOfEdges(
        theta_thresholds_DR, theta_thresholds_corr, FullG)
    pt_toeacfaoe = df_toeacfaoe.pivot_table(
        index = 'theta_EIO', columns='theta_corr',
        values = 'no_edges', aggfunc=np.sum)
    outfile = open(npzfile_name, 'wb')
    np.savez(outfile, ar_toeacfaoe=pt_toeacfaoe,
             col=pt_toeacfaoe.columns, ind=pt_toeacfaoe.index)
    outfile.close()
else:
    infile = open(npzfile_name, 'rb')
    npzfile = np.load(infile)
    infile.close()
    ar_toeacfaoe = npzfile['ar_toeacfaoe']
    pt_toeacfaoe = pd.DataFrame(ar_cteac)
    pt_toeacfaoe.columns = npzfile['col']
    pt_toeacfaoe.index = npzfile['ind']

pt_toeacfaoe.index = [round(i, 4) for i in pt_toeacfaoe.index]
pt_toeacfaoe.columns = [round(i, 4) for i in pt_toeacfaoe.columns]

f, ax = plt.subplots(figsize = (10, 4))
sns.heatmap(pt_toeacfaoe.iloc[:24,:81], cmap='gist_ncar', linewidths = 0.05, ax = ax)
ax.set_xlabel(r'$\theta_{corr}$')
ax.set_ylabel(r'$\theta_{EIO}$');

threshold_eio = 0.29225
threshold_corr = 0.378705
DiUnwtG = genPureDedirectedGraph(
    threshold_eio, #0.35
    threshold_eio, #0.35
    EIO_direct_requirements_matrix,
    EIO_direct_demands_matrix)

G = genWDGraphFromPureDirectedGraph(DiUnwtG, df_stock_normal_return, threshold_corr)
nonodes = G.number_of_nodes()
noedges = G.number_of_edges()
DiG_pureedge = rmvEdgeAttrOfGraph(G)
DiG_connected = rmvIndepNodesFromGraph(DiG_pureedge)

def genConventionalGraph(theta, return_list):
    global LENTCKR
    global lst_tickers_stp
    G = nx.Graph()
    G.add_nodes_from(lst_tickers_stp)
    for i in range(LENTCKR):
        T1 = lst_tickers_stp[i]
        S1 = return_list[T1]
        for j in range(i+1, LENTCKR):
            T2 = lst_tickers_stp[j]
            S2 = return_list[T2]
            corr = S1.corr(S2)
            if corr > theta: G.add_edge(T1, T2, corr=corr)
    return G

G_conv = genConventionalGraph(0.4983, df_stock_normal_return)
G_conv.number_of_edges()

data = [d for n, d in G.out_degree()]
plt.hist(data, density=1, bins=50, histtype='bar');
fit = powerlaw.Fit(data)
fit.distribution_compare('power_law', 'lognormal')
```

```
fig4 = fit.plot_ccdf(linewidth = 2)
fit.power_law.plot_ccdf(ax = fig4, color = 'r', linestyle = '--');
fig4.set_xlabel('centrality')
fig4.set_ylabel('$p(X\geq x)$')
fig4.legend(('CCDF','Power-law fit'))
set_size(4,4,fig4)

data = [d for n, d in G_rd.out_degree()]
plt.hist(data, density=1, bins=50, histtype='bar');

stats.probplot([d for n, d in G_rd.out_degree()], dist='norm', plot=pylab)

fig = sns.distplot([d for n, d in G_rd.out_degree()])
fig.set_xlabel('out-degree')
fig.set_ylabel('p(out-degree)')

data = [d for n, d in G_ws_mat.out_degree()]
plt.hist(data, density=1, bins=50, histtype='barstacked');

data = [d for n, d in G_ws_mat.out_degree() if d > 0]
powerlaw.plot_pdf(data, linear_bins = False, color = 'b');

fig = plt.figure()
ax = fig.add_subplot(1,1,1)
data = [d for n, d in G.out_degree() if d > 0]
powerlaw.plot_pdf(data, linear_bins = False, color = 'b')
ax.set_xlabel('out-degree')
ax.set_ylabel('p(out-degree)')
set_size(4,4,ax)

data = [d for n, d in G.in_degree() if d > 0]
powerlaw.plot_pdf(data, linear_bins = False, color = 'b');

data = [d for n, d in G.degree() if d > 0]
powerlaw.plot_pdf(data, linear_bins = False, color = 'b');

p0 = np.average([i[1] for i in G.out_degree()])/(nonodes-1)
ws_mat = watts_strogatz(L=nonodes, p0=p0, beta=0.5, directed=True)
G_ws_mat=nx.from_numpy_matrix(ws_mat, create_using=nx.DiGraph())
G_ws_mat.number_of_edges()

p = noedges / nonodes / (nonodes-1)
G_rd = nx.generators.gnp_random_graph(nonodes, p=p, directed=True)
G_rd.number_of_edges()

def calGlobalEfficiency(G, lst_nodes, N): #N = 0,
    #if N == 0: N = G.number_of_nodes()
    #if lst_nodes == None: lst_nodes = G.nodes()
    shortest_path = nx.shortest_path(G)
    acc = 0.0
    for i in lst_nodes:
        for j in lst_nodes:
            if i != j and (j in shortest_path[i]):
                acc += 1.0/(len(shortest_path[i][j])-1)
    return acc/N/(N-1)

def calLocalEfficiency(G):
    UndiG = G.to_undirected()
    lst_nodes = G.nodes()
    acc = 0.0
    for i in lst_nodes:
        print(i, end='')
        nodes_g = list(UndiG[i])
        n = len(nodes_g)
        if n > 0:
            #nodes_g.append(i)
            print('(%s)'%n, end=' ')
            acc += calGlobalEfficiency(G.subgraph(nodes_g), nodes_g, n)
    return acc/G.number_of_nodes()

calGlobalEfficiency(DiG_connected, G.number_of_nodes())
calGlobalEfficiency(G_ws_mat)
calLocalEfficiency(G)

def detectCommunityForDirectedGraph(G):
    lst_node = list(G.nodes)
    NONODES = G.number_of_nodes()
    NOEDGES = G.number_of_edges()
    NOINDEGREES = G.in_degree()
    NOOUTDEGREES = G.out_degree()
    overall_asgn = [(0, 0)] * NONODES

    def getNodeSpace(node_space, upd_asgn, val):
        return [node_space[i] for i in range(len(upd_asgn)) if upd_asgn[i] == val]
```

```
    ’’’
    def fineTune(node_space, upd_asgn):
        nonlocal G
        nonlocal overall_asgn
        nonlocal NONODES
        for i in upd_asgn:
            for j in node_space:
’’’
    def interateBisection(mod_mat, node_space, generation_mark):
        nonlocal G
        nonlocal overall_asgn
        upd_asgn = subdivideCommunities(mod_mat)
        # todo: fine-tune

        if len(np.unique(upd_asgn)) == 1: return
        delta_Q = calDeltaQ(upd_asgn, mod_mat)
        print(’calDeltaQ: %s’ % delta_Q)
        if delta_Q < 0: return
        node_space_1 = getNodeSpace(node_space, upd_asgn, -1)
        if len(node_space_1) == 0: return
        updCommunityAssignment(node_space_1, upd_asgn, generation_mark)
        print(’genGeneralisedModularityMatrix_1: %s:%s:%s’ % (time.localtime()[3],time.localtime()[4],time.localtime()[5]))
        mod_mat_1 = genGeneralisedModularityMatrix(node_space_1)
        interateBisection(mod_mat_1, node_space_1, generation_mark+1)
        node_space_2 = getNodeSpace(node_space, upd_asgn, 1)
        if len(node_space_2) == 0: return
        updCommunityAssignment(node_space_2, upd_asgn, generation_mark)
        print(’genGeneralisedModularityMatrix_2: %s:%s:%s’ % (time.localtime()[3],time.localtime()[4],time.localtime()[5]))
        mod_mat_2 = genGeneralisedModularityMatrix(node_space_2)
        interateBisection(mod_mat_2, node_space_2, generation_mark+1)
        return

    def genGeneralisedModularityMatrix(node_space):# Subgraph
        nonlocal G
        nonlocal lst_node
        nonlocal NOEDGES
        nonlocal NOINDEGREES
        nonlocal NOOUTDEGREES
        nonlocal overall_asgn
        LENNODESPECE = len(node_space)
        mod_mat = np.matrix(np.zeros((LENNODESPECE, LENNODESPECE), dtype=np.float64))
        for i in range(LENNODESPECE):
            print(i, end=’ ’)
            tckr_i = lst_node[node_space[i]]
            for j in range(LENNODESPECE):
                tckr_j = lst_node[node_space[j]]
                Bij = G.has_edge(tckr_j, tckr_i) - NOINDEGREES[tckr_i] * NOOUTDEGREES[tckr_j] / NOEDGES
                if overall_asgn[node_space[i]] == overall_asgn[node_space[j]]:
                    Ck = 0.0
                    for k in node_space:
                        tckr_k = lst_node[k]
                        Ck += G.has_edge(tckr_k, tckr_i) + G.has_edge(tckr_i, tckr_k) - (NOINDEGREES[tckr_i] * NOOUTDEGREES[tckr_k] + NOINDEGREES[tckr_k] * NOOUTDEGREES[tck
                    mod_mat[i, j] = Bij - Ck / 2.0
                else: mod_mat[i, j] = Bij
        print(’/’)
        return mod_mat

    def updCommunityAssignment(node_space, upd_asgn, generation_mark):
        nonlocal overall_asgn
        global asgn_history
        inreval_1 = 0
        inreval_2 = 0
        lst_gener_asgn = []
        for asgn in overall_asgn:
            if asgn[0] == generation_mark:
                lst_gener_asgn.append(asgn[1])
        for i in range(len(node_space)):
            if i not in lst_gener_asgn:
                inreval_1 = i
                break
        if upd_asgn.count(1) == 0: inreval_2 = inreval_1
        else:
            for i in range(len(node_space)):
                if i not in lst_gener_asgn:
                    inreval_2 = i
                    break
        for i in range(len(node_space)):
            if upd_asgn[i] == 1: overall_asgn[node_space[i]] = (generation_mark, inreval_1)
            if upd_asgn[i] == -1: overall_asgn[node_space[i]] = (generation_mark, inreval_2)
        #print(overall_asgn)
        asgn_history.append(overall_asgn.copy())

    def subdivideCommunities(mod_mat):
        sym_mat = mod_mat + mod_mat.T
```

```
        w, v = np.linalg.eigh(sym_mat)
        eigv = v[:, len(w)-1]
        return [np.sign(v.tolist()[0][0]) for v in eigv]

    def calDirectedGraphModularity(assignment):
        nonlocal G
        nonlocal lst_node
        nonlocal NONODES
        nonlocal NOEDGES
        nonlocal NOINDEGREES
        nonlocal NOOUTDEGREES
        Q = 0.0
        for i in range(NONODES):
            for j in range(NONODES):
                if assignment[i] == assignment[j]:
                    Q += G.has_edge(lst_node[j], lst_node[i]) - NOINDEGREES[lst_node[i]] * NOOUTDEGREES[lst_node[j]] / NOEDGES
        return Q / NOEDGES

    def calDeltaQ(upd_asgn, Bg):
        nonlocal NOEDGES
        sg = np.matrix(upd_asgn)
        return 0.25/NOEDGES*np.dot(np.dot(sg, (Bg+Bg.T)), sg.T)[0,0]

    MODMAT = np.matrix(np.zeros((NONODES, NONODES), dtype=np.float64))
    for i in range(NONODES):
        for j in range(NONODES):
            MODMAT[i, j] = G.has_edge(lst_node[j], lst_node[i]) - NOINDEGREES[lst_node[i]] * NOOUTDEGREES[lst_node[j]] / NOEDGES
    interateBisection(MODMAT, list(np.arange(NONODES)), 1)
    return overall_asgn, calDirectedGraphModularity(overall_asgn)

def calModularity(assignment):
    global G
    global lst_node
    global NONODES
    global NOEDGES
    global NOINDEGREES
    global NOOUTDEGREES
    Q = 0.0
    for i in range(NONODES):
        for j in range(NONODES):
            if assignment[i] == assignment[j]:
                Q += G.has_edge(lst_node[j], lst_node[i]) - NOINDEGREES[lst_node[i]] * NOOUTDEGREES[lst_node[j]] / NOEDGES
    return Q / NOEDGES

lst_node = list(G.nodes())
NONODES = G.number_of_nodes()
NOEDGES = G.number_of_edges()
NOINDEGREES = G.in_degree()
NOOUTDEGREES = G.out_degree()

over_best_asgn = [0] * len(lst_tickers_stp)
for i in range(len(lst_tickers_stp)):
    over_best_asgn[i] = int(G.node[lst_tickers_stp[i]]['community'])

best_asgn = over_best_asgn.copy()
origin_mod = calModularity(best_asgn)
for i in range(len(best_asgn)):
    asgn = best_asgn[i]
    for uniq in uniq_asgn:
        if asgn != uniq:
            best_asgn[i] = uniq
            new_mod = calModularity(best_asgn)
            if new_mod > origin_mod:
                origin_mod = new_mod
                asgn = uniq
            else: best_asgn[i] = asgn

lst_tckr_nonzero = [i[0] for i in G.degree if i[1]>0]
rdmchosen_tickers = np.random.choice(lst_tckr_nonzero, 1, replace=False)
nonzerodeg_subG = G.subgraph(lst_tckr_nonzero).copy()

asgn_history = []
overall_asgn, modularity = detectCommunityForDirectedGraph(nonzerodeg_subG)

stp_group_tckr_index = sri_overall_asgn[sri_overall_asgn == v_c.index[0]].index.tolist()
stp_group_tckr = [lst_tickers_stp[i] for i in stp_group_tckr_index]
stp_group_indcode = getIndustryCodeByStockCode(stp_group_tckr, code_type='Title')
sri_community_1 = pd.Series(stp_group_indcode).value_counts()
stp_group_tckr_index = sri_overall_asgn[sri_overall_asgn == v_c.index[1]].index.tolist()
stp_group_tckr = [lst_tickers_stp[i] for i in stp_group_tckr_index]
stp_group_indcode = getIndustryCodeByStockCode(stp_group_tckr, code_type='Title')
sri_community_2 = pd.Series(stp_group_indcode).value_counts()
stp_group_tckr_index = sri_overall_asgn[sri_overall_asgn == v_c.index[2]].index.tolist()
stp_group_tckr = [lst_tickers_stp[i] for i in stp_group_tckr_index]
```

```
stp_group_indcode = getIndustryCodeByStockCode(stp_group_tckr, code_type=’Title’)
sri_community_3 = pd.Series(stp_group_indcode).value_counts()
stp_group_tckr_index = sri_overall_asgn[sri_overall_asgn == v_c.index[3]].index.tolist()
stp_group_tckr = [lst_tickers_stp[i] for i in stp_group_tckr_index]
stp_group_indcode = getIndustryCodeByStockCode(stp_group_tckr, code_type=’Title’)
sri_community_4 = pd.Series(stp_group_indcode).value_counts()
stp_group_tckr_index = sri_overall_asgn[sri_overall_asgn == v_c.index[4]].index.tolist()
stp_group_tckr = [lst_tickers_stp[i] for i in stp_group_tckr_index]
stp_group_indcode = getIndustryCodeByStockCode(stp_group_tckr, code_type=’Title’)
sri_community_5 = pd.Series(stp_group_indcode).value_counts()

stp_group_tckr_index = []
for i in range(5,11): stp_group_tckr_index += sri_overall_asgn[sri_overall_asgn == v_c.index[i]].index.tolist()
stp_group_tckr = [lst_tickers_stp[i] for i in stp_group_tckr_index]
stp_group_indcode = getIndustryCodeByStockCode(stp_group_tckr, code_type=’Title’)
sri_community_6 = pd.Series(stp_group_indcode).value_counts()

uniq_ind = np.unique(df_codes_and_title.Title)

lst_community_1 = []
lst_community_2 = []
lst_community_3 = []
lst_community_4 = []
lst_community_5 = []
lst_community_6 = []
for ind in uniq_ind:
    if ind in sri_community_1.index: lst_community_1.append(sri_community_1[ind]) #/ sum(sri_overall_asgn == v_c.index[0]))
    else: lst_community_1.append(0)
    if ind in sri_community_2.index: lst_community_2.append(sri_community_2[ind]) #/ sum(sri_overall_asgn == v_c.index[1]))
    else: lst_community_2.append(0)
    if ind in sri_community_3.index: lst_community_3.append(sri_community_3[ind]) #/ sum(sri_overall_asgn == v_c.index[2]))
    else: lst_community_3.append(0)
    if ind in sri_community_4.index: lst_community_4.append(sri_community_4[ind]) #/ sum(sri_overall_asgn == v_c.index[3]))
    else: lst_community_4.append(0)
    if ind in sri_community_5.index: lst_community_5.append(sri_community_5[ind]) #/ sum(sri_overall_asgn == v_c.index[4]))
    else: lst_community_5.append(0)
    if ind in sri_community_6.index: lst_community_6.append(sri_community_6[ind]) #/ sum(sri_overall_asgn == v_c.index[5]))
    else: lst_community_6.append(0)

lst2_sector = [None] * len(uniq_ind)
for i in range(len(uniq_ind)):
    lst2_sector[i] = []
    if uniq_ind[i] in sri_community_1.index: lst2_sector[i].append(sri_community_1[uniq_ind[i]])
    else: lst2_sector[i].append(0)
    if uniq_ind[i] in sri_community_2.index: lst2_sector[i].append(sri_community_2[uniq_ind[i]])
    else: lst2_sector[i].append(0)
    if uniq_ind[i] in sri_community_3.index: lst2_sector[i].append(sri_community_3[uniq_ind[i]])
    else: lst2_sector[i].append(0)
    if uniq_ind[i] in sri_community_4.index: lst2_sector[i].append(sri_community_4[uniq_ind[i]])
    else: lst2_sector[i].append(0)
    if uniq_ind[i] in sri_community_5.index: lst2_sector[i].append(sri_community_5[uniq_ind[i]])
    else: lst2_sector[i].append(0)
    if uniq_ind[i] in sri_community_6.index: lst2_sector[i].append(sri_community_6[uniq_ind[i]])
    else: lst2_sector[i].append(0)

idx = np.arange(6)
plt.figure(figsize=(15,15))

colors = list(dict(mpl.colors.BASE_COLORS, **mpl.colors.CSS4_COLORS).values())
idx = np.arange(6)
plt.figure(figsize=(10,10))
p = []
for i in range(len(uniq_ind)):
    bottom = [0] * 6
    for j in range(i): bottom = [a+b for a, b in zip(bottom, np.array(lst2_sector[j]))]
    p.append(plt.bar(idx, lst2_sector[i], bottom=bottom, color=colors[np.random.randint(len(colors))]))
plt.xticks(idx, [’C1’,’C2’,’C3’,’C4’,’C5’,’Others’])

idx = np.arange(len(uniq_ind))
plt.figure(figsize=(15,15))
p1 = plt.bar(idx, lst_community_1, color=(0.85, 0.5176, 1))
p2 = plt.bar(idx, lst_community_2, bottom=lst_community_1, color=(0.553, 0.753, 0.0863))
p3 = plt.bar(idx, lst_community_3,
             bottom=np.array(lst_community_1)+np.array(lst_community_2), color=(0.4157, 0.7608, 1))
p4 = plt.bar(idx, lst_community_4,
             bottom=np.array(lst_community_1)+np.array(lst_community_2)+np.array(lst_community_3), color=[0.937255,0.851,0.25333])
p5 = plt.bar(idx, lst_community_5,
             bottom=np.array(lst_community_1)+np.array(lst_community_2)+np.array(lst_community_3)+np.array(lst_community_4), color=[0.898,0.5294,0.1294])
p6 = plt.bar(idx, lst_community_6,
             bottom=np.array(lst_community_1)+np.array(lst_community_2)+np.array(lst_community_3)+np.array(lst_community_4)+np.array(lst_community_5), color=[0.283,0.2823,0
plt.xticks(idx, uniq_ind, rotation=90)
plt.legend((p1[0], p2[0], p3[0], p4[0], p5[0], p6[0]), (’Community 1’, ’Community 2’, ’Community 3’, ’Community 4’, ’Community 5’, ’Unclustered’));
```

A.1.2 Output

```
Hello World!
```

A.1.3 Another way to include code

You can also use the capabilities of the `listings` package to include sections of code, it does some keyword highlighting.

```c
/* Hello world program */

#include <stdio.h>

int main(void)
{
    printf("Hello World!\n") ;
    return 0 ;
}
```