Imperial College London

Final Year Project Report

Community Detection in Networks

Author:
Hesam Ipakchi

Supervisor:
Dr. Moez Draief
Second marker:
Dr. Jeremy Pitt

This report is submitted in fulfilment of the requirements
for the degree of *MEng Electronic and Information Engineering*
in the
Department of Electrical and Electronic Engineering
Imperial College London

June 2014

Abstract

A very important area in network science is the detection of communities within networks, which helps to boost our understanding of many real-world complex systems. Community detection is a very challenging problem, and a range of algorithms have been proposed, many based upon different approaches and techniques. This project's aims are two-fold. Firstly, we investigate and study different community detection algorithms, with a synthetic data-driven testing approach considered to develop a comprehensive comparison based on a range of network properties, using well-known statistical generative models. Secondly, we fuse traditional community detection algorithms with analysis of correlation structure within the financial networks setting to investigate modified or tailored community detection algorithms that are applied to real-world and time-evolving financial data constructed from the daily prices of 80 stocks listed on the FTSE 100, during a period between 2004 and 2013. The combination of our analysis using both synthetic and empirical data enables us to advocate the use of specific algorithms based upon the underlying network properties one wishes to detect communities from, in addition to validating techniques which may become useful for investors in their risk management of portfolios. Furthermore, we are able to uncover interesting patterns in the evolving correlation structure of financial assets, which constitute a major equities exchange, during a significant period for economies worldwide.

Acknowledgements

I would like to thank my supervisor Dr. Moez Draief for his support and advice during the project work in addition to his inspiring knowledge that has guided me throughout.

I would also like to thank my great friends and fellow colleagues who have made my studies the fantastic experience it has been.

Finally, I would like to thank my family for their everlasting support and love.

Contents

Abstract ii

Acknowledgements iii

Notation ix

1 Introduction 1

2 Background 3
- 2.1 Graph Theory Preliminaries . . . 3
- 2.2 Community Structure in Networks . . . 6
  - 2.2.1 Planted Partition Model . . . 8
  - 2.2.2 Hidden Clique Model . . . 10
- 2.3 Financial Networks . . . 14
  - 2.3.1 Prices and Returns of Financial Assets . . . 14
  - 2.3.2 Mean-Variance Portfolio Theory . . . 16
  - 2.3.3 Constructing Financial Networks . . . 18

3 Community Detection Algorithms 21
- 3.1 Spectral Clustering . . . 21
- 3.2 Modularity-based Optimisation . . . 23
  - 3.2.1 Greedy Algorithm . . . 24
  - 3.2.2 Simulated Annealing . . . 24
  - 3.2.3 Extremal Optimisation . . . 25
  - 3.2.4 Spectral Algorithm . . . 26
- 3.3 Belief Propagation Algorithm . . . 28
- 3.4 NLPI and AMP Algorithms . . . 32
  - 3.4.1 Non-linear Power Iteration . . . 32
  - 3.4.2 Approximate Message Passing . . . 33

4 Experiments on Synthetic Data 35
- 4.1 Spectral Clustering and Modularity-Optimisation Algorithms . . . 35
- 4.2 Belief Propagation Algorithm . . . 40
- 4.3 NLPI and AMP Algorithms . . . 43

5 Community Detection in Financial Networks **47**
5.1 Constructing the Real-world Financial Network 47
5.1.1 Random Matrix Theory 48
5.2 Community Detection Algorithms 50
5.2.1 Modularity Optimisation Methods 51
5.2.2 Modified Modularity Optimisation Methods 53
5.3 Synthetic Data Testing 55
5.4 Application to Real-world Financial Network 55

6 Temporal Evolution of Financial Networks **60**
6.1 Constructing Time-evolving Financial Networks 61
6.2 Temporal Evolution of Asset Correlations 62
6.3 Generalised Modularity Optimisation 66
6.3.1 Laplacian Dynamics and Stability 66
6.3.2 Stability for Dynamic Networks 68
6.3.3 Generalised Louvain Method 70
6.4 Synthetic Data Testing 70
6.5 Community Detection in Real-world Time-evolving Financial Networks . . 73

7 Conclusion **74**

A List of Stocks **75**

Bibliography **78**

List of Figures

2.1 Visualisations of example undirected graphs. 5
2.2 Plots of adjacency matrices of graph generated by planted partition model. 9
2.3 Visualisation of a graph generated by the planted partition model. 9
2.4 Plots of adjacency matrices of graph generated by hidden clique model. . 11
2.5 Visualisation of a graph generated by the hidden clique model. 11
2.6 Plots illustrating spectral phase transition of Wigner Matrices. 13
2.7 Example plot for price and logarithmic return 16
2.8 Example plot for a correlation matrix 20

3.1 Visualisation of spectral clustering embedding. 22
3.2 Illustration of greedy algorithm for modularity optimisation. 25

4.1 Plot of overlap for spectral clustering and modularity methods in the dense regime. 37
4.2 Plot of overlap for spectral clustering and modularity methods in the sparse regime. 38
4.3 Visualisation of spectral clustering embedding in sparse and dense regimes. 38
4.4 Plot of overlap for belief propagation algorithm on small-sized networks with three different levels of sparsity. 41
4.5 Plot of overlap for belief propagation algorithm on small-sized networks with three different levels of sparsity and four ground-truth communities. 42
4.6 Plot of overlap for belief propagation algorithm on large network for two and four ground-truth communities. 43
4.7 Plot of accuracy for NLPI algorithm. 44
4.8 Plot of accuracy for AMP algorithm. 45

5.1 Plot of expected return against volatility for 80 FTSE 100 stocks. 48
5.2 Plots of empirical and RMT predicted eigenvalue spectrum 50
5.3 Plots of synthetically generated benchmark correlation matrices 56
5.4 Communities of the FTSE 100 data generated using four different algorithms. 57
5.5 Pie charts showing the relative composition of each generated community based on industry sectors of the FTSE 100 stocks for four different algorithms. 58
5.6 Plots of renormalised filtered correlation matrices for the communities generated by four different algorithms. 59

6.1 Plots characterising the distribution of correlation coefficients for a fixed roll-over period of 1 day and varying window lengths. 62
6.2 Plots characterising the distribution of correlation coefficients for a fixed roll-over period of 10 days and varying window lengths. 62
6.3 Plots of the contribution of the top 5 principal components to the total variance in returns against time. 65
6.4 Illustration of an example multislice network with three slices. 69
6.5 Plots of the normalised variation of information obtained by applying the generalised Louvain method with different parameters. 73

List of Tables

5.1 Colour representation for 10 industry sectors used to classify FTSE 100 stocks, to be used as a legend. 57

6.1 Modularities obtained for the generalised Louvain method with different values of ω and common parameter $\gamma = 1$ obtained from applying the methods to synthetic data generated with different SNR values. 71

A.1 List of FTSE 100 Stocks studied 75

Notation

$\lvert\mathcal{S}\rvert$	Cardinality of the set $\mathcal{S}$
$\mathbb{1}_{\mathcal{S}}$	Indicator variable over the set $\mathcal{S}$
y	Scalar y
$\lvert y\rvert$	Absolute value of y
$\boldsymbol{v}$	Vector $\boldsymbol{v}$
$\lVert\boldsymbol{v}\rVert$	Euclidean norm of the vector $\boldsymbol{v}$
$\mathbf{M}$	Matrix $\mathbf{M}$
M_{ij}	The element of the matrix $\mathbf{M}$ at row i and column j
$\mathbf{M}^T$	Transpose of the matrix $\mathbf{M}$
$\lvert\mathbf{M}\rvert$	Determinant of the matrix $\mathbf{M}$
$tr(\mathbf{M})$	Trace of the matrix $\mathbf{M}$
$\mathbb{E}(X)$	Expected value of X
$Var(X)$	Variance of X
$\exp(x)$	Exponential function
$\log(x)$	*natural logarithm* of x (logarithm to the base e)

Chapter 1

Introduction

Networks have been studied extensively to model many interesting complex systems, including the Internet, social networks, financial networks and biological networks [18, 54, 57]. Any network consists of *nodes* which represent items of interest, and *edges* which represent the connectivity between pairs of nodes. For example, considering social networks, nodes are the users and the edges correspond to interactions between the users. An interesting feature many networks exhibit is *community structure*, which involves the natural dividing of nodes into groups, called *communities*, where there are denser connections within a group, and sparser connections between different groups [18, 19, 29, 54]. This particular type of community structure is also known as *assortative* [54]. For instance, social networks contain communities corresponding to real-life communities consisting of the members, such as friendship or family circles. The problem of detecting communities within networks is known as *community detection*, and a current research area involves developing algorithms to partition a netowrk into communities.

In order to provide a theoretical setting to test and compare different community detection algorithms, generative models of random graphs are very useful, and one such commonly used model is the *stochastic block model* [45, 54]. These statistical models can provide synthetic data to capture varying network properties (e.g. size, sparsity, number of communities) that represent a variety of test case scenarios. We will investigate a selection of community detection algorithms, which we can distinguish based upon their differing approaches, and then apply them to the synthetic data generated from an appropriate block model to measure their performance. This process enables us to advocate specific techniques depending on the types of networks considered.

The underlying ingredients of the community detection algorithms have other interesting applications also, including the analysis of time series data within the context of financial networks. The goal involves seeking isolated groups of correlated financial assets to be used in *mean variance portfolio optimisation*. This will provide investors with 'baskets' of assets to be used as a first point of call for the selection of assets to make up their portfolio based upon their unique risk and return preferences, which is very useful for the purposes of risk management [57]. We consider the application of community detection algorithms to real-world time-evolving financial networks in order determine isolated groups of correlated stocks found on the FTSE 100 exchange.

This project's principal aims can be summarised as follows. Firstly, we investigate and study different community detection algorithms, with a data-driven testing approach considered. Secondly, we combine traditional community detection algorithms with analysis of the financial network setting to investigate modified or tailored community detection algorithms which we intend to apply to real-world and time-evolving financial data based on the prices of 80 FTSE 100 stocks during a period between 2004 and 2013.

Our contribution lies in the study, testing and comparison of a range of existing community detection algorithms in addition to the adaptation of a class of algorithms to the time-evolving financial networks setting, constructed from a recent 10-year period of financial market data. This enables us to identify important observation and draw conclusions regarding the evolving correlation structure of the assets which are traded on the FTSE 100 exchange, over the last decade.

The rest of the report is organised as follows. In chapter 2, we provide a technical background in graph theory, community structure within networks and financial networks required to understand the concepts investigated throughout the report. In chapter 3, several community detection algorithms, available in the literature, are introduced and explained. Chapter 4 details our experiments of several community detection algorithms on synthetic data and provides a summary of conclusions drawn in comparing the algorithms. In chapter 5, we motivate modified community detection algorithms for financial networks, and apply them to real-world data in order to investigate their performance. In chapter 6, we consider the temporal evolution of correlations in financial networks using community dynamics to uncover the changes in the structure of a financial market over a recent time period. We make concluding remarks in chapter 7.

Chapter 2

Background

In this chapter we will describe all the technical background required to understand and detail the different settings we investigate. Initially, we will highlight some basic results in graph theory. Then, we will outline the problem of community detection and present statistical models used to generate random graphs with community structure to be used as a testing playground for algorithms. Following this, we will discuss basic concepts within finance required to understand the behaviour of financial assets that will provide the motivation for applying community detection algorithms to financial networks.

2.1 Graph Theory Preliminaries

We assume the reader is familiar with some basic concepts in linear algebra such as matrix multiplication, eigenvectors and eigenvalues of matrices. Rather, we will cover some basic tools within spectral graph theory using definitions from [18, 21, 29, 44]. Spectral graph theory is the study of graphs through the eigenvalues and eigenvectors of matrices associated with the graphs [44]. We begin by defining some basic notions about graphs.

Definition 2.1. A *graph* $\mathcal{G}$ is a pair of sets (V,E), where V is a set of *vertices* or *nodes* and $E \subset V^2$, the set of unordered pairs of elements of V. The elements of E are called *edges* or *links*.

Definition 2.2. A graph $\mathcal{G} = (V, E)$ is called *undirected* if for all $v, w \in V$: $(v, w) \in E \iff (w, v) \in E$. Otherwise, $\mathcal{G}$ is called *directed*.

Definition 2.3. A *weighted* graph is a graph where a number (weight) is assigned to each edge.

We will assume, without loss of generality, that $V = \{1, \ldots, n\}$. See figure 2.1a for an example of an undirected graph with seven vertices and eleven edges.

Definition 2.4. A graph $\mathcal{G}' = (V', E')$ is a *subgraph* of $\mathcal{G} = (V, E)$ if $V' \subset V$ and $E' \subset E$. If $\mathcal{G}'$ contains all edges of $\mathcal{G}$ that join vertices of V', one says that the subgraph $\mathcal{G}'$ is *induced* or *spanned* by V'.

Definition 2.5. A partition of the vertex set V in two subsets S and $V - S$ is called a *cut*. The *cut size* is the number of edges of $\mathcal{G}$ joining vertices of S with vertices of $V - S$.

Definition 2.6. Two vertices are *adjacent* or *neighbours* if they are connected by an edge. The set of neighbours of a vertex v is called *neighbourhood*, and denoted by $\Gamma(v)$.

Definition 2.7. The *degree* d_v of a vertex v is the number of its neighbours, $|\Gamma(v)|$.

We will be interested in using certain graphs in the models, such as bipartite graphs.

Definition 2.8. A *bipartite* graph, is a graph whose vertices can be decomposed into two disjoint sets such that no two vertices within the same set are adjacent.

Definition 2.9. A *clique* of an undirected graph is a subset of its vertices such that every pair of vertices in the subset are adjacent in the graph.

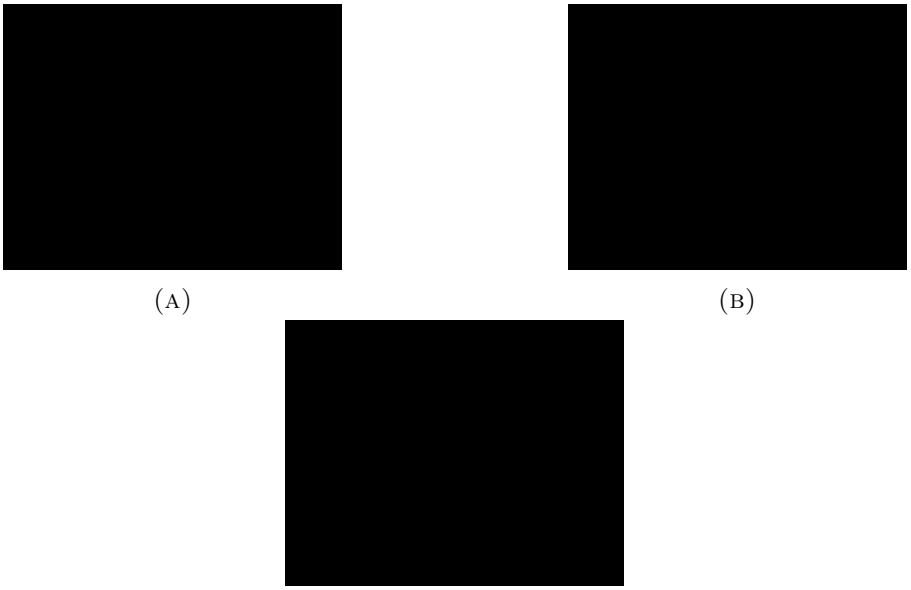
An example of an undirected bipartite graph with nine vertices and eight edges is shown in figure 2.1b, whilst an example of a clique within an undirected graph is shown in figure 2.1c.

There is a very close connection between graphs and matrices, since the whole information about the topology of a graph can be entailed in matrix form.

Definition 2.10. The *adjacency matrix*, $\mathbf{A} \in \{0, 1\}^{n \times n}$ of a graph $\mathcal{G} = (V, E)$, is a $n \times n$ matrix whose element A_{ij} equals 1 if there exists an edge joining vertices i and j in $\mathcal{G}$, and zero otherwise.

From definition 2.10, it follows that elements of the adjacency matrix, $\mathbf{A}$, can be written as

$$A_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$



(A) (B)

(C)

FIGURE 2.1: A set of visualisations of undirected graphs. In (a), the graph has seven nodes and eleven edges. In (b), a bipartite graph, with nine nodes (elements of disjoint sets are coloured black and red denoting membership) and eight edges, is shown. In (c), an undirected graph, with six nodes and six edges is shown, where the nodes coloured red form a clique within the graph.

Note that the sum of elements of the i-th row of the adjacency matrix yields the degree of node i of the graph, $d_i = \sum_j A_{ij}$. Also, the adjacency matrix is symmetric if the graph is undirected.

Definition 2.11. The *weighted adjacency matrix*, $\mathbf{A} \in \mathbb{R}^{n \times n}$ of a weighted graph $\mathcal{G} = (V, E)$, is a $n \times n$ matrix whose element A_{ij} equals the weight of the edge connecting nodes i and j, if it exists, and zero otherwise.

There are other matrices that have also been studied extensively in spectral graph theory, including the Laplacian which is applied in topics such as graph partitioning, synchronisation and graph connectivity [29].

Definition 2.12. The *degree matrix*, $\mathbf{D}$, of a graph $\mathcal{G} = (V, E)$, is a $n \times n$ diagonal matrix whose element D_{ii} equals the degree of vertex i.

From definition 2.12, it follows that elements of the degree matrix, $\mathbf{D}$, can be written as

$$D_{ij} = \begin{cases} d_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

Definition 2.13. The matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is called the *unnormalised Laplacian matrix*.

From definition 2.13, it follows that elements of the unnormalised Laplacian matrix of a graph $\mathcal{G} = (V, E)$, $\mathbf{L}$, can be written as

$$L_{ij} = \begin{cases} d_i & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

Definition 2.14. The matrix $\widetilde{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ is called the *normalised Laplacian matrix*, where $\mathbf{I}$ is the $n \times n$ identity matrix.

Note that from definitions 2.13 and 2.14, the normalised Laplacian matrix can also be written as $\widetilde{\mathbf{L}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$.

An important property of matrices is their spectra, which we shall analyse for certain matrices later in the report to motivate community detection algorithms.

Definition 2.15. The *spectrum* of a graph $\mathcal{G}$ is the set of eigenvalues of its adjacency matrix, $\{\lambda_1, \ldots, \lambda_n\}$.

Definition 2.16. Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of a matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$. The *spectral radius* is defined as $\rho(\mathbf{M}) = \max_i |\lambda_i|$.

2.2 Community Structure in Networks

An intuitive notion of communities within graphs involves the assignment of nodes to communities such that there are denser connections between nodes belonging to the same community, and sparser connections between nodes belonging to different communities. If a graph exhibits this property, it is said to contain assortative community structure [18, 19, 29, 54]. For instance, within social networks, where nodes are users and edges between nodes represent interactions between the users, community structure within the graph corresponds to real-life communities consisting of the users and friendship circles. The aim of community detection algorithms is to estimate or recover the node assignments. The algorithms need to be efficient due to the moderate size of graphs realised in many real-world applications, so we require the computationally complexity to not be worse than nearly linear in the number of edges in the graph (approximately $O\left(n^2 \log(n)\right)$

where n represents the number of nodes in the graph). It is crucial to note that we are seeking to identify communities in graphs of a moderate size, and not very large scale graphs as are now increasingly studied in research for social networks applications (e.g. Facebook, Twitter, Google networks). The reason for this decision lies in the very different approaches taken by researches for large scale applications than for small or moderate size networks.

Within the literature, the terms *groups* and *clusters* are synonymous with communities, and as such we will use all three terms interchangeably through the report; so the reader should note all these terms refer to the same notion of communities in graphs. Moreover we also refer to the process of estimating node assignments as *partitioning* the network.

In order to help provide a setting where different algorithms may be compared, we wish to study particular models which generate random graphs. One popular model is called the stochastic block model. Many special cases of this model have been studied, but we consider two versions, both widely used in the literature. Firstly, there is a model considered by Decelle et al. [54] and Nadakuditi et al. [45], also known as the *planted partition model*. Secondly, there is a model used by Montanari [58, 61], which we will refer to as the *hidden clique model*. We emphasise that we do not exclusively focus on detecting cliques for the latter model, but the name is simply convenient for reference in this report.

Let us define the stochastic block model following Decelle et al. [54]. The stochastic block model has parameters: k, n_a, $\mathbf{P}$. k represents the number of communities (or groups), n_a refers to the expected fraction of nodes within each group a, for $1 \leq a \leq k$, and $\mathbf{P}$ is a $k \times k$ matrix whose element P_{ab} equals the probability of an edge occurring between nodes belonging to groups a and b. It is known as the *affinity matrix*. We proceed to generate a random directed graph, $\mathcal{G}$, consisting of n nodes. Firstly, though, assign, to each node i of the graph, $\sigma_i \in \{1, \ldots, k\}$, a label indicating which community the node belongs to. These labels are chose independently, where, for each node i, $\mathbb{P}(\sigma_i = a) = n_a$. Let $\boldsymbol{u} = [\sigma_1, \ldots, \sigma_n]^T$ be the *ground-truth node assignments* of the graph. The random graph is generated to have an adjacency matrix, $\mathbf{A}$, whose elements are defined by

$$A_{ij} = \begin{cases} 0 & \text{if } i = j \\ X & \text{otherwise} \end{cases} \tag{2.4}$$

where $X \sim Be(P_{\sigma_i, \sigma_j})$.

This formulation matches the intuition of community structure, that the connectivity between two nodes depends solely on the community memberships of the two nodes. Note, also, that we do not allow self-loops.

The framework for testing community detection algorithms, which we will use, can now be summarised. Firstly, we generate synthetic datasets, by creating random graphs from the models described in sections 2.2.1 and 2.2.2, with varying parameters and known underlying ground-truth node assignments. Then we use the synthetically-generated graphs as input to various algorithms (an appropriate model is chosen for each algorithm), which provides, as output, an estimate to the community assignments. Finally, for each algorithm, we compare the estimated community assignments to the ground-truth values. This provides a notion of performance and accuracy to compare between the algorithms.

We now describe two models; one is a special case of the stochastic block model, created by imposing specific properties on the parameters, and the other a slightly modified version.

We also stress to the reader that we are only considering *non-overlapping communities*, where each node may only belong to one particular community.

2.2.1 Planted Partition Model

We will consider the formulation of the planted partition model as given by Decelle et al. [54]. To construct the planted partition model, we consider the stochastic block model, with $n_a = 1/k$, and the affinity matrix defined by

$$P_{ab} = \begin{cases} p_{in} & \text{if } a = b \\ p_{out} & \text{otherwise} \end{cases} \tag{2.5}$$

These properties essentially result approximately equal number of nodes and edges within each community. From this definition, p_{in} represents the probability of an edge occurring between two nodes belonging to the same community and p_{out} represents the probability of an edge occurring between two nodes belonging to the different communities. From now on, we refer to p_{in} and p_{out} as the *edge occurrence probabilities* for this model. We assume assortative structure so that $p_{in} > p_{out}$. This just matches the intuition of edges more likely to appear between nodes belonging to the same community than between nodes belonging to different communities.

(A) (B)

FIGURE 2.2: A set of plots for unlabelled, (a), and labelled, (b), adjacency matrices for random graph generated by planted partition model. The graphs were generated with $n = 300$, $k = 3$, $p_{in} = 0.7$ and $p_{out} = 0.3$.

FIGURE 2.3: A visualisation of an instance of a random graph generated by the planted partition model with $n = 240$, $k = 3$, $p_{in} = 0.2$ and $p_{out} = 0.01$.

An example of a random graph generated by the planted partition model is shown in figure 2.2a. We labelled nodes using $\sigma_i = 1 + (i \bmod k)$ for $i = 1, \ldots, n$ and generated the graph with $n = 300$, $k = 3$, $p_{in} = 0.7$, $p_{out} = 0.3$. The adjacency matrix of this graph is plotted with a pixel shaded red if the element in the adjacency matrix, corresponding to the location of the pixel, equals 1; while a pixel is shaded white if the element equals 0. Since we know the ground truth labelling of nodes, we can, without loss of generality, reorder the rows and columns of the adjacency matrix, such that it consists of blocks of nodes associated with the node community memberships. This is plotted in figure 2.2b. Note that since $k = 3$, there are $3 \times 3 = 9$ blocks, where the blocks are denser along the main diagonal since these correspond to edges between nodes belonging to the same community and $p_{in} > p_{out}$.

In figure 2.3, we show a visualisation of an instance of a random graph generated by the planted partition model with parameters with $n = 240$, $k = 3$, $p_{in} = 0.2$, $p_{out} = 0.01$.

Decelle et al. [38] conjectured a phase transition for sparse graphs generated from the planted partition model, using non-rigorous ideas from statistical physics [47]. Nadakuditi et al. [45] used methods from random matrix theory to present an asymptotic analysis

of spectra of random graphs to also demonstrate the presence of a phase transition. Essentially, we can distinguish between a *detectable* phase where it is possible to learn node assignments in a way that is correlated with the ground-truth node assignments of the graph, and an *undetectable* phase, where learning is impossible.

Let us define, for convenience, the variables $c_{in} = np_{in}$ and $c_{out} = np_{out}$. Consider a graph, generated by the planted partition model, and following the argument of [45], which we will not explain, one finds a transition occurring at the point

$$c_{in} - c_{out} = \sqrt{k(c_{in} + (k-1)c_{out})}. \tag{2.6}$$

In particular, let us consider the case where $k = 2$, so we find a transition at

$$c_{in} - c_{out} = \sqrt{2(c_{in} + c_{out})}. \tag{2.7}$$

Mossel et al. [47] proved the undetectable phase region of the conjecture given by equation equation (2.7). That is to say, it is impossible to meaningfully recover the node assignments when $c_{in} - c_{out} < \sqrt{2(c_{in} + c_{out})}$. Massoulié [55] and then, independently using a different proof, Mossel et al. [56] proved the detectable phase region of the conjecture, meaning it is possible to recover node assignments positively correlated with the ground-truth when $c_{in} - c_{out} > \sqrt{2(c_{in} + c_{out})}$. The techniques used to prove these results are beyond the scope of this report, however these results provide a very important limit on the ability of algorithms to detect communities (for the case of two underlying ground-truth communities). This motivates the development of algorithms which can efficiently (in nearly linear time) detect communities, in the sparse regime, up to this limit.

2.2.2 Hidden Clique Model

We will consider the following formulation as explained by Montanari [58, 61]. To construct the hidden clique model, we consider the stochastic block model with some modifications. We proceed to generate a graph with n nodes and k communities but with the affinity matrix, $\mathbf{P}$, becoming a $(k+1) \times (k+1)$ matrix defined by

$$P_{ab} = \begin{cases} p_{in} & \text{if } a = b,\ a \leq k,\ b \leq k \\ p_{out} & \text{otherwise} \end{cases} \tag{2.8}$$

Another tweak is that we now consider the variable n_a to represent the number of nodes within community a (rather than the expected fraction of nodes). Note that the number

(A) (B)

FIGURE 2.4: A set of plots for unlabelled, (a), and labelled, (b), adjacency matrices for random graph generated by hidden clique model. The graphs were generated with $n = 300$, $k = 3$, $p_{in} = 0.8$, $p_{out} = 0.2$, $n_1 = 50$, $n_2 = 40$, $n_3 = 20$.

FIGURE 2.5: A visualisation of an instance of a random graph generated by the hidden clique model with $n = 150$, $k = 1$, $n_1 = 30$, $p_{in} = 1.0$, $p_{out} = 0.1$. Nodes belonging to the hidden community, which in this case is in fact a clique, are coloured red whilst other nodes are coloured blue.

of nodes within each community does not necessarily sum to n, since we consider them to be 'hidden' within the graph. Also, we are interested in the regime where the size of these communities is small relative to the size of the graph. Once more, we assume assortative structure within the hidden communities so that $p_{in} > p_{out}$.

An example of a random graph generated by the hidden clique model is shown in figure 2.4a. We generated the graph with $n = 300$, $k = 3$, $p_{in} = 0.8$, $p_{out} = 0.2$, $n_1 = 50$, $n_2 = 40$, $n_3 = 20$. The adjacency matrix of this graph is plotted with a pixel shaded red if the element in the adjacency matrix, corresponding to the location of the pixel, equals 1; while a pixel is shaded white if the element equals 0. Since we know the ground truth labelling of nodes, we can, without loss of generality, reorder the rows and columns of the adjacency matrix, such that it consists of blocks of nodes associated with the node community memberships. This is plotted in figure 2.4b. We can see three dense blocks of size 50, 40 and 20 respectively, corresponding to the three hidden communities.

In figure 2.5, we show a visualisation of an instance of a random graph generated by the hidden clique model with parameters with $n = 150$, $k = 1$, $n_1 = 30$, $p_{in} = 0.3$, $p_{out} = 0.05$.

An interesting phase transition can also be derived for these models also. We follow the argument of Montanari [58, 61] to show this. We consider the simplest case of the model with only one hidden community (i.e. $k = 1$). We begin by generating a graph from the hidden clique model with n nodes and one community. Assume the n_1 nodes belonging to the hidden community make up a hidden community set, $\mathcal{S}$ (so that $|\mathcal{S}| = n_1$). Define $\mathbb{1}_n \in \{1\}^n$ as the n-dimensional vector with every element equal to 1. Also let $\mathbb{1}_{\mathcal{S}}$ be the indicator variable for nodes belonging to the hidden community set. Denote the adjacency matrix of the graph by $\mathbf{A}$, where each element is defined by

$$A_{ij} \sim Be(p_{ij}) \tag{2.9}$$

where

$$p_{ij} = \begin{cases} p_{in} & \text{if } i \in \mathcal{S},\ j \in \mathcal{S} \\ p_{out} & \text{otherwise} \end{cases} \tag{2.10}$$

Then, we get the following

$$\mathbb{E}(\mathbf{A}) = (p_{in} - p_{out})\mathbb{1}_{\mathcal{S}}\mathbb{1}_{\mathcal{S}}^T + p_{out}\mathbb{1}_n\mathbb{1}_n^T \tag{2.11}$$

and

$$Var(A_{ij}) = p_{out}(1 - p_{out}) \text{ if } \{i, j\} \not\subseteq \mathcal{S} \tag{2.12}$$

Denote $\widetilde{\mathbf{A}}$ as the *normalised adjacency matrix* of the graph, defined by

$$\widetilde{\mathbf{A}} \equiv \frac{1}{\sqrt{np_{out}(1 - p_{out})}}(\mathbf{A} - p_{out}\mathbb{1}_n\mathbb{1}_n^T) \tag{2.13}$$

By taking the expectation and using equation (2.11), we obtain

$$\mathbb{E}(\widetilde{\mathbf{A}}) = \frac{1}{\sqrt{np_{out}(1 - p_{out})}}(p_{in} - p_{out})\mathbb{1}_{\mathcal{S}}\mathbb{1}_{\mathcal{S}}^T \tag{2.14}$$

Let us write $\widetilde{\mathbf{A}} = \mathbb{E}(\widetilde{\mathbf{A}}) + (\widetilde{\mathbf{A}} - \mathbb{E}(\widetilde{\mathbf{A}}))$. Now using equation (2.14), we get

$$\widetilde{\mathbf{A}} = \frac{1}{\sqrt{np_{out}(1 - p_{out})}}(p_{in} - p_{out})\mathbb{1}_{\mathcal{S}}\mathbb{1}_{\mathcal{S}}^T + (\widetilde{\mathbf{A}} - \mathbb{E}(\widetilde{\mathbf{A}})) \tag{2.15}$$

Let us now define the following

$$\lambda \equiv \frac{p_{in} - p_{out}}{\sqrt{np_{out}(1 - p_{out})}} \tag{2.16}$$

$$\boldsymbol{u} \equiv \mathbb{1}_{\mathcal{S}} \tag{2.17}$$

(A) (B)

FIGURE 2.6: We plot the limiting spectral density of the normalised adjacency matrix under two regimes. We illustrate the case where $\lambda < 1$, in (a) and the case where $\lambda > 1$, in (b). The blue dot in (b) represents the eigenvalue $(\lambda + \lambda^{-1})$ associated with he eigenvector that pops-out of the main semicircle lobe. Both figures obtained from [61].

$$\mathbf{Z} \equiv \widetilde{\mathbf{A}} - \mathbb{E}(\widetilde{\mathbf{A}}) \tag{2.18}$$

Then we can re-write equation (2.15) as

$$\widetilde{\mathbf{A}} = \lambda \boldsymbol{u}\boldsymbol{u}^T + \mathbf{Z} \tag{2.19}$$

One can interpret λ as a signal-to-noise ratio, $\boldsymbol{u}$ as a signal (i.e. the ground-truth node assignments we which to infer) and $\mathbf{Z}$ as zero-mean noise with i.i.d. entries. We have essentially represented the problem of inferring the hidden community from the graph by a problem of estimating a rank-1 matrix in noise. Notice that for the generalisation with k communities, we would get a rank-k matrix plus noise for the normalised adjacency matrix.

Assume we generate a network associated with a normalised adjacency matrix, $\widetilde{\mathbf{A}}$, defined in equation (2.19) and are interested in reconstructing the vector of node assignments, $\boldsymbol{u}$. This problem has been investigated in many application under the guise of 'Low-rank deformation of Wigner matrices' [61]. Moreover much is known about the eigenvalue spectrum of such matrices. There is a very important spectral phase transition that exists: if $\lambda < 1$, the top eigenvector of the adjacency matrix, $\mathbf{A}$, is orthogonal to the vector we wish to reconstruct (i.e. $\langle \boldsymbol{v}_1(\mathbf{A}), \boldsymbol{u} \rangle \approx 0$), whereas, if $\lambda > 1$, the top eigenvector of $\mathbf{A}$ is correlated with the vector we wish to reconstruct and $\langle \boldsymbol{v}_1(\mathbf{A}), \boldsymbol{u} \rangle \approx (1 - \lambda^{-2})$ [61]. In the latter regime, one eigenvector pops out of the semicircle lobe, as illustrated in figure 2.6. This particular eigenvector is associated with the eigenvalue $\lambda + \lambda^{-1}$ [61].

This result is key since it specifically describes a threshold where traditional spectral methods such as standard principal component analysis (PCA) will not work (i.e. when $\lambda < 1$) and when it will produce a reconstructed vector correlated with the ground-truth

(i.e. when $\lambda > 1$). Moreover, we now have sufficient motivation to investigate methods where we can do better; more specifically we wish to study algorithms where we can essentially 'beat' this spectral threshold by producing a reconstructed vector correlated with the ground-truth in the regime where $\lambda < 1$. There is hope of achieving the improvement over standard PCA since we observe that the structure of the principal eigenvector of the matrix has two special properties. Firslty, it is *non-negative* (since the elements are node assignments or indicator variables and are thus either zero or one) and, secondly, it is *sparse* (since we are interested in the regime where the size of the hidden community or clique is small relative to the size of the graph). We will see how to utilise of this observation in more detail in section 3.4.1 since it forms the basis of a class of algorithms.

2.3 Financial Networks

2.3.1 Prices and Returns of Financial Assets

Financial assets are instruments claiming to have monetary value that can be bought and sold. Financial assets can be separated into broad classes, with examples including stocks, bonds or real estate [42, 60]. The values of these assets is reflected in their price, which varies with time. Investors may wish to decide between which asset classes to invest in at any time, a process known as *asset allocation* [60]. Also, within a particular asset class, an investor wishes to allocate money to specific assets, a process known as *portfolio selection* [60]. For this report, we will focus on portfolio selection of stocks in our application of community detection algorithms, due to data availability constraints; however a very similar scheme may be used to tackle asset allocation also.

Investors tend not to consider the prices of assets they have invested in, but rather the *return* generated. Let us consider a financial asset whose price at time t is $p(t)$. One popular measure of return is called the *rate of return* [43, 60] at a time t, denoted by $r(t)$, which is defined as

$$r(t_0) = \frac{p(t_1) - p(t_0)}{p(t_0)} \tag{2.20}$$

where we can interpret t_1 as the time when the investor sold the asset, and t_0 as the time when the investor bought the asset.

Critically, the rate of return is sensitive to large changes for longer time horizons [7]. In particular, we can consider a different measure of return, which is equivalent to a return with a constant interest rate [7]. We can generalise the concept interest rates with the

simple example of an investor placing money (investing) in a bank account, as explained in [3, 41]. The amount of money initially invested is referred to as the *principal*. We then assume money grows by a multiplicative factor, where the gain is paid into the account by the bank. This process is often called *compounding*. The time at which the interest is compounded, is called the *compounding period*. *Compound interest* involves interest being paid on both the principal and the accumulated interest up to the present [7]. Typically, we are interested in the number of compounding periods in one year (i.e. the number of times the interest on our principal is compounded each year) [41]. Denote the principal by w_0, the amount in the account at time t by w_t, the interest rate by y and the number of compounding periods in a year by m. Then the amount within the account holdings after 1 year is given by

$$w_1 = w_0(1 + (y/m))^m \tag{2.21}$$

We can imagine diving a year into infinitesimally small compounding periods, and then determine the effect of this continuous compounding by taking the limit of ordinary compounding [3]. Notice the total number of compounding periods in a length of t years is given by mt. Thus the effect of continuous compounding is

$$w_t = \lim_{m\to\infty} w_0(1 + (y/m))^{mt} = w_0 \exp(yt) \tag{2.22}$$

If we divide equation (2.22) by the initial investment w_0, and take the natural logarithm, we get a representation for the return, $r = yt$. This indicates that taking the natural logarithm results in a constant interest rate. We have thus arrived at another measure for return, called the *logarithmic return*, which is defined by

$$r(t_0) = \log\left(p(t_1)\right) - \log\left(p(t_0)\right) \tag{2.23}$$

where, once more, we can interpret t_1 as the time when the investor sold the asset, and t_0 as the time when the investor bought the asset.

There are several advantages to using logarithmic returns, as explained in [66], which we will briefly summarise. Firstly, if we assume asset prices have a *log normal* distribution, then the logarithmic returns are conveniently normally distributed [66]. The reasons why assuming a log normal distribution may be appropriate for dynamic pricing of assets is beyond the scope of this report. Secondly, for small rates of return, the logarithmic return is approximately equal to the rate of return [66]. To see this, consider the approximation

FIGURE 2.7: Plots of prices and logarithmic returns for Anglo American plc (AAL) between 2004 and 2014

result from equation (2.24) combined with equations (2.20) and (2.23).

$$\log(1+x) \approx x, \text{ for } x \ll 1 \tag{2.24}$$

Thirdly, we benefit from numerical stability since the addition of small numbers is numerically stable, whilst multiplying small numbers is subject to *arithmetic underflow* [66].

However, there are disadvantages to using the logarithmic return, including the issue that the derivation is only correct if the interest rate is constant [7, 66]. Nevertheless, the logarithmic return is widely used in the literature (e.g. see [7, 8, 9, 33, 39, 57]), and hence we shall use it for the rest of the report, and the reader should note we shall use the terms 'return' and 'logarithmic return' interchangeably from now on. An example plot of price and logarithmic return for a stock is shown in figure 2.7.

2.3.2 Mean-Variance Portfolio Theory

The term *portfolio* relates to investing in a combination of different assets. We can characterise a portfolio by *portfolio weights*, where the weight of an asset within the portfolio is given by the ratio of the value of the position in the asset divided by the total value of the portfolio. We are particularly interested in the return of the portfolio, which is related to the mean of the returns of the individual assets that make up the portfolio and the risk of the portfolio, which is related to the variance of the returns of the individual assets. This is the source of the term mean-variance portfolio theory. We realise an intuitive and inherit trade-off between risk and return; if the investor wishes to realise a larger return, he must bear a higher risk. A more detailed explanation of this relationship is given in [3, 43]. Rather, we are simply interested in finding the most efficient, or *minimum-variance portfolio* for a given requested portfolio return (that is to say, the investor requests a specific expected return, and wishes to form a portfolio that

has the lowest variance of all possible portfolio that can deliver the specified expected return). We can formalise the mean-variance portfolio setting in the following way, which summarises the explanations from [3, 7, 43].

Let P be a portfolio comprising of n assets, where the return of the portfolio is denoted by r_P and the variance of the portfolio is denoted by σ_P^2. Denote the return of asset i by r_i, the variance of the asset i return by $\sigma_i^2 \equiv Var(r_i)$, and the weight of asset i in the portfolio by w_i. Also let X_0 denote the total amount invested in the portfolio (i.e. initial investment in the portfolio by the investor) and X_{0i} represent the the amount invested in asset i. We select the amounts in the assets forming the portfolio such that

$$\sum_{i=1}^{n} X_{0i} = X_0 \tag{2.25}$$

We define the portfolio weights using

$$w_i = \frac{X_{0i}}{X_0} \text{ for } i = 1, \ldots, n \tag{2.26}$$

meaning that

$$\sum_{i=1}^{n} w_i = 1 \tag{2.27}$$

Notice that a negative weight indicates a *short position* in that asset, and that the returns of the individual assets and portfolio are random variables.

In particular we can represent the return of the portfolio by

$$r_P = \sum_{i=1}^{n} w_i r_i \tag{2.28}$$

By using equation (2.28), we obtain the expected return of the portfolio

$$\mathbb{E}(r_P) = \sum_{i=1}^{n} w_i \mathbb{E}(r_i) \tag{2.29}$$

and the variance of the portfolio return

$$Var(r_P) \equiv \sigma_P^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \rho_{ij} \sigma_i \sigma_j \tag{2.30}$$

where ρ_{ij} is the correlation coefficient of the returns of assets i and j, defined as

$$\rho_{ij} = \frac{Cov(r_i, r_j)}{\sigma_i \sigma_j} \tag{2.31}$$

Note that the standard deviation of the returns of the portfolio (or any asset) is often called its *volatility*.

This formulation serves a key question; given estimates of each assets returns, variances and covariances (which one can obtain from historical data), how does one pick a selection of these assets, for any given time period, in order to form the best portfolio for the investor? We see in equation (2.30), that by simply investing in assets which have a lower correlation with one another (i.e. a lower value of ρ_{ij}), we can reduce the variance, and thus, the risk of the portfolio. This process is known as *diversification* [3, 60]. Therefore, for any given time period (and possibly dynamically), finding groups of assets, where the returns have higher correlation within groups and lower correlation between groups would help by presenting 'baskets' of assets that the investor can pick from knowing selecting from a range of baskets would be beneficial (of course which assets to select from inside the basket relates to the risk-return trade off). This serves as the main motivation for the application of community detection algorithms within financial networks.

2.3.3 Constructing Financial Networks

From section 2.3.2, we understand one way to help minimise risk in constructing portfolios involves analysing the correlation coefficients of returns between two assets. In order to study all possible correlations between all available assets, we construct a weighted, undirected and fully-connected network of assets, which we call the *financial network*. The following model has been considered by [5, 8, 9, 33, 57].

Let us consider the situation where the investor is faced with n financial assets, and has access to historical price data for all these assets for T time steps. The time steps may be trading days, or weeks, for instance, and the appropriate choice will depend on the type of assets available.

We proceed to construct a graph with n nodes, where each nodes represents an asset, and assign to the i-th node a single time series, denoted by X_i, which is defined as

$$X_i = \{x_i(1), \ldots, x_i(T)\} \tag{2.32}$$

where $x_i(t)$ describes the logarithmic return of asset i at time t, defined by equation (2.23). This time series describes the evolution of the logarithmic return of the asset over T time steps. We then model the weight of an edge connecting nodes i and j of the graph by the cross-correlation between the time series corresponding to assets i and j. We form a cross-correlation matrix, denoted by $\mathbf{C}$, whose elements are defined by

$$C_{ij} = \frac{\langle X_i X_j \rangle - \langle X_i \rangle \langle X_j \rangle}{\sqrt{\left[\langle X_i^2 \rangle - \langle X_i \rangle^2\right]\left[\langle X_j^2 \rangle - \langle X_j \rangle^2\right]}} \tag{2.33}$$

where the $\langle \cdots \rangle$ notation denotes a time average, so that

$$\langle X_i \rangle = \frac{1}{T}\sum_{t=1}^{T} x_i(t) \tag{2.34}$$

$$\langle X_i^2 \rangle = \frac{1}{T}\sum_{t=1}^{T} x_i^2(t) \tag{2.35}$$

$$\langle X_i X_j \rangle = \frac{1}{T}\sum_{t=1}^{T} x_i(t) x_j(t) \tag{2.36}$$

We also assume each time series X_i has been standardised (before we assign to node i) by using

$$X_i \coloneqq \frac{X_i - \langle X_i \rangle}{\sqrt{\langle X_i^2 \rangle - \langle X_i \rangle^2}} \tag{2.37}$$

so that

$$\langle X_i \rangle = 0 \tag{2.38}$$

$$\langle X_i^2 \rangle - \langle X_i \rangle^2 = 1 \tag{2.39}$$

Note that the cross correlation values is just a sample estimate for the correlation coefficient, ρ_{ij}, used in section 2.3.3, calculated from the historical data.

We can then characterise the financial network by the correlation matrix, which we also refer to as the network's weighted adjacency matrix.

In figure 2.8, we have plotted a correlation matrix using data of 80 stocks listed on the FTSE 100 (see appendix A for a list) between 2011 and 2013. Notice that the main diagonal has all elements equal to one, as you would expect, and that there are very few negative elements (i.e. very few assets that are anti-correlated with one another).

FIGURE 2.8: Example of a correlation matrix. Evaluated from an ensemble of 80 stocks listed on the FTSE 100 (see appendix A for a list) using data between 01/01/2011 and 01/01/2013.

The problem statement can now summarised. Given a financial network, how can we group nodes into communities where correlations are higher within the communities and lower between the communities? Contrary to graphs with community structure described in section 2.2, the weights of the edges rather than the topology of the network are crucial in determining community memberships. In other words, we focus solely on the weighted adjacency matrix of the graph. Also the reader should note that the correlations between asset returns will vary over time, and thus representing this relationship dynamically (rather than over a one long period of time) is very important since investors may wish to change their positions in order to react to the dynamics of market conditions.

Chapter 3

Community Detection Algorithms

In this chapter we introduce several community detection algorithms present in the literature that are based upon different approaches. We introduce spectral clustering, modularity-based optimisation, non-linear power iteration and message-passing algorithms, with specific reference to their application on generative block models. We seek to explain sufficient intuition motivating the algorithms in addition to summarising their mathematical derivations.

3.1 Spectral Clustering

The basis of all spectral clustering algorithms is the transformation of a set of variables into the set of points in space whose coordinates are elements of eigenvectors of a matrix, and then the clustering of these points using well-known clustering algorithms [20, 29].

We consider the spectral clustering algorithm described in [20, 29], whose intuition is explained in [20, 21, 29].

Firstly, we compute the Laplacian matrix of the network, using definition 2.13, where we assume n nodes in the graph, as usual. We then compute the k eigenvectors of the Laplacian matrix associated with the k largest eigenvalues. Denote the eigenvectors by $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k$ and the eigenvalues by $\lambda_1, \ldots, \lambda_k$. The eigenvectors represent a k-tuple of real numbers associated with each vertex in the graph. We think of this association as a mapping from the vertices into a k-dimensional space. This embedding is characterised by $F : V \to \mathbb{R}^k$ where $F(i) = (\boldsymbol{u}_1^{(i)}, \ldots, \boldsymbol{u}_k^{(i)})$ and $\boldsymbol{u}_j^{(i)}$ denotes the i-th element of the

FIGURE 3.1: A visualisation of the embedding for the spectral clustering algorithm. The graph was generated using the planted partition model with $n = 150$, $k = 3$, $p_{in} = 0.8$ and $p_{out} = 0.2$. We have chosen 3 distinct communities since we can easily display the embedding in a 2-dimensional space. The coordinates for each point are the corresponding entries in the 2 eigenvectors of the Laplacian matrix considered. We label the ground-truth node assignments by colour (i.e. points with the same colour represent nodes belonging to the same community in the graph), and we can see that a k-means clustering algorithm would be applied to return cluster memberships that match the true community memberships.

j-th eigenvector. We have essentially represented node i of the graph as a point in a k-dimensional space where the coordinates are the i-th elements of all the top k eigenvectors of the Laplacian matrix. Finally, we apply the embedding as input to the popular *k-means clustering* algorithm. The cluster memberships of the n data points are precisely the estimated node assignments for the initial network. Notice, from the definition of the Laplacian matrix, that the vector of all ones (i.e. a vector with every element equal to one) is the principal eigenvector, so these embedded values remain the same for all nodes. Therefore, knowledge of this eigenvector does not help discriminate between different vertices, and hence the information is not useful. Thus we shall only apply the embedding to the top $k - 1$ eigenvectors (i.e. the top k eigenvectors excluding the all-ones vector), and can therefore represent the mapping in a $k - 1$ dimensional space.

To visualise what such an embedding looks like, refer to figure 3.1, where we generated an example graph using the planted partition model and 3 communities (this is purposely chosen so we can easily identify the embedding in a 2-dimensional space). The coordinates for each point are the corresponding entries in the 2 eigenvectors of the Laplacian matrix considered. We labelled the ground-truth node assignments by colour in the figure (i.e. points with the same colour represent nodes belonging to the same community in the underlying graph), and we can see that a k-means clustering algorithm would be applied to return cluster memberships that match the ground-truth, since the data points corresponding to different clusters are well separated.

Note that the algorithm does require to compute k eigenvectors of a matrix, and this can be achieved using the power method. Also, the accuracy of the algorithm will largely

depend on the k-means algorithm which has been shown to converge to local minima in a cost measure (rather than global), but despite this, has been shown to work well in practical applications [20, 29].

3.2 Modularity-based Optimisation

We introduce the algorithms, firstly considered by [18, 19], by explaining the intuition behind 'good' community partitions. In essence, the key ingredient involves determining partitions of the network where there are fewer edges *than expected* between nodes belonging to different communities. For instance, if the number of links between nodes associated between different communities is approximately the same as what one would expect to find given random assignment of edges within the network, then it is unlikely this provides evidence of meaningful community structure [19]. Moreover, we can also consider partitions where there are more edges than expected between nodes belonging to the same community.

Definition 3.1. The *null model* with respect to a network, whose adjacency matrix is given by $\mathbf{A}$, is the random graph denoted by $\mathcal{G}$, where each edge has a probability of $\frac{d_i d_j}{2m}$ of occurring. d_i is the degree of node i and $2m \equiv \sum_{ij} A_{ij}$.

The null model defined above is proposed as a baseline distribution if edges were randomly placed within the network.

Definition 3.2. Given a partition, $\boldsymbol{\sigma}$, of a network, the *modularity* is defined as: $Q(\boldsymbol{\sigma}) = \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{d_i d_j}{2m}\right)\delta(\sigma_i, \sigma_j)$.

The modularity is therefore considered a cost function for a partition of the network where larger modularity values indicate stronger community structure [18].

Definition 3.3. The *modularity matrix* is denoted by $\mathbf{B}$, whose elements B_{ij} are defined by $B_{ij} = A_{ij} - \frac{d_i d_j}{2m}$.

The aim of modularity optimisation algorithms is to find a partition of the network with the maximum value of modularity associated. Since searching over all possible partitions is exponential in the number of nodes of the network, the problem is NP-hard computationally [19]. Thus, we seek approximate methods that provide near-optimal solutions.

In the literature, there exists a variety of approximation algorithms for accurate and fast modularity optimisation, such as greedy algorithms, simulated annealing, spectral algorithms and extremal optimisation [29]. Within this report, we describe all these algorithms but only implement and test the greedy agglomerative method on synthetic data, since we consider it to be a faster version whilst maintaining similar accuracy to other modularity optimisation algorithms.

3.2.1 Greedy Algorithm

The greedy algorithm of Clauset et al. [13] starts with all nodes as single groups and successively mergers two groups to form a larger community such that the modularity of the new partition increases after the joining [13, 29]. Moreover, the algorithm keeps, permanently, the merger with the largest increase in modularity (hence at each step we compute ΔQ, the change in modularity, using definition 3.2). This is iterated until no further increase in modularity is possible [13]. Note that for a network with n nodes and m edges, the algorithm has complexity $O\left((m+n)n\right)$, or $O\left(n^2\right)$ for a sparse graph [29].

A different greedy algorithm has been proposed by Blondel et al. [23], that is also applicable for weighted networks [29]. This algorithm is commonly known as the *Louvain method*, and we shall also often refer to the algorithm by this name. We initialise each node to belong to an individual community, and then repeat the following two phases until there is no further increase in modularity possible. In the first phase, we sequentially consider each node, and given node i, we compute the increase in modularity, ΔQ, that results from moving node i into a neighbour community, and then permanently select the transition that yields the greatest increase in modularity [23, 29]. In the second phase, two communities are connected if an edge exists between any node belonging to the communities [23, 29]. Figure 3.2 illustrates the two phases of the algorithm on an example network. This figure is taken from the Blondel et al. reference [23].

3.2.2 Simulated Annealing

The simulated annealing algorithm of Kirkpatrick et al. [2] is an iterative procedure that explores a space of possible states looking for the global optimum of modularity, which we denote by Q [2, 29]. Updates from one state to another are accepted with probability 1 if the transition results in an increase in the modularity. Otherwise, the update is only accepted with a small probability $\exp(-\beta\Delta Q)$, where ΔQ is the change in modularity

FIGURE 3.2: Illustration of the two phases of the greedy method of [23]. The first phase involves optimisation of modularity using local changes, and the second phase aggregates the nodes into communities. The two phases are repeated until no further improvement of modularity is possible. For this example, only two passes are required until termination. This figure is reprinted from the Blondel et al., reference [23].

(i.e. value of modularity after the transition minus the value before) and β represents the inverse-temperature of the system [2, 29]. The idea behind accepting a transition that results in a decrease in modularity with a small probability is to increase the chance of finding the global maximum (i.e. decrease the chance of converging towards local maxima) [29]. The algorithm converges to a stable state at some point, depending on the number of states explored and how β is varied, but it can be a good approximation for Q.

A more recent implementation by Guimera et al. [14] consists of iterations that involve both individual and collective steps. Within the individual step, an individual node is moved to a community at random, whilst the collective step consists of merging two communities or splitting one community [14]. Typically, each iteration involves n^2 individual steps and n collective steps, where n represents the number of nodes in the network [29]. This method can approximate the true maximum of modularity very accurately, and note that, due to the variation of parameter selection (such as initial temperature and inverse-temperature chosen), an exact complexity cannot be estimated, but it is typically very slow and hence should only be used on small graphs [29].

3.2.3 Extremal Optimisation

The extremal optimisation algorithm of Duch and Arenas [16] is a heuristic search method that involves recursively bi-partitioning the network [16, 29]. It begins with a random partition and uses the contribution of each node to the modularity as a fitness measure with the movement of nodes with the lowest fitness value. The fitness function value of a node i is given by

$$q_i = \kappa_{\sigma(i)} - d_i e_{\sigma(i)} \tag{3.1}$$

where d_i is the degree of node i, $\kappa_{\sigma(i)}$ is the number of neighbours node i has in the community it belongs to, and $e_{\sigma(i)}$ is the fraction of edges in the network that connects at least one node which belongs to the community of node i [16, 29]. Using this notation, one can re-write the modularity by $Q = \frac{1}{2m}\sum_i q_i$. We also normalise the variables q_i by dividing by d_i to obtain

$$\rho_i = \frac{\kappa_{\sigma(i)}}{d_i} - e_{\sigma(i)} \tag{3.2}$$

so that $-1 \leq \rho_i \leq 1$ for all i. Therefore we have expressed the global cost function in terms of a sum over all vertices (through the local variables ρ_i) and, hence, we can optimise the global variable, Q, by optimising over the local variables [29]. At each iteration of the algorithm we calculate ρ_i for every node i and move the node with the lowest value to the other community. Note that this transition alters the overall partition so the fitness values need to be re-evaluated, whilst we repeat this process until no further improvement in the modularity is possible [16, 29]. Extremal optimisation has empirically shown to achieve similar accuracy to simulated annealing but with a faster run time [29].

3.2.4 Spectral Algorithm

We shall describe the spectral method of Newman [18] using the derivation explained in [18, 19, 29], and by firstly considering networks with only two ground-truth communities. Recall the notation used for modularity and the modularity matrix in definitions 3.2 and 3.3, and let $\boldsymbol{\sigma}$ represent the vector of node assignments, where $\sigma_i = 1$ if node i belongs to class 1 and $\sigma_i = -1$ if it belongs to class 2. Then the modularity can be written as

$$\begin{aligned} Q &= \frac{1}{2m}\sum_{i,j}\left(A_{ij} - \frac{d_i d_j}{2m}\right)\delta(\sigma_i, \sigma_j) \\ &= \frac{1}{4m}\sum_{i,j}\left(A_{ij} - \frac{d_i d_j}{2m}\right)(\sigma_i\sigma_j + 1) \\ &= \frac{1}{4m}\sum_{i,j} B_{ij}\sigma_i\sigma_j \\ &= \frac{1}{4m}\boldsymbol{\sigma}^T\mathbf{B}\boldsymbol{\sigma} \end{aligned} \tag{3.3}$$

We can rewrite $\boldsymbol{\sigma}$ as a linear combination of the eigenvectors of $\mathbf{B}$, which we denote by $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ (where we label in decreasing order corresponding to the magnitude of its

associated eigenvalue), so that

$$\boldsymbol{\sigma} = \sum_{i=1}^{n} (\boldsymbol{u}_i^T \boldsymbol{\sigma}) \boldsymbol{u}_i \tag{3.4}$$

Using this result in equation (3.3) yields

$$Q = \sum_i (\boldsymbol{u}_i^T \boldsymbol{\sigma}) \boldsymbol{u}_i^T \mathbf{B} \sum_j (\boldsymbol{u}_j^T \boldsymbol{\sigma}) \boldsymbol{u}_j = \sum_{i=1}^{n} (\boldsymbol{u}_i^T \boldsymbol{\sigma})^2 \lambda_i \tag{3.5}$$

where λ_i is the eigenvalue of $\mathbf{B}$ associated with the eigenvector $\boldsymbol{u}_i$. We aim to maximise the modularity by choosing the elements of $\boldsymbol{\sigma}$. From equation (3.5), we see this can be achieved by increasing the weights of the largest (i.e. most positive) eigenvalues. However, we cannot just set $\boldsymbol{\sigma}$ to be proportional to the largest eigenvector, $\boldsymbol{u}_1$, as we imposed each element to be either +1 or -1. Instead, we seek an approximate method, where we proceed to set the values of σ_i based upon the sign of the i-th component of $\boldsymbol{u}_1$. Essentially, the algorithm involves computing the leading eigenvector of the modularity matrix and then partitioning the nodes of the network into two groups according to the signs of the corresponding elements in the eigenvector.

In order to extend this approach for networks which contain more than two communities, we repeat this procedure for dividing any one community into two communities until no further sub-division increases the value of modularity, at which point the algorithm terminates.

The spectral method for optimising modularity is quite fast, since computing the leading eigenvector of the modularity matrix can be computed using the well-known power method. Due to the special structure of the modularity matrix, the computation of the leading eigenvector takes $O(m+n)$ time, so that one partition of the network takes $O(n(m+n))$ time or $O(n^2)$ for a sparse graph [29]. As we need to repeatedly partition the network in order to optimise the modularity, the overall complexity is $O(dn(m+n))$ where d represents the depth of the hierarchical division. Typically, in practice $d \approx \log(n)$, so that for sparse graphs, the total time taken for this spectral algorithm is approximately $O(n^2 \log(n))$. The spectral method is faster than simulated annealing and extremal optimisation, although not as fast as the greedy algorithm, based on empirical results [29]. An added benefit is the extensibility of the spectral method to applications with weighted networks.

3.3 Belief Propagation Algorithm

The belief propagation (BP) algorithm of [17, 54] is designed to infer the group assignment from an instance of a graph generated by the planted partition model. We shall describe Decelle et al.'s BP algorithm [54] for the particular case of the planted partition model, but note that message passing algorithms such as BP are very similar in nature and incredibly useful in many other applications outside the problem of community detection [29]. We also notify the reader that this algorithm is derived from principles in statistical physics, that we do not wish to delve into too much detail (since it is mostly inconsequential to the motivation of this report), and therefore shall present a higher-level overview and less detailed summary of the derivation by Decelle et al. [54].

Recall the notation used for introducing the planted partition model in section 2.2.1. We begin by realising the probability the planted partition model with parameters $\theta = \{k, n_a, \{P_{ab}\}\}$ generates a graph $\mathcal{G}$ consisting of n nodes, with an associated adjacency matrix $\mathbf{A}$ and estimated node assignments q_i is given by

$$\mathbb{P}(\mathcal{G}, \{q_i\} \mid \theta) = \prod_{i \neq j} \left[P_{q_i q_j}^{A_{ij}} (1 - P_{q_i q_j})^{1 - A_{ij}} \right] \prod_i n_{q_i} \tag{3.6}$$

Assuming we know the underlying parameters of the block model, θ, as well as observing the graph $\mathcal{G}$, we form the probability distribution over the group assignments by applying Bayes' theorem

$$\mathbb{P}(\{q_i\} \mid \mathcal{G}, \theta) = \frac{\mathbb{P}(\mathcal{G}, \{q_i\} \mid \theta)}{\sum_{t_i} \mathbb{P}(\mathcal{G}, \{q_i\} \mid \theta)} \tag{3.7}$$

where $\{t_i\}$ now represent the ground-truth node assignments (instead of $\{\sigma_i\}$) for convenience.

We make an important connection between this problem and a result studied in statistical physics. The *Boltzmann distribution* of a generalised Potts model (consider $\{\sigma\}$ simply as a set of discrete variables) with Hamiltonian $H(\{\sigma\})$ at inverse temperature β is given by

$$\mu(\{\sigma\}) = \frac{\exp(-\beta H(\{\sigma\}))}{\sum_{\{\sigma\}} \exp(-\beta H(\{\sigma\}))} \tag{3.8}$$

where each assignment of the variable $\{\sigma\}$ has a weight $\exp(-\beta H(\{\sigma\}))$ known as the *Boltzmann weight* and $Z(\beta) = \sum_{\{\sigma\}} \exp(-\beta H(\{\sigma\}))$ is called the *partition function* (i.e. the sum of the Boltzmann weights over all possible configurations) [72, 75, 76]. We now use that equation (3.7) corresponds to a Boltzmann distribution of a generalised Potts

model at unit temperature (i.e. $\beta = 1$) with Hamiltonian given by

$$H(\{q_i\} \mid \mathcal{G}, \theta) = -\sum_i \log(n_{q_i}) - \sum_{i \neq j} \left[A_{ij} \log\left(c_{q_i q_j}\right) + (1 - A_{ij}) \log\left(1 - \frac{c_{q_i q_j}}{n}\right) \right] \tag{3.9}$$

where we define $c_{ab} = nP_{ab}$ to be convenient notation for sparse networks. The corresponding Boltzmann distribution is

$$\mu(\{q_i\} \mid \mathcal{G}, \theta) = \mathbb{P}(\{q_i\} \mid \mathcal{G}, \theta) = \frac{\exp(-H(\{q_i\} \mid \mathcal{G}, \theta))}{\sum_{\{q_i\}} \exp(-H(\{q_i\} \mid \mathcal{G}, \theta))} \tag{3.10}$$

and the corresponding partition function is

$$Z(\mathcal{G}, \theta) = \sum_{\{q_i\}} \exp(-H(\{q_i\} \mid \mathcal{G}, \theta)) \tag{3.11}$$

The BP algorithm is essentially an iterative produce used to compute the partition function by ignoring the correlation between neighbours of a node while conditioning on its label [54]. Moreover, [54] states that these correlations do not exist if the network of interactions between nodes is a tree and that, if the network observed is locally treelike, then the correlation terms become negligible making the BP algorithm exact in the asymptotic limit (i.e. as $n \to \infty$). We can derive the BP algorithm update equations for the case of an undirected network (the directed case is a little more complex with additional equations). We write the BP algorithm as a set of *messages*, denoted by $m_{q_i}^{i \to j}$, that are essentially marginal probabilities of a node i belonging to a community q_i excluding the evidence from its neighbour node j [54]. The messages that govern this algorithm (similar to the derivation for loopy belief propagation [78]) are then given by

$$m_{u_i}^{i \to j} = \frac{1}{Z^{i \to j}} n_{u_i} \prod_{k \in N(i) \setminus j} \left[\sum_{u_k} c_{u_i, u_k}^{A_{ik}} \left(1 - \frac{c_{u_i, u_k}}{n}\right)^{1 - A_{ik}} m_{u_k}^{k \to i} \right] \tag{3.12}$$

where $N(i)$ denotes the neighbourhood of node i, $\{u_i\}$ represents the estimated node assignments at a particular iteration step of the algorithm and $Z^{i \to j}$ is a constant that normalises the messages ensuring they form a probability distribution (i.e. so that $\sum_{u_i} m_{u_i}^{i \to j} = 1$) [54]. In essence, this is the normalised belief in the node i belonging to a community u_i excluding the evidence from the node j [78]. The algorithm involves applying equation (3.12) iteratively until a fixed point is reached with messages $\{m_{q_i}^{i \to j}\}$ [54]. We can write the marginal probability (or normalised belief) of node i belonging to

community u_i, denoted by $b^i_{u_i}$, as

$$b^i_{u_i} = \frac{1}{Z^i} n_{u_i} \prod_{k \in N(i)} \left[\sum_{u_k} c^{A_{ik}}_{u_i,u_k} \left(1 - \frac{c_{u_i,u_k}}{n}\right)^{1-A_{ik}} m^{k \to i}_{u_k} \right] \tag{3.13}$$

where, once more, we use a normalisation constant Z^i [54].

The update for each step iteratively has $O\left(n^2\right)$ complexity, since there are messages between every pair of nodes, but, for large sparse networks (i.e. for large n and $c_{ab} = O\left(1\right)$), we ignore terms of order in n [54]. This implies a node simply sends the same message to all non-adjacent nodes (which is described as an external field in statistical physics) and thus the algorithm only needs to consider a number of messages equal to twice the number of edges in the networks, hence each iteration requires $O\left(n\right)$ time [54]. We can understand this by considering the messages for two cases. Firstly, if nodes i and j are not adjacent,

$$m^{i \to j}_{u_i} = \frac{1}{Z^{i \to j}} n_{u_i} \prod_{k \notin N(i) \setminus j} \left[1 - \frac{1}{n} \sum_{u_k} c_{u_k u_i} m^{k \to i}_{u_k} \right] \prod_{k \in N(i)} \left[\sum_{u_k} c_{u_k u_i} m^{k \to i}_{u_k} \right] = b^i_{u_i} + O\left(\frac{1}{n}\right) \tag{3.14}$$

where the messages are independent of j to the leading order [54]. Now, if nodes i and j are adjacent in the network

$$m^{i \to j}_{u_i} = \frac{1}{Z^{i \to j}} n_{u_i} \prod_{k \notin N(i)} \left[1 - \frac{1}{n} \sum_{u_k} c_{u_k u_i} m^{k \to i}_{u_k} \right] \prod_{k \in N(i) \setminus j} \left[\sum_{u_k} c_{u_k u_i} m^{k \to i}_{u_k} \right] \tag{3.15}$$

We can now re-write equation (3.13) as

$$m^{i \to j}_{u_i} = \frac{1}{Z^{i \to j}} n_{u_i} \exp(-h_{u_i}) \prod_{k \in N(i) \setminus j} \left[\sum_{u_k} c_{u_k u_i} m^{k \to i}_{u_k} \right] \tag{3.16}$$

where we ignored the $O\left(\frac{1}{n}\right)$ term and have defined the external field by

$$h_{u_i} = \frac{1}{n} \sum_k \sum_{u_k} c_{u_k u_i} b^k_{u_k} \tag{3.17}$$

as explained by [54]. The marginal probabilities can now also be re-written as

$$b^i_{u_i} = \frac{1}{Z^i} n_{u_i} \prod_{k \in N(i)} \exp(-h_{u_i}) \prod_{j \in N(i)} \left[\sum_{u_j} c_{u_j u_i} m^{j \to i}_{u_j} \right] \tag{3.18}$$

Equations (3.16) to (3.18) define the crucial steps in each iteration of the BP algorithm, which is called 'BP-INFERENCE' in [54]. The precise details can be seen in [54], but we summarise the key steps. We start with the messages as a random vector, compute the initial marginal probabilities and external field. We then iteratively, until a convergence criterion is met and/or for a fixed number of steps, apply the three update equations. Finally, we output the estimated group assignment by using $\{q_i\} = \underset{q}{\operatorname{argmax}}\, b_q^i$ (i.e. make the assignment of node i to the group that maximises the marginal probabilities of node i belonging to any of the groups). As [54] have analysed, the main body of the algorithm has $O(n)$ complexity, while the number of iterations required until convergence is not known exactly and varies for different networks, so choosing an appropriate value for the maximum number of iterations based on a training set of networks would be practical.

Recall that, for the above derivations, we assumed knowledge of the underlying parameters of the block model, θ. However, in practice, given a new network, this is not the case, so we need to be able to learn the underlying parameters of the network before applying the BP inference algorithm. Decelle et al. [54] do provide another BP algorithm for learning or estimating the parameters that is based on the popular *expectation-maximisation* (EM) algorithm. We do not aim to derive the update equations for this case, though, but will rather explain the intuition behind it. In order to infer the parameters, we can aim to find the maximum a-posteriori estimator. Let $\widehat{\theta}$ denote the estimator for the parameters, then $\widehat{\theta} = \underset{\theta}{\operatorname{argmax}}\, \mathbb{P}(\theta \mid \mathcal{G})$. Applying Bayes' theorem we find that

$$\mathbb{P}(\theta \mid \mathcal{G}) = \frac{\mathbb{P}(\mathcal{G} \mid \theta)\mathbb{P}(\theta)}{\mathbb{P}(\mathcal{G})} = \frac{\mathbb{P}(\theta)}{\mathbb{P}(\mathcal{G})} \sum_{\{q_i\}} \mathbb{P}(\mathcal{G}, \{q_i\} \mid \theta) \tag{3.19}$$

where $\{q_i\}$, again, represents the estimated group assignments. Therefore $\widehat{\theta}$ can also be found by maximising the partition function defined in equation (3.11) over θ. By minimising this function for θ results in stationarity conditions known as the *Nishimori conditions* in statistical physics [54]. There is an iterative method for learning the parameters based on the Nishimori conditions and can be written in terms of messages to be used in a BP algorithm

$$n_a = \frac{1}{n} \sum_i b_a^i \tag{3.20}$$

$$c_{ab} = \frac{1}{n_a n_b n} \sum_{(i,j) \in E} \frac{c_{ab}(m_a^{i \to j} m_b^{j \to i} + m_b^{i \to j} m_a^{j \to i})}{Z^{ij}} \tag{3.21}$$

where we denote the set of edges of $\mathcal{G}$ by E and denote Z^{ij} as the BP estimate for the partition function defined by

$$Z^{ij} = \sum_{a<b} c_{ab}(m_a^{i\to j} m_b^{j\to i} + m_b^{i\to j} m_a^{j\to i}) + \sum_a c_{aa} m_a^{i\to j} m_a^{j\to i} \text{ for } (i,j) \in E \tag{3.22}$$

The EM algorithm uses the above BP update equations for the expectation step starting with random initialisation for θ. Combining the parameter learning steps with the inference of node assignments enables a complete BP algorithm, which Decelle et al. have described in [54] and called 'BP-LEARNING'.

For the purposes of our testing using the synthetically generated data from the planted partition model, we shall use the C++ implementation of this algorithm provided by Decelle et al. [54] which is available from [69].

3.4 NLPI and AMP Algorithms

The following algorithms aim to partition networks based upon the hidden clique model described in section 2.2.2, in order to identify the underlying hidden community.

3.4.1 Non-linear Power Iteration

The intuition behind the non-linear power iteration (NLPI) method is fairly straightforward. Recall we wish to reconstruct the node assignment vector denoted by $\boldsymbol{u}$. Equation (2.19) shows $\boldsymbol{u}$ is the principal eigenvector of a rank-1 matrix in noise, called the normalised adjacency matrix, and denoted by $\widetilde{\mathbf{A}}$. We use the standard power iteration algorithm with one extra step; we additionally apply a separable non-linear function that acts component-wise. We choose the non-linear function to force the reconstructed vector to adhere to one of the properties desired. In particular we can apply 'positive-part thresholding' [61], where we keep only the positive elements of the vector (and set the negative elements to zero) at each iteration. The following is a recursive definition of one iteration of the general approach, where t indexes the iteration

$$\begin{aligned} \boldsymbol{z}^{t+1} &= \widetilde{\mathbf{A}}\widehat{\boldsymbol{u}}^t \\ \widehat{\boldsymbol{u}}^t &= f_t(\boldsymbol{z}^t) \end{aligned} \tag{3.23}$$

where

$$\begin{aligned} \widehat{\boldsymbol{u}}^0 &= [1, \ldots, 1]^T \\ \boldsymbol{z} &= [z_1, \ldots, z_n]^T \\ f_t(\boldsymbol{z}) &= [f_t(z_1), \ldots, f_t(z_n)]^T \end{aligned} \tag{3.24}$$

Since we will consider positive-part thresholding,

$$f_t(z_i) = \begin{cases} z_i & \text{if } z_i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.25}$$

for all $i = 1, \ldots, n$.

This algorithm takes advantage of the fact that the leading eigenvector is non-negative, an observation identified in section 2.2.2. We will test this algorithm based on positive-part thresholding empirically for synthetically generated networks for varying SNR and sizes of the hidden community in section 4.3. However, it is also important to analyse the algorithm theoretically, for instance by quantifying the (possible) improvement over spectral methods in different regimes. Unfortunately, analysing this algorithm in terms of precise asymptotics is very difficult since there are dependencies existent after any number of iterations [61].

3.4.2 Approximate Message Passing

Although the NLPI method works well in practice (and we shall show this empirically later), we still seek an algorithm that can also be analysed theoretically. The *approximate message passing* (AMP) algorithm involves one modification to the NLPI, where a memory term is subtracted. The following is a recursive definition of one iteration of the general approach, where t again indexes the iteration

$$\begin{aligned} \boldsymbol{z}^{t+1} &= \widetilde{\mathbf{A}}\widehat{\boldsymbol{u}}^t - b_t\widehat{\boldsymbol{u}}^{t-1} \\ \widehat{\boldsymbol{u}}^t &= f_t(\boldsymbol{z}^t) \end{aligned} \tag{3.26}$$

where we define

$$\begin{aligned}
\widehat{\boldsymbol{u}}^{-1} &= [0, \ldots, 0]^T \\
\widehat{\boldsymbol{u}}^{0} &= [1, \ldots, 1]^T \\
\boldsymbol{z} &= [z_1, \ldots, z_n]^T \\
f_t(\boldsymbol{z}) &= [f_t(z_1), \ldots, f_t(z_n)]^T \\
b_t &\equiv \frac{1}{n} \sum_{i=1}^{n} f_t^{'}(z_i)
\end{aligned} \tag{3.27}$$

and again consider positive-part thresholding. We remark that the explicit formula for b_t is chosen since it cancels the statistical bias (i.e. decorrelates) on $\widehat{\boldsymbol{u}}_i^{t+1}$ due to $\widehat{\boldsymbol{u}}_i^{\leq t}$. The explanation for this result is beyond the scope of this report, but we refer the reader to [26, 28, 36, 37, 59] for details.

Chapter 4

Experiments on Synthetic Data

In this chapter we aim to experiment with community detection algorithms on synthetically generated data. We shall consider the algorithms described in the previous chapter, and use data created from the appropriate generative block model. The goal of the experiments on synthetic data is to understand how the underlying network structure, and the variation of parameters therein, affects the performance of different algorithms [49]. In general the experiments will proceed as follows. We generate a network with the appropriate block model and specified parameters, with underlying ground-truth node assignments set. We then measure the accuracy of the specific algorithm investigated as we vary model parameters. These parameters include the sparsity of the graph and the edge occurrence probabilities. This then allows us to draw conclusions regarding the relative performance of community detection algorithms in controlled conditions given by networks with common properties. After undertaking this data-driven testing approach, we conclude by comparing the algorithms investigated by discussing their advantages and issues in order to advocate of the application on particular empirical circumstances based on their the suitability .

4.1 Spectral Clustering and Modularity-Optimisation Algorithms

We shall test the spectral clustering algorithm of section 3.1 and the greedy modularity optimisation method of section 3.2.1 using identical synthetic data generated using the planted partition model described in section 2.2.1. We aim to understand the quality (i.e.

accuracy) of the partitions generated by the two algorithms as we vary both the sparsity of the network and the edge occurrence probabilities. We have found very similar performance for all the algorithms as we increase the number of communities, and therefore, in the interests of being concise in our analysis, we only consider the case of graphs with two ground-truth communities. Furthermore, we wish to test, empirically, how close we can reliably detect communities up to the phase transition point. Additionally, we assume assortative community structure throughout. We apply each of generated networks as input to the Laplacian spectral clustering and greedy modularity method, and measure the accuracy of reconstruction of the node assignments by computing the *overlap* using the definition by Decelle et al. [38, 54]. Denote the overlap by T, and let the estimated node assignments be given by $\{q_i\}$ with ground-truth node assignments given by $\{\sigma_i\}$, then we define T by

$$T = \max_{\pi} \frac{\frac{1}{n}\sum_i \delta(\sigma_i, \pi(q_i)) - \frac{1}{k}}{1 - \frac{1}{k}} \tag{4.1}$$

where π ranges over the permutations on q elements and k represents the number of communities. This definition means that $0 \leq T \leq 1$ with a higher value implying improved reconstruction and more accurate results, and an overlap of 0 meaning the algorithm is, on average, no better than random uniform selection of node assignments.

The traditional spectral clustering algorithm has been implemented in MATLAB based on the description given in section 3.1, whilst the MATLAB implementation for the greedy modularity algorithm has been obtained from Le Martelot [67].

We begin by considering the dense regime and generating 100 networks from the planted partition model with common parameters $n = 200$, $k = 2$, $p_{in} = 0.9$. However, we vary p_{out} for each of the 100 networks to get a different value for the ratio c_{out}/c_{in}, signifying a variation in the edge occurrence probabilities. The results are shown in figure 4.1, where the overlap is plotted in the vertical axis and the horizontal axis is the value for the ratio c_{out}/c_{in}. We see very similar performance for both algorithms in this regime and can also identify the phase transition region described in section 2.2.1, since there is a very sharp drop in overlap for a value of $c_{out}/c_{in} \approx 0.85$. Overlap values for both algorithms for $c_{out}/c_{in} < 0.85$ are approximately 1 implying perfect reconstruction of the ground-truth node assignments, whilst for $c_{out}/c_{in} > 0.85$, the overlap drops very sharply towards 0. Using equation (2.7) we find that, for this specification of the model, the predicted phase transition point occurs at $c_{out}/c_{in} \approx 0.8564$. Therefore we realise the performance of both algorithms matches up to the theoretical phase transition point in this dense regime.

FIGURE 4.1: A plot for the overlap of reconstructed node assignments of the Laplacian spectral clustering (blue) and greedy modularity optimisation methods (red) as the relative edge occurrence probabilities (c_{out}/c_{in}) are varied. 100 networks were generated with common parameters, $n = 200$, $k = 2$, $p_{in} = 0.9$. Both algorithms perform very similarly in this regime and we can clearly identify the detectable and undetectable phases since the there is a very sharp drop in the overlap values for both algorithms. Using equation (2.7) we find that, for this specification of the model, the predicted phase transition point occurs at $c_{out}/c_{in} \approx 0.8564$.

Let us now consider a sparser regime, where we again generate 100 networks from the planted partition model but now with common parameters $n = 200$, $k = 2$, $p_{in} = 0.6$. The results are shown in figure 4.2. Notice that both algorithms perform similarly well for $c_{out}/c_{in} < 0.5$ but the modularity method seems to perform better until we get the sharp drop in overlap values. This is a very important observation to note, that the sharp drop in overlap values, for both algorithms, occurs at $c_{out}/c_{in} \approx 0.75$. Using equation (2.7) we find that, for this specification of the model, the predicted phase transition point occurs at $c_{out}/c_{in} \approx 0.8256$, suggesting that, as the network gets sparser, both algorithms performance declines with respect to obtaining perfect reconstruction within the detectable phase. These observations should not be a surprise given the derivation of the algorithms, and we will now explain known issues with both these algorithms that cause this decline in accuracy.

Recall the spectral clustering algorithm requires applying a k-means algorithm to the embedded vectors in a $k - 1$ dimensional subspace. For the case with dense graphs, the eigenvalue spectrum of the Laplacian is separated and the embedded vectors can be seen to be arbitrarily well separated also. However, for the case where the graph is sparse, the eigenvector elements (i.e. coordinates in the subspace) are very similar, and the k-means algorithm performance will be very poor, results in much lower accuracy for the spectral clustering algorithm. This observation is summarised as an illustration in figure 4.3.

For the dense regime, we generate a network from the planted partition model with $n = 150$, $k = 3$, $p_{in} = 0.8$ and $p_{out} = 0.2$, whilst for the sparse regime, we generate a network with the same values for n and k, but $p_{in} = 0.08$ and $p_{out} = 0.02$. Note

FIGURE 4.2: A plot for the overlap of reconstructed node assignments of the Laplacian spectral clustering (blue) and greedy modularity optimisation methods (red) as the relative edge occurrence probabilities (c_{out}/c_{in}) are varied. 100 networks were generated with common parameters, $n = 200$, $k = 2$, $p_{in} = 0.6$. Both algorithms perform similarly for $c_{out}/c_{in} < 0.5$ but the modularity method seems to perform better until we get the sharp drop in overlap values due to the phase transition. Using equation (2.7) we find that, for this specification of the model, the predicted phase transition point occurs at $c_{out}/c_{in} \approx 0.8256$.

(A) (B)

FIGURE 4.3: A visualisation of the embedding for the spectral clustering algorithm in the dense, (a), and sparse, (b), regimes. The graph for the dense regime was generated using the planted partition model with $n = 150$, $k = 3$, $p_{in} = 0.8$ and $p_{out} = 0.2$, whilst the graph for the sparse regime was generated using the same values of n and k but with $p_{in} = 0.08$ and $p_{out} = 0.02$. We label the ground-truth node assignments by colour (i.e. points with the same colour represent nodes belonging to the same community in the graph). Notice that, for the dense regime, all data points with different colours are well separated so the k-means algorithm can accurately recover the cluster memberships. However, for the sparse regime, every data (red green and blue) are concentrated in the same region in the 2-dimensional embedding space, so the k-means algorithm cannot recover the cluster memberships accurately at all. This illustrates an issue with the Laplacian spectral clustering method.

the important regulation that, for this illustration, the ratio c_{out}/c_{in} is identical for both regimes, and is below the theoretical phase transition (i.e. we are in the detectable region). We see that, for the dense regime, all data points belonging to different communities are well separated, and points belonging to the same community are clustered together, so the k-means algorithm can accurately recover the cluster memberships. However, for the sparse regime, all data points are clustered together, so the k-means algorithm cannot recover the cluster memberships accurately, and the overlap for the spectral clustering method will be very low indeed.

We now consider an issue associated with all modularity optimisation algorithms, known as the *resolution limit* [29, 34]. Essentially, communities that are small when compared to the whole graph may not be distinguished even though they are well defined communities or even cliques, and therefore this problem has an impact on practical applications [29]. As [29, 34] explain, if the change in modularity, denoted by ΔQ, arising by merging two communities is positive, then the two groups will be clustered together. Let e_i be the number of edges within community i and e_{ij} be the number of edge between communities i and j. The expression for the change in modularity is then given by

$$\Delta Q_{ij} = \frac{e_{ij}}{m} - 2\left(\frac{d_i}{2m}\right)\left(\frac{d_j}{2m}\right) \tag{4.2}$$

where d_i is the sum of the degrees of the vertices that belong to community i. Notice that $\Delta Q_{ij} > 0$ (and therefore the two communities will be merged) if and only if $e_{ij} > \frac{d_i d_j}{2m}$ [34]. This is problematic because of the null model considered; the modularity tends to expect a value of $e_{ij} < 1$, which makes just a single edge between the communities (i.e. $e_{ij} = 1$) unexpected, so that two communities will be merged even though there is just one edge between them [34].

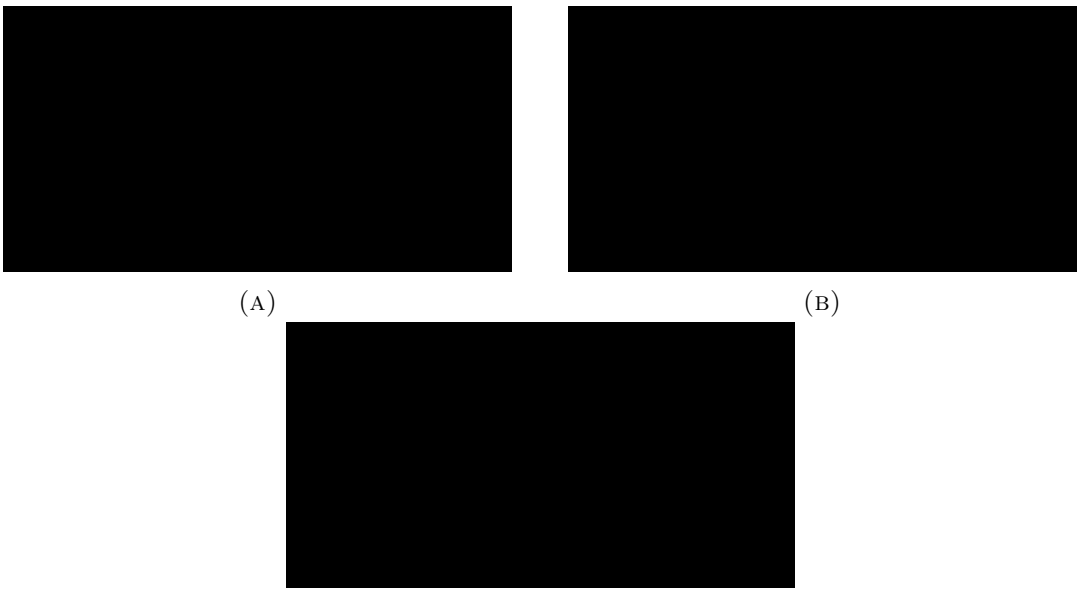
Another problem with modularity optimisation is known as *extreme near-degeneracy* [29, 34]. Good et al. [34] explain that the number of partitions with near-optimal modularity (i.e close with respect to the global maximum modularity value) grows exponentially with the number of nodes in the network [29]. This results in extreme degeneracies in the modularity which causes problems in determining the partition which maximises modularity as well as finding high modularity partitions, since partitions that have similar high modularity values associated are not necessarily similar to one another [29, 34]. This means that, the more modular a network is (i.e. the more communities it contains), it is actually becomes more difficult to determine the optimum partition among the suboptimal ones, which is a counter-intuitive result [34].

To summarise, both spectral clustering and modularity optimisation algorithms can detect communities up to the phase transition in the dense regime. However, we have seen how the performance declines, if the network is sparse, for both algorithms. We have explained some of the issues observed that can account for the disappointing results, although there is no deep understanding regarding the modularity optimisation methods [29]. Describing the behaviour of the modularity methods is important, though, and there have been many situations where the algorithm works well in practical applications [18, 19, 29, 34]. This should not be startling given how the synthetic data we have used for our experimental tests are, in general, not well representative of real-world networks considered in practice. Since the greedy algorithm is very fast and will be favourable for larger networks, we would recommend using this algorithm instead of other modularity optimisation methods as well as traditional spectral clustering as it achieves similar or better accuracy with faster run time. For the case of small sized networks, the pragmatic option seems most suitable, where these different algorithms may be run and then compared in training cases, with the best one evaluated used for further investigation in test applications.

4.2 Belief Propagation Algorithm

We wish to find a class of algorithms that, empirically, seem to reliably detect communities in networks generated by the planted partition model up to the phase transition in the sparse regime as well as the dense regime. Both the spectral clustering and modularity optimisation methods could not achieve this, but we wish test whether the BP algorithm of Decelle et al. [54] will succeed. We shall analyse the accuracy of the partitions generated by the BP algorithm as we vary the following parameters: size of the graph, sparsity of the graph, number of communities and the edge occurrence probabilities. All of the test networks generated from the block model will be used as input to the algorithm and the accuracy of reconstruction will, once more, be measured by computing the overlap (given by equation (4.1)). The algorithm has been implemented in C++ by Decelle et al. [54], and we have obtained the code from [69].

We begin by considering the effect on the overlap as the edge occurrence probabilities are varied for small sized networks all consisting of two communities but with different sparsity. We therefore generate three sets of 100 networks using the planted partition model with common parameters $n = 200$ and $k = 2$ but with a different value for the parameter for p_{in} for each set. The three different values are $p_{in} = \{0.1, 0.5, 0.9\}$ and they represent different levels of sparsity for the generated graph. For each set of networks (and



(A) (B)

(C)

FIGURE 4.4: Plots for the overlap of reconstructed node assignments from the belief propagation algorithm as the ratio c_{out}/c_{in} is varied for three different levels of sparsity. Three different values of p_{in} are considered, each one representing a different level of sparsity, whilst 100 networks were generated with common parameters, $n = 200$ and $k = 2$, for each set. The results with $p_{in} = 0.1$ is shown in (a), the results with $p_{in} = 0.5$ is shown in (b), whilst the results with $p_{in} = 0.9$ is shown in (c). We see the BP algorithm performs similarly well up to the phase transition point across the three levels of sparsity, given this small-sized network test case.

thus each value for p_{in}), we vary the parameter p_{out} in steps for each of the 100 networks to obtain a different value for the ratio c_{out}/c_{in}, signifying the variation in the edge occurrence probabilities for different levels of sparsity on small-sized graphs. The results are plotted in figure 4.4. We see the BP algorithm performs similarly well across the different regimes of sparsity. This can be inferred from the points at which the overlap sharply declines from a value near 1 to a value close to 0 is approximately those predicted by the phase transition. This indicates good performance on small-sized networks, especially compared to the spectral clustering and greedy modularity algorithms.

In order to see if this effect is also true for small-sized networks but with more than two ground-truth communities, we generate two sets of 100 networks using the planted partition model with common parameters $n = 200$ and $k = 4$, again, with a different value for the parameter for p_{in} for each set of networks. Once more, we vary the parameter p_{out} in steps for each of the 100 networks to obtain a different value for the ratio c_{out}/c_{in}, signifying the variation in the edge occurrence probabilities for different levels of sparsity on small-sized graphs (but this time with 4 ground truth communities rather than 2). The

(A) (B)

FIGURE 4.5: Plots for the overlap of reconstructed node assignments from the belief propagation algorithm as the ratio c_{out}/c_{in} is varied for two different levels of sparsity and four ground-truth communities. Two different values of p_{in} are considered, each one representing a different level of sparsity, whilst 100 networks were generated with common parameters, $n = 200$ and $k = 4$, for each set. The results with $p_{in} = 0.5$ is shown in (a), whilst the results with $p_{in} = 0.9$ is shown in (b). We see the BP algorithm performs similarly well up to the phase transition point across the three levels of sparsity, given this small-sized network test case.

plots are shown in figure 4.5. These plots for overlap against c_{out}/c_{in} looks very similar to those shown in figures 4.4b and 4.4c . This indicates that the performance of the BP algorithm does not vary in small-sized networks for differing number of communities.

We also noted, through the simplistic derivation in section 3.3, that the algorithm should also work well for large sparse graphs since each step of the algorithm takes $O(n)$ time. To test this result empirically, we generate two sets of sparse and larger networks, differing only in the number of ground-truth communities. We generate two sets of 100 networks using the planted partition model with common parameters $n = 10000$ and $p_{in} = 0.01$ but with two different values of k. The plots are shown in figure 4.6. Using equation (2.6) we find that, for the specification of the model with $k = 2$, the predicted phase transition point occurs at $c_{out}/c_{in} \approx 0.8098$, whilst, for the specification of the model with $k = 4$, the predicted phase transition point occurs at $c_{out}/c_{in} \approx 0.6555$. The results do indicate the BP algorithm accurately detects communities for different values of k up to their respective predicted phase transition points for large and sparse networks.

Combining the implications of our empirical testing for the belief propagation algorithm and comparing it with the synthetic data results for the spectral clustering and greedy modularity methods in section 4.1, we do find the BP algorithm has several advantages for the specific test case of the planted partition model. Firstly, we can detect communities reliably in the sparse regime up to the corresponding phase transition point in addition to the dense regime. Also, the algorithm is more scalable in the sparse regime due to faster run time. However, these remarks should not be a surprise given that the BP

(A) (B)

FIGURE 4.6: Plots for the overlap of reconstructed node assignments from the belief propagation algorithm as the ratio c_{out}/c_{in} is varied for both two and four ground-truth communities in an example of a large, sparse graph. Two different values of k are considered, whilst 100 networks were generated with common parameters, $n = 10000$ and $p_{in} = 0.01$, for each set. The results with $k = 2$ is shown in (a), whilst the results with $k = 4$ is shown in (b). Using equation (2.6) we find that, for the specification of the model with $k = 2$, the predicted phase transition point occurs at $c_{out}/c_{in} \approx 0.8098$, whilst, for the specification of the model with $k = 4$, the predicted phase transition point occurs at $c_{out}/c_{in} \approx 0.6555$. Hence, we observe the BP algorithm performs well up to the phase transition points for both values of k in the large, sparse regime.

algorithm was specifically designed to detect communities from graphs generated by the planted partition model and, therefore, we could have concerns about the generalisation achieved, since statistical block models do not generate graphs with similar properties to those found in real world networks (i.e. they are not well representative) [54].

The BP algorithm has been applied to real-world networks in [54], but they considered only small-sized networks that are not very practical, such as the popular Zachary's karate club network as well as a network of political books. Whereas, modularity methods have been tested on a wide array of data sets and networks in the literature, and have performed well. Therefore we conclude that, for applications involving moderately-sized real-world networks, modularity optimisation methods should be a first point of call. We also suggest possible tweaks could be made to certain modularity methods that tailor them to the specific application at hand (similarly to how the BP algorithm we considered has been tailored to the planted partition model), which could also improve performance. An example application of this is a financial network, as we shall discuss in chapters 5 and 6.

4.3 NLPI and AMP Algorithms

We shall use the procedure outlined by Montanari in [61] to test the NLPI and AMP algorithms, where the synthetic data used is simply an instance of a normalised adjacency matrix. Recall equation (2.19), where we decomposed the matrix into a signal term

FIGURE 4.7: A plot of $\langle \boldsymbol{u}, \widehat{\boldsymbol{u}} \rangle$ for the NLPI algorithm for different values of λ and ε. The synthetic data is generated from a network with $n = 500$ nodes and a grid with resolution 100, whilst the algorithm is run for 50 iterations.

(weighted by a signal-to-noise ratio term) plus a noise term. We will consider the case of a network generated by the hidden clique model with n nodes and one hidden community consisting of k nodes. We may choose the community memberships arbitrarily. Using previous notation, let us denote $\boldsymbol{u}$ as the indicator variable for nodes belonging to the hidden community (i.e. so that precisely k elements of $\boldsymbol{u}$ are equal to one and the rest equal zero), λ as the SNR and $\mathbf{Z}$ as the noise term. We set the elements of $\mathbf{Z}$ to be $Z_{ij} \sim \mathcal{N}(0, 1/n)$ i.i.d. entries. Therefore, given a value of λ and our ground-truth community assignments, we can construct the normalised adjacency matrix, denoted by $\widetilde{\mathbf{A}}$, using equation (2.19). This matrix will serve as input to the NLPI and AMP algorithms, which produce an output representing their reconstructed node assignments, which we shall denote by $\widehat{\boldsymbol{u}}$. We represent the accuracy of the algorithms by the inner product of the ground-truth and reconstructed node assignments, $\langle \boldsymbol{u}, \widehat{\boldsymbol{u}} \rangle$. Notice this value lies between 0 and 1, where a larger value indicates improved reconstruction and better accuracy.

Now, with the framework set up, we can construct our own tests, seeking the accuracy as both the SNR and the size of the hidden community are varied. Define a new variable $\varepsilon \equiv k/n$ representing the proportion of all nodes that belong to the hidden community, then we can construct a grid of points representing different values of λ and ε. We, finally, plot the values of $\langle \boldsymbol{u}, \widehat{\boldsymbol{u}} \rangle$ for each point on the grid to analyse the behaviour of the algorithms.

Utilising this approach with $n = 500$ nodes, varying λ between 0 and 1.2, choosing a grid resolution of 100 and running 50 iterations of the NLPI algorithm, we show in figure 4.7, the accuracy of the NLPI algorithm plotted against λ and ε.

The motivation for the NLPI algorithm is to detect the node assignments more accurately than standard PCA for small-sized hidden communities. The results can be analysed by

FIGURE 4.8: A plot of $\langle \boldsymbol{u}, \widehat{\boldsymbol{u}} \rangle$ for the AMP algorithm for different values of λ and ε. The synthetic data is generated from the same network with $n = 500$ nodes as used in figure 4.7, and a grid with resolution 100, whilst the algorithm is run for 50 iterations.

focusing on the accuracy for small values of ε and all ranges of λ. We have previously discussed that the transition for standard PCA algorithms for small-size hidden communities is $\lambda = 1$, where, for values above this threshold, reconstruction is possible, and below, it is impossible. Figure 4.7 illustrates that for small values of ε (e.g. between 0.05 and 0.2), the reconstructed vector is correlated with ground-truth (i.e. $\langle \boldsymbol{u}, \widehat{\boldsymbol{u}} \rangle > 0$). In particular, we can get good reconstruction for values of λ all the way down to 0.8. Thus we have shown to beat the spectral threshold!

Although the NLPI algorithm can be shown to beat the spectral threshold empirically by using the synthetic data strategy of [61] that we used above, analysing its asymptotic behaviour mathematically is not trivial, as we have already noted. This served as the principal motivation for the AMP algorithm where precise asymptotics could be derived theoretically (however this is beyond the scope of this report). We shall now analyse the empirical performance of the AMP algorithm using identical synthetic data (i.e. the same normalised adjacency matrix input for every value pair (λ, ε) in the grid) as we tested the NLPI algorithm with. We also chose to run the AMP algorithms for 50 iterations. We have plotted the accuracy for the AMP algorithm in figure 4.8.

Figure 4.8 illustrates that for small values of ε (e.g. between 0.05 and 0.2), the reconstructed vector is correlated with ground-truth. In particular, we get good reconstruction for values of λ all the way down to 0.8. Thus we have shown to beat the spectral threshold once more! We also note that the striking similarity between figure 4.7 and figure 4.8 is to be expected given the formulations of these algorithms.

We have seen how NLPI and AMP algorithms can be used to detect hidden communities in networks generated by a hidden clique model. In particular we have seen the empirical improvements over standard spectral methods such as PCA. Moreover, the NLPI approach of applying a suitable non-linear function as an extra step to traditional power-iteration

algorithm may be used for any problem where an eigenvector with special properties (e.g. sparsity or non-negativity) needs to be found. Although the application is beyond the scope of this report, the AMP algorithm has had much success when applied to other problems such as *compressed sensing*; we refer the reader to [26, 28, 36, 37, 59] for more details.

Chapter 5

Community Detection in Financial Networks

In this chapter we aim to motivate and study modified modularity algorithms to detect communities within real-world financial networks and compare their performance with baseline methods. Firstly, we explain the process of gathering the data to create the financial network. Then we investigate and apply algorithms on both synthetically generated and real-world data. We focus on modularity optimisation and study two naive modularity maximisation methods studied in previous chapters as well as two modified modularity optimisation techniques that are tailored to detect communities in our specific setting of financial networks based on empirical correlation matrices. By determining the 'best' method for the static financial network, we make an important step in identifying communities in time-varying financial networks.

5.1 Constructing the Real-world Financial Network

The dataset we use consists of daily closing prices of 80 stocks in the FTSE 100 index, which we obtained from [64, 65]. The time period considered is between the beginning of 2004 to the end of 2013, a total of 2501 prices. This is the data we obtained after the removal of a few data points due to incomplete data across different stocks. The complete list of stocks is given in appendix A. We then calculated the logarithmic return for each stock and for each time period. We generated a time series of these returns and associated each stock with a single time series. By using the method described in section 2.3.3, we

FIGURE 5.1: A plot of expected return against volatility of the 80 stocks in the FTSE 100 index considered. Data points obtained from price data during the period 01/01/2004 - 11/11/2013.

proceeded to construct a financial network represented by a fully-connected, undirected and weighted graph. There are 80 nodes in this network (each one representing one of the stocks), and the weights on the edges connecting any two nodes is the cross-correlation between the time series of returns associated with the stocks represented by the two nodes. We stress, at this point, the data of the whole period (01/01/2004 - 11/11/2013) is currently represented by one single network. Figure 5.1 shows a plot of the expected return against the volatility for each stock considered during this period.

5.1.1 Random Matrix Theory

The correlation matrix is representing the weighted adjacency matrix of the network, and in order to better understand the weights in the network, we wish to refer to an important result from Random Matrix Theory (RMT) that has been outlined in [4, 5, 15, 57]. In particular, we wish to distinguish between random and non-random properties of empirical correlation matrices.

A correlation matrix created from n random time series of length T, in the limits $n \to +\infty$ and $T \to +\infty$ with $1 < T/n < +\infty$, has a specific distribution of eigenvalues known as the *Sengupta-Mitra distribution* [4, 15, 39, 57]. This distribution is defined by

$$\rho(\lambda) = \begin{cases} \frac{T}{n} \frac{\sqrt{(\lambda_+ - \lambda)(\lambda - \lambda_-)}}{2\pi\lambda} & \text{if } \lambda_- \leq \lambda \leq \lambda_+ \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

where the maximum and minimum eigenvalues (λ_+ and λ_- respectively) are given by

$$\lambda_+ = \left(1 + \sqrt{\frac{n}{T}}\right)^2 \tag{5.2}$$

and

$$\lambda_- = \left(1 - \sqrt{\frac{n}{T}}\right)^2 \tag{5.3}$$

Therefore the set of eigenvalues of an empirical correlation matrix that lies within this distribution is considered to occur purely as a result of random noise [15, 39, 57]. Moreover, we may regard any eigenvalue larger than λ_+ to represent important structure within the data [15, 39, 57].

Analysing the deviation of the eigenvalue spectrum of empirical correlation matrices constructed from real-world financial data from the RMT distribution constitutes an effective method to filter noise out from the data. For example, we constructed the correlation matrix from the FTSE 100 data set (described in section 5.1) and plotted the eigenvalue spectrum for this matrix alongside the corresponding Sengupta-Mistra distribution (i.e. the RMT prediction with $n = 80$ and $T = 2501$) in figure 5.2. We observe two interesting regions of the eigenvalue spectrum outside the RMT prediction. Firstly, the largest eigenvalue of the correlation matrix, which we shall denote by λ_m, is much larger than all other eigenvalues. Also, the eigenvector associated with the largest eigenvalue, denoted by v_m, has all elements positive. This has been observed in many previous studies of empirical correlation matrices, and this eigenvalue is also called the *market mode*, meaning this component acts as a common factor influencing all assets in the market [39, 57]. Secondly, we observe a few eigenvalues just outside the RMT predicted region (i.e. eigenvalues just larger than λ_+ and much smaller than λ_m). We believe these components reflect a mesoscopic level of groups of stocks within the market (i.e. neither at the level of individual stocks in the form of noise, nor at the level of the entire market in the form of the market mode eigenvalue) [57], hence we expect members of these groups of stocks to demonstrate similar underlying properties, such as related sector classifications.

We proceed to utilise the eigenvalue spectrum observed for the data set and the RMT prediction to filter out the empirical correlation matrix to reflect a mesoscopic structure, as achieved by [57]. Recall the correlation matrix for our FTSE 100 data set is a 80×80 matrix denoted by $\mathbf{C}$ and that we denote λ_i as the $i-th$ eigenvalue of $\mathbf{C}$ and $\boldsymbol{v}_i$ represents the eigenvector associated with λ_i. We are able to decompose this matrix as the sum of three matrices

$$\mathbf{C} = \mathbf{C}^{(r)} + \mathbf{C}^{(g)} + \mathbf{C}^{(m)} \tag{5.4}$$

(A) (B)

FIGURE 5.2: Plots of eigenvalue spectra for empirical correlation matrix and RMT prediction, (a), in addition to a zoomed-in version, (b). The empirical correlation matrix was constructed from the daily log-returns of the FTSE 100 data set, and its eigenvalue spectrum is plotted in red. The RMT prediction is the Sengupta-Mitra distribution with appropriate parameters ($n = 80$, $T = 2501$), and is plotted in blue. The zoomed-in graph identifies the existence of eigenvalues outside of the region predicted by RMT, whilst the zoomed-out graph clearly shows the maximum eigenvalue (i.e the market mode eigenvalue) with a value of about 28.

where $\mathbf{C}^{(r)}$ represents the correlation matrix corresponding to the random components, defined by

$$\mathbf{C}^{(r)} \equiv \sum_{i:\lambda_i \leq \lambda_+} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^T \tag{5.5}$$

$\mathbf{C}^{(m)}$ represents the correlation matrix corresponding to the market mode component, defined by

$$\mathbf{C}^{(m)} \equiv \lambda_m \boldsymbol{v}_m \boldsymbol{v}_m^T \tag{5.6}$$

and $\mathbf{C}^{(g)}$ represents the remaining correlations

$$\mathbf{C}^{(g)} \equiv \sum_{i:\lambda_+ < \lambda_i < \lambda_m} \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^T \tag{5.7}$$

Therefore, we now have a representation of a filtered empirical correlation matrix, $\mathbf{C}^{(g)}$, which represents the mesoscopic (group level) correlations of the stocks, which we shall use, crucially, as the input to several community detection algorithms. Simply, it can be thought of as a weighted adjacency matrix of a new filtered network.

5.2 Community Detection Algorithms

So far we have been able to construct a financial network based on the correlations of daily logarithmic returns of stocks and, using RMT, a filtered correlation matrix that

represents a new financial network with links (hopefully) representing group-level correlation. Although, the question still remains, given either the initial or filtered correlation matrix, how does one produce a set of groups of stocks with greater correlations within a group than between groups? From previous sections, we understand the notion of community detection within graphs, which we shall also refer to as *binary networks*, and have analysed several algorithms that tackle this problem. However, in these problems we analysed adjacency matrices that contained binary elements (i.e. a '1' if an edge exists in the graph and a '0' otherwise), whereas, in this problem, we study a weighted adjacency matrix with elements as real numbers. The reader should also note the adjacency matrix studied in previous sections is directly related to the structure of the network in question, whereas in this case, it is related to the weights of links between nodes. This suggests the possibility of having to modify previously studied algorithms to tailor their behaviour for this scenario.

Given our comments in previous chapters, modularity optimisation has worked well in other practical applications and does not require prior knowledge of the number of or sizes of the communities to be detected [50, 57]. Hence, we shall use modularity optimisation methods as a basis for community detection in financial networks.

5.2.1 Modularity Optimisation Methods

Recall in section 3.2, we introduced the notion of modularity optimisation as a method for community detection within networks. We shall once more consider algorithms for modularity optimisation, but for the case of financial networks.

Let us denote a partition of n nodes, in the financial network, into communities/groups by the vector $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_n]^T$, where σ_i denotes the group to which node i belongs to. This is essentially the group to which stock i belongs. We then define the modularity for this partition, $Q(\boldsymbol{\sigma})$, by

$$Q(\boldsymbol{\sigma}) = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(\sigma_i, \sigma_j) \tag{5.8}$$

where $\mathbf{A}$ is the weighted adjacency matrix of the network, $k_i \equiv \sum_j A_{ij}$ and $2m \equiv \sum_{ij} A_{ij}$. Note that in the case of binary networks, A_{ij} represented the presence or absence of an edge between nodes i and j, d_i represented the degree of node i and m represented the total number of edges in the networks. Since we are interested in financial networks, a naive approach would be to use the empirical correlation matrix, denoted by $\mathbf{C}$, as the

network's weighted adjacency matrix. Note that we are essentially ignoring the results of section 5.1.1 for this naive approach.

For now, though, let us use the following relationship

$$A_{ij} = \frac{1}{2}\big(C_{ij} + 1\big) - \delta(i,j) \tag{5.9}$$

where C_{ij} denotes the elements of the correlation matrix, and the $\delta(i,j)$ term removes self edges. From this definition, we simply note that $A_{ij} \in [0,1]$.

We now focus on finding the partition, $\boldsymbol{\sigma}$, that maximises the modularity. We notice that a larger value of A_{ij} implies larger correlation between the stocks i and j, whilst a smaller value implies a lower correlation. Recalling how, for the case with binary networks, we sought after denser connections within groups and sparser connections between groups, we realise the modularity maximisation algorithms used on binary networks should have the same effect on financial networks. This is intuitive since, in both types of networks, we aim to find the partition which maximises the sum of correlations or number of edges between nodes within same community and minimises the sum of correlations or number of edges between nodes belonging to different communities.

We shall select two familiar approaches, widely known in the literature, as algorithms for the modularity maximisation. We consider a greedy agglomerative method discussed earlier in the report in addition to a method that uses spectral relaxation.

Firstly, we use the greedy method of Clauset et al. [13] described in section 3.2.1, which we noted is also applicable in this case of weighted networks, so we use equation (5.9) as input to the algorithm. The algorithm we use is implemented in MATLAB and the code has been written by Le Martelot [67].

The other approach using the spectral relaxation can be summarised using the argument from [68]. Let $\mathbf{B}$ denote the modularity matrix, whose elements are defined by

$$B_{ij} = A_{ij} - \frac{d_i d_j}{2m} \tag{5.10}$$

where A_{ij}, d_i and m are defined as before. Also denote the set of nodes belonging to group a by $\mathcal{S}_a \equiv \{i : \sigma_i = a\}$. The algorithm iterates by attempting to split the node members of a single group in an optimal fashion by using modularity. Assume at one iteration there are q groups so the partition is indexed by $[q] \equiv \{1, 2, \ldots, q\}$. For some $a \in [q]$, let $\mathbf{B}_a$ denote the submatrix restricted to nodes in $\mathcal{S}_a$. Let $\boldsymbol{v} \in \mathbb{R}^{|\mathcal{S}_a|}$ denote the

sign vector given the algorithm operating on $\mathbf{B}_a$. Then the change in modularity is given by

$$\begin{aligned} \Delta Q &= \frac{1}{2m}\left(\sum_{i,j\in\mathcal{S}_a} B_{ij}\frac{(1+v_i v_j)}{2} - \sum_{i,j\in\mathcal{S}_a} B_{ij}\right) \\ &= \frac{1}{4m}\left(\sum_{i,j\in\mathcal{S}_a} B_{ij}v_i v_j - \sum_{i,j\in\mathcal{S}_a} B_{ij}\right) \\ &= \frac{1}{4m}\boldsymbol{v}^T\widetilde{\mathbf{B}}_a\boldsymbol{v} \end{aligned} \tag{5.11}$$

where $\widetilde{\mathbf{B}}_a = \mathbf{B}_a - diag\left(\sum_{j\in\mathcal{S}_a} B_{ij}\right)$.

The algorithm accepts the splitting of the group which maximises the modularity difference, and terminates when reaches a threshold regarding the size of the groups and the possible improvement in modularity at a given iteration. We have implemented this algorithm of [68] using MATLAB, stressing the input is defined by equation (5.9).

5.2.2 Modified Modularity Optimisation Methods

We move on to understand the potential issues with the naive approach hinted in the previous section. Intuitively, the term $d_i d_j/2m$ reflects the null hypothesis that the observed network structure is wholly based on the degrees of the nodes. This idea is sound when applied to binary networks, however, these terms do not have a specific meaning when applied to financial networks since we are interested in the correlation matrix rather than the structure of the underlying graph. The quantities $d_i \equiv \sum_j C_{ij}$ and $2m \equiv \sum_{ij} C_{ij}$ do not make up a clear notion of a null model when combined. Furthermore, MacMahon and Garlaschelli [57] have shown the naive approach is mathematically incorrect and leads to biased results. In order to detect communities within the correlation matrix that leads to correct results, we seek improvement in the construction of the null model for the network. We have already discussed, in section 5.1.1, how to construct a filtered correlation matrix that takes into account the random noise at the level of individual stocks and a market-wide component. Therefore, this filtered matrix, which we have denoted by $\mathbf{C}^{(g)}$ and defined in equation (5.7), can be used as a modularity matrix, since it reflects a suitable null model (i.e. random noise plus market wide correlation) subtracted from the observed correlation matrix.

Our aim, here, is to use $\mathbf{C}^{(g)}$ as input to an algorithm that can detect communities by maximising modularity. We hope this provides better results than both of the naive modularity methods discussed previously.

We must first, though, introduce a new formulation of modularity given a partition, as used by [57], which we denote by $Q_n(\boldsymbol{\sigma})$,

$$Q_n(\boldsymbol{\sigma}) = \frac{1}{C_{norm}} \sum_{ij} C_{ij}^{(g)} \delta\left(\sigma_i, \sigma_j\right). \tag{5.12}$$

where C_{norm} is a normalisation term defined by

$$C_{norm} \equiv \sum_{ij} |C_{ij}| \tag{5.13}$$

which just ensures the value of the newly-defined modularity lies within the interval $[-1, +1]$. The new formulation of modularity is specifically aimed to detect mesoscopic-level communities.

Our first 'modified' modularity method is a spectral clustering algorithm based on the exact same technique considered for binary networks considered in section 3.1, but instead we shall use the filtered correlation matrix, $\mathbf{C}^{(g)}$, as input. We simply find the eigenvectors of $\mathbf{C}^{(g)}$, use them to construct the embedded vectors, which are then clustered using the familiar k-means clustering algorithm (same procedure as with binary networks). We use different values for the number of groups (within an appropriate range), then run the algorithm for each one and choose the partition with the best value for the re-defined modularity defined in equation (5.12).

The second modified modularity method is a greedy algorithm based on the Louvain method proposed by Blondel et al. [23], that we discussed in section 3.2.1. However, we shall instead use the filtered correlation matrix, $\mathbf{C}^{(g)}$, as the input to the algorithm. We use the MATLAB implementation of this algorithm where the code has been written by Mucha and Porter [70]. The intention of utilising this method over the spectral clustering approach just mentioned is twofold. Firstly, we do not need to run the algorithm for various values for the number of groups, since the Louvain method generates the node assignments without requiring the number of communities in advance. Secondly, this algorithm has a faster run time, and this will be important considering performance on time-evolving financial networks, which we shall discuss in chapter 6.

To summarise, we will study four algorithms based on modularity optimisation; two are considered to be naive approach using a weighted adjacency matrix as input with the same null model as with conventional community detection block models, whereas the other two methods are tailored to detect communities in our application of financial networks by optimising the modified definition of modularity using the results from RMT.

5.3 Synthetic Data Testing

Before applying the four algorithms discussed previously to our financial network based on the FTSE 100 data set, we must run tests to confirm we can correctly detect correlated sets of time series in synthetically generated benchmark cases, as also outlined by [57]. We consider a benchmark data set of correlation matrices with 100 time series (i.e. we consider $n = 100$ stocks in the data set) divided into 10 communities of 10 correlated time series. Note the length of the time series is chosen to be $T = 2500$ to reflect similar conditions to the real data set and is also prescribed by RMT (i.e. $T > n$). The set consists of correlation matrices generated with different levels of correlations between the groups and within a group reflecting different signal-to-noise ratios (SNR), where a lower value for SNR results in more difficulty in identifying the ground-truth partition. We set the ground-truth partition, which we denote by $\boldsymbol{\sigma}^*$, to be the same across all correlation matrices. We considered six such benchmarks within the correlation matrix set with varying SNR, and have illustrated two such examples in figure 5.3. We confirmed that, in all the benchmarks, all four methods succeeded in identifying $\boldsymbol{\sigma}^*$ to a sufficiently high accuracy.

5.4 Application to Real-world Financial Network

We now apply the modularity optimisation methods discussed in section 5.2 to the FTSE 100 data collected. We have two clear intentions. Firstly, we wish to understand the community structure of the stocks, in particular with regard to the composition of each community based upon the industry sectors of the stocks. Secondly, we wish to build a comparative analysis of the different algorithms based on empirical data.

We begin by looking at the communities generated by the four modularity methods discussed in section 5.2, using all of the data. Figure 5.4 displays the communities identified by all the algorithms by grouping the tickers of the stocks based upon label colouring.

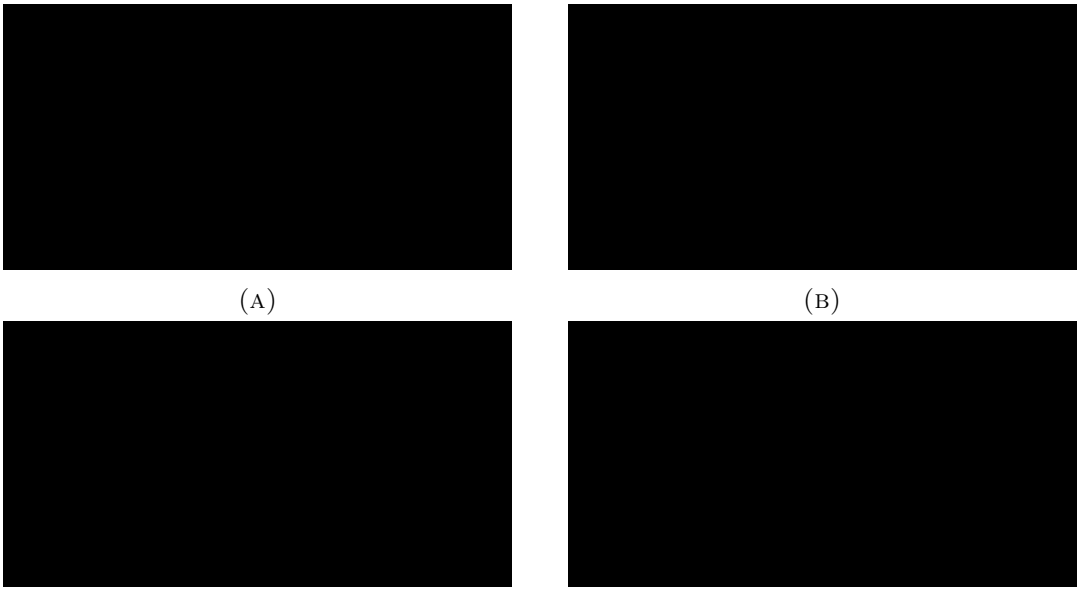
(A) (B)

FIGURE 5.3: Plots of synthetically generated benchmark correlation matrices, one with low SNR (a value of 0.5), (a), in addition to one with higher SNR (a value of 1), (b). These are two examples from a set created from 100 time series divided into 10 communities of 10 correlation time series. The blocks along the diagonal represent cross-correlations between members of the same group, and are thus high (i.e. close to one), whereas off-diagonal blocks represent cross-correlations between time series belonging to different communities reflecting noise (system-wide or additional inter-community correlations).

We see from figures 5.4a and 5.4b that both traditional modularity methods, the greedy algorithm and the spectral relaxation method respectively, have identified two distinct communities; whilst from figures 5.4c and 5.4d, we notice both the modified modularity based techniques have identified four communities. This is the outcome we expected given the assumed null model discounted both random and market-wide correlations [57], meaning the spectral clustering algorithm was successful in finding previously undetected communities.

A first look at the tickers of stocks belonging to each community does not help to uncover any particular patterns. Therefore we analyse the relative composition of each community identified based on the industry sector of each stock as shown in figure 5.5. The industry sectors for each stock we consider is given in appendix A. Each community is represented by a single pie chart (labelled by letter), where the colour legend defined in table 5.1 indicates the industry sector association. We note that the actual sizes of the pie charts do not provide any information, only the fractions of the different coloured regions are important.

A few interesting observations can be made. Certain industry sectors tend to subjugate their respective communities. For example, stocks belonging to the Utilities, Basic Materials and Financials sectors seem to have remained correlated over the time period between 2004 and 2013. But there also examples of sectors that are split amongst different communities. A similar outcome has also been observed in [57] with a data set consisting of stocks from the S&P 500 index. This suggests that subgroups of stocks within a sector are often more correlated with stock from different top-level sectors than from their

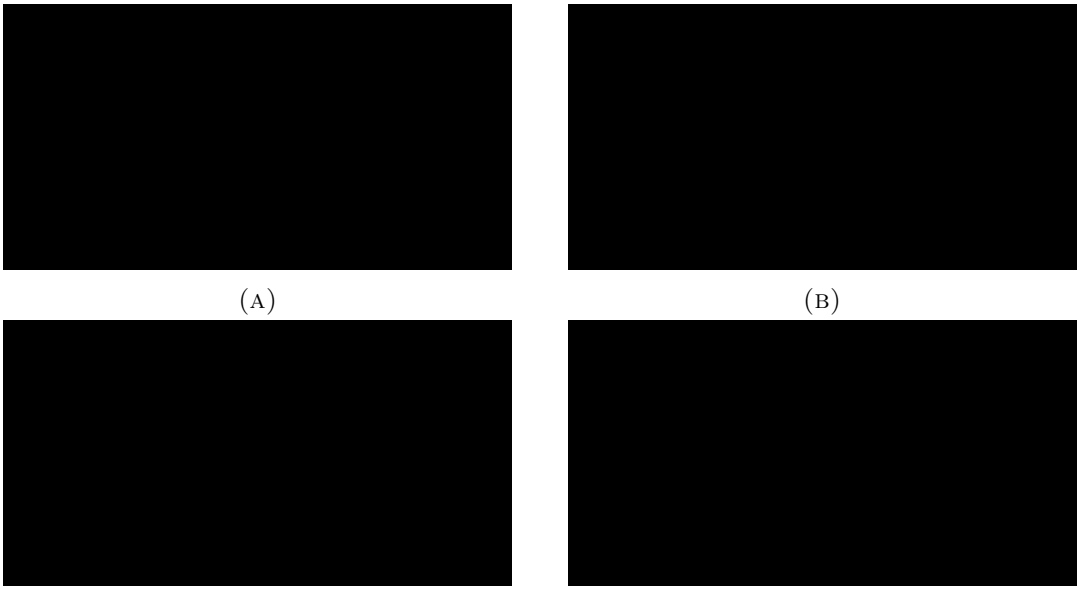


(A) (B)

(C) (D)

FIGURE 5.4: Communities of the FTSE 100 data generated using four different algorithms. The name of each label represents the associated stock's ticker (see appendix A) whilst the colour of the label represents community memberships (i.e. stocks associated with labels that have different colours belong to different communities and stocks whose labels have the same colour belong to the same community). The resulting communities after the greedy algorithm was applied are shown in (a), where there were two communities generated. The resulting communities after the algorithm based on a spectral relaxation was applied are shown in (b), where there were also two communities generated. The resulting communities after the spectral clustering algorithm based on the modified modularity matrix was applied is shown are (c), where there were four communities generated and, finally, resulting communities after the modified modularity Louvain method was applied are shown in (d), where there were also four communities.

TABLE 5.1: Colour representation for 10 industry sectors used to classify FTSE 100 stocks, to be used as a legend.



(A) (B) (C) (D)

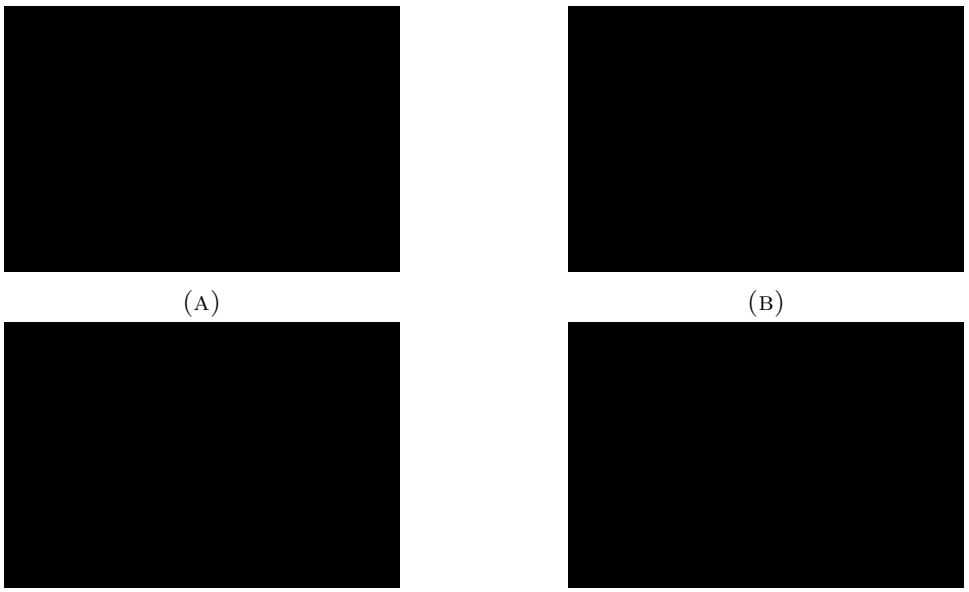
FIGURE 5.5: Pie charts showing the relative composition of each generate community based on the industry sectors of the stocks (see appendix A). Individual communities are labelled by letter, where each community is represented by one pie chart, and the colour legend for all these pie charts is shown in table 5.1. The result for communities generated by the greedy algorithm is shown in (a), whilst the result for communities generated by the spectral relaxation method is shown in (b). The output for communities generated by the modified modularity spectral clustering method is show in (c), whilst the result for communities generated by the Louvain method is shown in (d).

own. Unfortunately, we were not able to obtain sub-classifications for industry sectors due to lack of information available, however, we believe it would be beneficial to extract this form of qualitative information and combine with our quantitative approach identify interesting patterns in correlations.

We now introduce the notion of a renormalised filtered correlation between two communities A and B as described in [57]. We consider the renormalised filtered correlation matrix, denoted by $\widetilde{\mathbf{C}}^{(g)}$, as a matrix whose dimension equal the number of communities detected with each element defined by

$$\widetilde{C}_{AB}^{(g)} \equiv \sum_{i \in A} \sum_{j \in B} C_{ij}^{(g)} \tag{5.14}$$

We shall use the renormalised filtered correlation to test whether different communities, generated are mutually less correlated than expected under the null model [57]. Figure 5.6 plots the renormalised filtered correlation matrices for the communities generated by the four different algorithms.



(A) (B)

(C) (D)

FIGURE 5.6: Plots of renormalised filtered correlation matrices for the communities generated by four different algorithms. Each heat map represents the cross-correlations between the communities identified by each algorithm. Each off-diagonal element is calculated as the sum of the correlations between the corresponding pair of communities, whilst the diagonal entries represents the sum of correlations of nodes belonging to the specific community. The filtered correlation matrix for the output of the greedy algorithm is shown in (a), whilst the result for the communities generated by the spectral relaxation method is shown in (b). The output for communities generated by the modified modularity spectral clustering method is show in (c), whilst the result for the communities generated by the Louvain method is shown in (d).

We see that all algorithms identify partitions that, on aggregate, are anti-correlated with one another using the definition of renormalised filtered correlation.

We conclude that introducing the notion of modified modularity has helped to detect more communities at the mesoscopic scale by applying either a spectral clustering algorithm or a greedy method. The properties of these communities are similar, and the partitions are anti-correlated with one another. The main benefit of applying the Louvain method over the spectral clustering approach is faster convergence to its generated node assignments. Overall, as discussed in section 2.3.2, this property is very useful for investors following mean-variance portfolio theory, since it provides a basis for picking stocks belonging to mesoscopic-level communities that are, on average over a specified period of time (decided by the investor and given by selective use of available data), negatively correlated with one another.

Chapter 6

Temporal Evolution of Financial Networks

In this chapter we aim to understand the temporal evolution of a financial system with respect to the evolving correlation structure of the constituting assets. In chapter 5 we considered a static financial network, where the weight of an edge represents the cross correlation between the two time series associated with the nodes connected by the edge, considering the entire time period to construct one financial network. We then applied community detection techniques to uncover groups of stocks from the FTSE 100 that were correlated more than a null hypothesis suggests. An issue with this approach is the static nature of the network, since investors wish to understand the strength of correlations in price movements in order to dynamically manage investment risk in their portfolios [39]. We build on the results given in chapter 5 and investigate community dynamics utilising time-dependent correlation structures with application to the same FTSE 100 data set, complementing the work of [8, 9, 27, 33, 39]. This approach enables us to identify major changes in the underlying financial market, with the same ideas applicable to other financial markets and asset classes (by use of other available data sets), underlining the potential utility of the techniques considered. We believe this setting of time-varying (weighted) network structure is an important application in regard to the process of adapting existing community detection algorithms to moderately-sized, real-world dynamic networks.

6.1 Constructing Time-evolving Financial Networks

Once more we consider the same FTSE 100 data set used earlier in the report, but we shift away from using the data of the whole period to construct one single network. Instead, we examine the data for several overlapping time windows that collectively cover the whole time period. We generate one financial network for each time window in the following way, as also considered by [8, 9, 27, 33, 39]. Recall from section 2.3.3 each node in the network (i.e. each asset) is associated with a single time series consisting of the daily logarithmic return. We now let the number of time steps considered, T, equal the length of each time window rather than the length of the whole data set time period. Proceeding to create a correlation matrix based upon the standardised time series, as before, we have developed one correlation matrix for each time window. As previously, each cross-correlation (i.e. entry in the correlation matrix) between any two time series is the weight of the edge connecting the nodes associated with the time series in the network. We have created a sequence of financial networks by rolling the time windows of returns through the whole data set, with each network describing the underlying correlation structure of the market at a particular time interval. By shifting the time window there is an overlap in data in the consecutive windows, but this allows us to track the evolution of correlations and identify changes in the behaviour of the market at many different times during the whole time period [39]. This is particularly consequential given that our data set (01/01/2004 - 11/11/2013) includes some of the most significant periods for the state of developed economics worldwide in modern times.

Immediately, though, there is the requirement of deciding upon the values for both parameters, the window length and the length of the overlap (i.e. the duration of time any one window has data overlapped with the preceding time window). Any particular choice of T is a compromise between overly noisy and overly smoothed correlation coefficients [8, 39]. We study the distribution of correlation coefficients for different choices of the parameters to decide on appropriate values. For instance, figure 6.1 compares the distribution of the correlation coefficients for three different values of the time window length, $T = 100$, 150 and 200 (days), whilst fixing the overlap period to 1 (day). Here we plot the mean, variance and skewness to characterise the distribution and realise these signals are quite noisy.

Whereas, figure 6.2 compares the distribution of the correlation coefficients for three different values of the time window length, $T = 100$, 150 and 200 (days), whilst fixing

FIGURE 6.1: Plots characterising the distribution of correlation coefficients over time for a fixed roll-over period of 1 day and varying window lengths, $T = 100$, 150 and 200. The top graph plots the mean value, the centre graph plots the variance whilst the bottom graph plots the skewness all against the whole time period for the FTSE 100 data set.

FIGURE 6.2: Plots characterising the distribution of correlation coefficients over time for a fixed roll-over period of 10 days and varying window lengths, $T = 100$, 150 and 200. The top graph plots the mean value, the centre graph plots the variance whilst the bottom graph plots the skewness all against the whole time period for the FTSE 100 data set.

the overlap period to 10 (days). Once more we plot the mean, variance and skewness to characterise the distribution, but now realise these signals are smoother.

We decide to trade-off the over-smoothing of shorter window lengths with the longer overlap period, and proceed with $T = 100$ and a value of overlap period equal to 10 days. These choices results, for our particular data set, in 240 correlation matrices (and hence financial networks) spanning the period between 01/01/2004 and 11/11/2013. This completes the construction of time evolving financial networks, which we can now use as an empirical test basis for our proceeding analysis of a community detection algorithm in dynamic networks.

6.2 Temporal Evolution of Asset Correlations

We have seen in section 5.1.1, how the eigenvalue spectrum of the empirical correlation matrix generated by the aggregate set indicated correlations not compatible with the null model of a combination of random and market-wide components. This result became

a justification for seeking mesoscopic level communities by discovering patterns in the correlation structure of the financial network, and was a building block of our modified modularity approach. We now wish to investigate the temporal evolution of the correlations between the stocks by analysing the behaviour of the respective correlation matrix eigenvalues and eigenvectors at each time window. Having previously used tools from RMT, we now utilise a closely linked and widely used technique, known as principal component analysis (PCA) [39]. We mentioned PCA earlier in the report in regard to the detection of hidden communities generated by the hidden clique model, although, now we delve into the technical details of the procedure. PCA is a statistical technique that uses an orthogonal transformation to represent the covariance structure of a set of variables through a smaller number of linear combinations of these variables. [6, 39, 79].

We shall consider the following derivation based on [6, 11, 33, 39, 79]. Firstly, recall T represents the time window length and n represents the number of assets in the network (these values equal to 100 and 80, respectively, in our specific empirical example). Let us denote the matrix with entries consisting of the standardised logarithmic returns, for any one particular time window, by a $n \times T$ matrix $\mathbf{X}$. The empirical correlation matrix, also equal to the covariance matrix of $\mathbf{X}$, is denoted by $\mathbf{C}$ and defined by

$$\mathbf{C} = \frac{1}{T}\mathbf{X}\mathbf{X}^T \tag{6.1}$$

where each element C_{ij} represents the cross-correlation between time series i and j. The matrix $\mathbf{X}$ constitutes the set of observed (original) variables, and the PCA method seeks to find a linear transformation, denoted by the matrix, $\mathbf{\Omega}$, that maps $\mathbf{X}$ into a set of uncorrelated variables given by $\mathbf{Y}$. $\mathbf{Y}$ is thus an $n \times T$ matrix defined by

$$\mathbf{Y} = \mathbf{\Omega}\mathbf{X} \tag{6.2}$$

where each row $\boldsymbol{y}_i$ (for $i = 1, \ldots, n$) corresponds to a principal component (PC) of $\mathbf{X}$. The first row of the matrix $\mathbf{\Omega}$, denoted by $\boldsymbol{\omega}_1$, is selected so the first PC, $\boldsymbol{y}_1$, is aligned with the direction of maximal variance in the n-dimensional space defined by $\mathbf{X}$. Each subsequent PC accounts for the maximal variance of $\mathbf{X}$ subject to the condition that the vectors $\boldsymbol{\omega}_j$ are mutually orthonormal. This implies that

$$\boldsymbol{\omega}_j \boldsymbol{\omega}_k^T = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

for all $j, k = 1, \ldots, n$. The correlation matrix is an $n \times n$ symmetric and diagonalisable matrix, which can be written as

$$\mathbf{C} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T \tag{6.4}$$

where $\mathbf{\Phi}$ is an orthogonal matrix of its eigenvectors and $\mathbf{\Lambda}$ is diagonal matrix consisting of the associated eigenvalues. From the result that the eigenvectors of the covariance matrix correspond to the directions of maximal variance, $\mathbf{\Omega} = \mathbf{\Phi}^T$, and thus we can determine the PCs using the decomposition given by equation (6.4) [6].

Section 5.1.1 compares the eigenvalue spectra of an empirical correlation matrix (using our entire data set) against a correlation matrix created from n random time series of length T in the limiting case. The analysis found significant features of the spectra of the empirical correlation matrix. Most of the eigenvalues were contained in a region explained as random noise and given by the Sengupta-Mitra distribution. However, a selection of eigenvalues lay outside this region, as illustrated in figure 5.2, suggesting some form of correlation structure between the microscopic and macroscopic exists. We crucially note that the condition of $T/n > 1$ must be satisfied for the result to hold, but only requires appropriate selection of the parameter T. We have achieved in satisfying this constraint and thus the result from RMT applies to all of our time-windowed financial networks as well.

We can now combine the results from PCA and RMT to study the temporal evolution of correlation structure, considering the approaches from [11, 33, 39]. We denote the covariance matrix for the PC matrix $\mathbf{Y}$ by $\mathbf{\Sigma}$, which is defined as

$$\mathbf{\Sigma} = \frac{1}{T}\mathbf{Y}\mathbf{Y}^T \tag{6.5}$$

Using equations (6.1), (6.2) and (6.4), we can re-write $\mathbf{\Sigma}$ as

$$\mathbf{\Sigma} = \frac{1}{T}\mathbf{\Omega}\mathbf{X}\mathbf{X}^T\mathbf{\Omega}^T = \mathbf{\Omega}\mathbf{C}\mathbf{\Omega}^T = \mathbf{\Phi}^T\mathbf{C}\mathbf{\Phi} = \mathbf{\Lambda} \tag{6.6}$$

We wish to find the total variance in the logarithmic returns for all assets. Let us denote the i-th entry along the diagonal of $\mathbf{\Lambda}$ by λ_i (i.e. $\lambda_i = \Lambda_{ii}$). Then the total variance for $\mathbf{X}$ is given by

$$tr(\mathbf{C}) = \sum_{i=1}^{n} \lambda_i = tr(\mathbf{\Lambda}) = n \tag{6.7}$$

FIGURE 6.3: Plots of the contribution of the top 5 principal components to the total variance in returns against time. Each data point corresponds to the term λ_i/n, defined in equation (6.8), for $i = 1, \ldots, 5$ and a particular time window.

Therefore the proportion of the total variance in $\mathbf{X}$ given by the i-th PC is given by

$$\frac{\Sigma_{ii}}{tr(\mathbf{C})} = \frac{\lambda_i}{n} \tag{6.8}$$

We can now analyse the time-varying nature of the proportion of variance of returns explained by certain PCs.

Figure 6.3 shows the proportion of variance of returns explained by the top 5 PCs for each time window, so we plot λ_i/n for $i = 1, \ldots, 5$ and every time-windowed financial network. The fraction of the variance explained by the first PC increased between 2005 and towards the end of 2006, then dipped for a few months through towards the middle of 2007. There was a sharp rise beginning at the middle of 2007, around the time when the United States subprime mortgage industry collapsed [63]. Also several central banks stepped in with lending to the interbank lending market in August 2007 in order to prevent a liquidity crisis [60, 63]. The bursting of the United States housing bubble had a major impact on the health of major worldwide financial institutions due to their massive exposure to mortgage-backed securities on their balance sheets [60, 63]. For instance, Merrill Lynch was taken over by Bank of America and Lehman Brother filed for bankruptcy on 15/09/2008, and we see the proportion of variance of returns contributed by the first PC increased from this date onwards towards 2011, a significant period of time during the recession [39, 63]. The large contribution in variance of returns by only one principal component indicates a significant amount of common variation in the market (i.e. in the FTSE 100 exchange in our case) [39]. In other words, the market, as a whole, became more correlated during these years, in particular a very distinct reaction to a major crisis to a few international financial institutions. This information, implied by figure 6.3, relates to the intuition that the performance of stocks should decline, as a whole, during a financial crisis, so the returns will be more positively correlated during this period. In addition, in 2004, the top twelve PCs contributed to 64.12% of the variance

of returns, whilst, in 2009, just the top six PCs contributed to 62.82%. The fact that fewer components are required to account for a similar amount of variation in returns, in 2009, compared to five years earlier suggests an increase in the common variation between stocks in the exchange. Moreover, it also implies the correlation structure of this marker can be explained by many fewer factors than n sets of asset time series. The last observation supports the eigenvalue spectra analysis of section 5.1.1.

6.3 Generalised Modularity Optimisation

We have seen in previous chapters how modularity optimisation algorithms can be used to partition static networks (for both unweighed and weighted cases) to uncover community structure. Rather, we now we seek to detect communities in a time-dependent weighted network, for instance a network created by using the method outlined in section 6.1. The previous setting for modularity optimisation is not sufficient for this regime, and thus we require a more general framework. Mucha et al. [35] have developed a framework that enables us to study the community structure of *multislice networks* by using a multislice modularity function. Multislice networks are essentially combinations of networks (slices) coupled through links connecting each node of one slice to itself in other slices [35]. This is a general setting that includes time-evolving networks and is thus suitable for our application. The idea considered by [35] is to derive a generalisation of community detection to multislice networks by formulating an appropriate null model derived from Laplacian dynamics, a concept introduced by Lambiotte et al. [24]. We shall provide a summary of the derivations available from [24, 32, 35] that highlights the approach. Then we will discuss one algorithm to optimise generalised modularity in time-dependent networks known as the *generalised Louvain method*, outlined also in [35, 51].

6.3.1 Laplacian Dynamics and Stability

We first consider a quality function, called *stability*, developed by [24]. One idea motivating stability is that the flow of probability on a network should be trapped in a community for a long period of time [32]. Let $\mathcal{M}(t)$ be an ergodic Markov process on the network. Due to the ergodicity property, any initial configuration will eventually reach the same steady state distribution. Suppose the process $\mathcal{M}(t)$ is the motion of a random walker on the network and consider a partition of the network given by $\mathcal{P} = \{G_1, \ldots, G_k\}$, where

G_i represents the i-th community. We define the stability, denoted by $R_{\mathcal{M}(t)}$, by

$$R_{\mathcal{M}(t)} = \sum_{C \in \mathcal{P}} P(C,t) - P(C,\infty) \tag{6.9}$$

where $P(C,t)$ is the probability of the walker being within the community C both initially and at the time t. This definition arises from the autocovariance function of a Markov process on the network [25]. Intuitively, stability measures the quality of a partition based on how persistent it is [25].

Lambiotte et al. [24] consider normalised Laplacian dynamics to model an unbiased discrete-time random walker on the network, where the probability of moving from one node to another is proportional to the weight of the edge which connects them. Therefore,

$$p_i(t+1) = \sum_j \frac{A_{ij}}{d_j} p_j(t) \tag{6.10}$$

where $p_i(t)$ represents the density of a random walker on node i at time t, A_{ij} is the (i,j)-th element of the network's adjacency matrix and $d_j = \sum_i A_{ij}$ is the strength of node i. In order to find the stationary distribution, we solve the following equation

$$p_i^* = \sum_j \frac{A_{ij}}{d_j} p_j^* \tag{6.11}$$

which results in the steady state distribution given by $p_i^* = d_i/2m$, where $2m = \sum_i d_i = \sum_{ij} A_{ij}$. Notice that this is the same null model considered for our derivation of the modularity function. Proceeding to find the stability at time $t = 1$,

$$R(1) = \sum_{C \in \mathcal{P}} \sum_{i,j \in C} \left(\frac{A_{ij}}{d_j} \frac{d_j}{2m} - \frac{d_i}{2m} \frac{d_j}{2m} \right) = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{d_i d_j}{2m} \right) \delta(g_i, g_j) \tag{6.12}$$

which is equivalent to our formulation of modularity, Q, considered for undirected networks!

In order to encompass time as a parameter within the formulation, we consider a continuous-time process associated with the random walker. Consider independent and identical homogeneous Poisson processes defined on each node of the graph [24], so that the walkers moving at a constant rate from each node, then

$$\dot{p_i} = \sum_j \frac{A_{ij}}{d_j} p_j - p_i \tag{6.13}$$

This equation is driven by the operator $A_{ij}/d_j - \delta_{ij} = -L_{ij}$, where we denote $\mathbf{L}$ as the normalised Laplacian matrix we defined in definition 2.14 (notice a change in notation is made for convenience) [35]. The stationary distribution of this continuous-time process is the same as with the discrete-time process, $p_i^* = d_i/2m$, Furthermore, the stability of the partition is given by

$$R(t) = \sum_{C\in\mathcal{P}} \sum_{i,j\in C} \left(e^{t\mathbf{L}} p_j^* - p_i^* p_j^*\right) = \sum_{C\in\mathcal{P}} \sum_{i,j\in C} \left(e^{t\mathbf{L}} \frac{d_j}{2m} - \frac{d_i}{2m}\frac{d_j}{2m}\right) \tag{6.14}$$

By expanding the matrix exponential in equation (6.14) to the first order in t [24], so that $\left(e^{t\mathbf{L}}\right)_{ij} \approx \delta_{ij} + tL_{ij}$, and ignoring the δ_{ij} factors (since they always contribute to the sum and are thus irrelevant for identifying the optimal partition) the equation becomes

$$R(t) \approx \frac{1}{2m} \sum_{ij} \left(tA_{ij} - \frac{d_i d_j}{2m}\right) \delta(g_i, g_j) \tag{6.15}$$

By dividing by t, which does not affect the optimality of a partition given t, and defining a resolution parameter, $\gamma = 1/t$, we yield the following quality function

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \gamma\frac{d_i d_j}{2m}\right) \delta(g_i, g_j) \tag{6.16}$$

6.3.2 Stability for Dynamic Networks

So far we have focused the stability function for single, static networks. We now consider the approach Mucha et al. [35] used to extend this model to develop a modularity function for multislice networks. For our application, we may only think of the case of *dynamic networks*, where the nodes or edges change over time. An example for the most general case is illustrated in figure 6.4, which we have obtained from [73].

This approach requires a generalisation of the stability function. Replace the $p_i^* p_j^*$ independent contribution from equation (6.15) with a conditional independent contribution $\rho_{i\,|\,j} p_j^*$, where $\rho_{i\,|\,j}$ is the conditional probability at stationarity of jumping from node j to node i along the edge type allowed in the network (in our case, this will be edges between nodes in a given slice or between the same node in different slices). By considering the linearisation of the exponential, we can write the stability of a partition as

$$R(t) = \sum_{ij} \left(tL_{ij} p_j^* - \rho_{i\,|\,j} p_j^*\right) \delta(g_i, g_j) \tag{6.17}$$

FIGURE 6.4: Illustration of an example multislice network with three slices. We can differentiate between the *inter-slice* connections represented by C_{jrs} and the *intra-slice* connections represented by A_{ijs}. The couplings are only between adjacent slices, as would be appropriate for ordered slices. We have obtained this illustration from [73].

We can represent each slice, s, of a multislice network by adjacencies A_{ijs} between nodes i and j with the inter-slice couplings, denoted by C_{jrs}, representing the connection between node j in slice r to itself in slice s. Let us define the notation, $d_{js} = \sum_i A_{ijs}$ and $c_{js} = \sum_r C_{jsr}$, whilst defining the multislice strength of node j in slice s by $\kappa_{js} = k_{js} + c_{js}$. We can now re-write equation (6.13) to give the continuous-time Laplacian dynamics equation

$$\dot{p_{is}} = \sum_{jr} \left(A_{ijs}\delta_{sr} + \delta_{ij}C_{jsr}\right) \frac{p_{jr}}{\kappa_{jr}} - p_{is} \tag{6.18}$$

The steady state distribution becomes $p^*_{jr} = \kappa_{jr}/2\mu$, where $2\mu = \sum_{jr} \kappa_{jr}$. Notice that the $\rho_{is\,|\,jr}$ term accounts for two contributions, one from the intra-slice step where $s = r$, and the other from the inter-slice step where $i = j$. The intra-slice step contribution is given by $(k_{is}/2m_s)\,(k_{js}/\kappa_{js})$, whilst the inter-slice step contribution is given by $(C_{isr}/c_{ir})\,(c_{ir}/\kappa_{ir})$, where $m_s = \sum_i k_{is}$. We can now write the stationary distribution as

$$\rho_{is\,|\,jr}p^*_{jr} = \left(\frac{k_{is}}{2m_s}\frac{k_{jr}}{\kappa_{jr}}\delta_{sr} + \frac{C_{jsr}}{c_{jr}}\frac{c_{jr}}{\kappa_{jr}}\delta_{ij}\right)\frac{\kappa_{jr}}{2\mu} = \frac{1}{2\mu}\left(\frac{k_{is}k_{jr}}{2m_s}\delta_{sr} + C_{jsr}\delta_{ij}\right) \tag{6.19}$$

Substituting equation (6.19) into equation (6.17) and again considering a linear approximation provides the following expression

$$Q = \frac{1}{2\mu}\sum_{ijsr}\left[\left(A_{ijs} - \gamma\frac{k_{is}k_{js}}{2m_s}\right)\delta_{sr} + (1-\gamma)\delta_{ij}C_{jsr}\right]\delta(g_{is}, g_{jr}) \tag{6.20}$$

We can also reweight the conditional probabilities which will allow for more flexible resolution parameters, where we have a different resolution parameter for each slice. This gives our final expression for the multislice generalisation of modularity

$$Q = \frac{1}{2\mu}\sum_{ijsr}\left[\left(A_{ijs} - \gamma_s\frac{k_{is}k_{js}}{2m_s}\right)\delta_{sr} + \delta_{ij}C_{jsr}\right]\delta(g_{is}, g_{jr}) \tag{6.21}$$

where the resolution parameters for the inter-slice couplings have been absorbed into the magnitude of the elements of C_{jsr}. As in [35], we assume, for simplicity, $C_{jsr} \in \{0, \omega\}$ indicating the presence (ω) or absence (0) of inter-slice edges. Therefore, only the variable ω is required to adjust the couplings between communities in different network slices [51].

6.3.3 Generalised Louvain Method

We have seen the derivation for the generalised modularity function which we can use to partition dynamic networks, such as time-evolving financial networks. However, as we have seen with standard modularity optimisation, it is NP-hard to find the global optimum partition, thus we seek approximation methods which give near-optimal solutions. One such algorithm is the generalised Louvain method, developed by Mucha et al. [35]. As the name suggests, it uses a procedure similar to the Louvain method we have discussed throughout the report, but works for dynamic networks also. This algorithm has been implemented by Mucha et al. in MATLAB, and the code is available from [70]. We remark that the decision of the parameters γ and ω are important and do alter the communities generated, and intelligent choice of such parameters depending on the specific application is required to provide 'good' partitions.

This algorithm has been applied to real-world networks previously. For instance [35] detected communities in a time-dependent networks constructed from roll call voting in the Unites States Senate. In [50], the technique was applied to a network based on geographic and social information as well as a computationally-expensive application on an image segmentation problem. More recently, Bassett et al. [51], analysed empirical neuroscience data in the form of brain and behavioural networks.

6.4 Synthetic Data Testing

Similarly for the case of static networks in section 5.3, we wish to test the algorithm on synthetic data before applying to our real-world data. We consider a benchmark data set of correlation matrices with 100 time series, each with 2500 data points, divided into 10 communities of 10 correlated time series. The set consists of correlation matrices generated with different levels of correlations between the groups and within a group reflecting different signal-to-noise ratios (SNR), where a lower value for SNR results in more difficulty in identifying the ground-truth partition. For this synthetic data set, we consider four different values (0.5, 0.6, 0.7, 0.8). We can then use this data to construct

time-evolving financial networks, by applying the rolling-window procedure outlined in section 6.1, where we choose the window length to be $T = 100$ and the overlap period to be equal to 10. This generates 240 correlation matrices that combined represent the temporal evolution of the correlation structure forming our synthetic data set. We then utilise the modified modularity techniques discussed in chapter 5 to construct the modified modularity matrix from each network slice (i.e. from each of the 240 correlation matrices) to finally generate our time-evolving modified modularity matrices, which we will be used as input to the generalised Louvain method.

We wish to use benchmark data to find good values for the parameters of the generalised Louvain method. Since the computation time for this algorithm is rather long, we choose to simply let $\gamma_s = 1$ for all s. Choosing the best parameter value for ω is trickier, and thus we decide to consider four different values, $\omega \in \{0.25, 0.5, 0.75, 1\}$. Table 6.1 shows the modularities obtained by applying the generalised Louvain method with the different values of ω and $\gamma = 1$ to the synthetic data set of different SNRs. The first, and obvious, observation is that the modularity values obtained increase, on the whole, as the SNR increases, matching the intuition that the community detection problem is simpler as we increase the inter-community correlations and decrease the intra-community correlations. A more insightful observation is that the modularities for the algorithm with $\omega = 0.75$ and $\omega = 1$ are the best across the varying SNRs. This suggests either one of these two values for the parameter would be appropriate.

TABLE 6.1: Modularities obtained for the generalised Louvain method with different values of ω and common parameter $\gamma = 1$ obtained from applying the methods to synthetic data generated with different SNR values.

SNR	**Generalised Louvain Method Modularities**			
	$\boldsymbol{\omega = 0.25}$	$\boldsymbol{\omega = 0.50}$	$\boldsymbol{\omega = 0.75}$	$\boldsymbol{\omega = 1.0}$
0.5	0.586862	0.596196	0.602225	0.602948
0.6	0.615603	0.624678	0.631786	0.629889
0.7	0.650051	0.658629	0.665545	0.660805
0.8	0.659997	0.667846	0.676168	0.668351

In order to analyse the performance with respect to different values of ω from a different perspective, and to further back up the observations made from table 6.1, we compare the community partitions using the *normalised variation of information*, which we denote by V [12, 33]. The *entropy* of a partition $\mathcal{P}$ of graph, consisting of n nodes, into k communities, $\{G_1, \ldots, G_k\}$, is defined as

$$S(\mathcal{P}) = -\sum_{i=1}^{k} p(i) \log\left(p(i)\right) \tag{6.22}$$

where $p(i) = |G_i|/n$ is the probability of a randomly selected node belonging to community i [12, 33]. Intuitively, entropy for a partition indicates the uncertainty in the community membership of a node [33]. The *mutual information* between two partitions, $\mathcal{P}$ and $\mathcal{P}'$ is defined as

$$I(\mathcal{P}, \mathcal{P}') = \sum_{i=1}^{k} \sum_{j=1}^{k'} p(i,j) \log \left(\frac{p(i,j)}{p(i)p(j)} \right) \tag{6.23}$$

where we assume the partition $\mathcal{P}'$ groups the nodes into k' communities and $p(i,j) = |G_i \cap G_j|/n$. The mutual information is the amount by which the knowledge of a node community membership in $\mathcal{P}$ reduces the uncertainty of its community membership in $\mathcal{P}'$ [33]. We can now define the normalised variation of information between $\mathcal{P}$ and $\mathcal{P}'$ as

$$V(\mathcal{P}, \mathcal{P}') = \frac{S(\mathcal{P}) + S(\mathcal{P}') - 2I(\mathcal{P}, \mathcal{P}')}{\log(n)} \tag{6.24}$$

The division by $\log(n)$ enables $V(\mathcal{P}, \mathcal{P}') \in [0, 1]$ (when comparing data sets with the same size), with a value of 0 indicating identical partitions and a value of 1 indicating that all nodes are in individual communities in one partition and in a single community in the other [33]. As [33] explains, the normalised variation of information is a useful measure for quantifying the difference between partitions, since it is metric on the space of community assignments and satisfies the triangle inequality. Therefore, if two partitions of a network are close to a third partition, they cannot differ too much from one another [33]. Hence, we can compare the normalised variation of information between each of the partitions generated by the generalised Louvain method (for the same prescription of parameters as described previously) with the ground truth partition that we used to generate the synthetic data. Note that lower values indicate the generated partition is close to the ground-truth node assignments and thus performed better. In order to establish a comparison between the generalised Louvain method and just applying the Louvain method for each network slice individually, we run the modified modularity Louvain method also (note this essentially corresponds to the special case of $\omega = 0$ of the generalised Louvain method, but the computation time to run the algorithm is reduced). In order to present a comparison, we average the normalise variation of information values for all five methods over all synthetic data sets (i.e. across all SNR values) in order to provide a measure of performance across varying difficulties of the problem. The results are plotted in figure 6.5. We notice from figure 6.5 that, once more, the best choice seems to be $\omega = 0.75$ or $\omega = 1$.

We therefore conclude from the synthetic testing that we shall use the generalised Louvain

FIGURE 6.5: Plots of the normalised variation of information obtained by applying the generalised Louvain method (denoted as GLM in the key) with different parameters and comparing the output partitions with the ground-truth node assignments used to generate the synthetic data. We also apply the modified modularity based Louvain method to individual network slices to provide a baseline comparison. The values for V are averaged over synthetic data sets with varying SNRs, to provide a measure of performance across varying difficulties of the problem.

method, with parameters $\gamma = 1$ and $\omega = 1$, with the modified modularity matrices, at each network slice, passed as input in order to obtain the best partitions possible for the application on our real-world data set. We also noticed that the generalised Louvain method is more computationally expensive that just applying the modified modularity based Louvain method for every network slice independently, as the observation of [50] suggested. Therefore we shall also apply modified modularity based Louvain method to our time-evolving financial networks and compare both the output partitions and run time.

6.5 Community Detection in Real-world Time-evolving Financial Networks

Chapter 7

Conclusion

Appendix A

List of Stocks

TABLE A.1: The ticker symbol, name and industry of the 80 FTSE 100 companies studied. Price data and classifications were obtained from [62, 64, 65].

Ticker	Name	Industry
AAL	Anglo American	Basic Materials
ABF	Associated British Foods	Consumer Goods
ADN	Aberdeen Asset Management	Financials
AGK	Aggreko	Industrials
AHT	Ashtead Group	Industrials
AMEC	AMEC	Basic Materials
ANTO	Antofagasta	Basic Materials
ARM	ARM Holdings	Technology
AV	Aviva	Financials
AZN	AstraZeneca	Health Care
BAB	Babcock International Group	Industrials
BARC	Barclays	Financials
BATS	British American Tobacco	Consumer Goods
BG	BG Group	Basic Materials
BLND	British Land Company	Technology
BLT	BHP Billiton	Basic Materials
BNZL	Bunzl	Industrials
BP	BP	Basic Materials
BRBY	Burberry Group	Consumer Goods
BSY	British Sky Broadcasting Group	Consumer Services
BT-A	BT Group	Technology

CCL	Carnival	Consumer Goods
CNA	Centrica	Utilities
CPI	Capita	Industrials
DGE	Diageo	Consumer Goods
EZJ	easyJet	Consumer Services
GFS	G4S	Industrials
GKN	GKN	Consumer Goods
GSK	GlaxoSmithKline	Health Care
HMSO	Hammerson	Financials
HSBA	HSBC Holdings	Financials
IAG	International Consolidated Airlines Grp	Consumer Services
IHG	InterContinental Hotels Group	Consumer Services
IMT	Imperial Tobacco Group	Consumer Goods
ITRK	Intertek Group	Industrials
ITV	ITV	Consumer Services
JMAT	Johnson Matthey	Basic Materials
KGF	Kingfisher	Consumer Services
LAND	Land Securities Group	Financials
LGEN	Legal & General Group	Financials
LLOY	Lloyds Banking Group	Financials
LSE	London Stock Exchange Group	Financials
MGGT	Meggitt	Industrials
MKS	Marks and Spencer Group	Consumer Services
MRO	Marathon Oil Corporation	Oil & Gas
MRW	Wm. Morrison Supermarkets	Consumer Services
NG	National Grid	Utilities
NXT	NEXT	Consumer Services
OML	Old Mutual	Financials
PRU	Prudential	Financials
PSN	Persimmon	Industrials
PSON	Pearson	Consumer Services
RB	Reckitt Benckiser Group	Consumer Goods
RBS	Royal Bank of Scotland Group	Financials
RDSB	Royal Dutch Shell	Basic Materials
REL	Reed Elsevier	Consumer Services
REX	Rexam	Consumer Goods

RIO	Rio Tinto	Basic Materials
RR	Rolls-Royce Holding	Industrials
RRS	Randgold Resources Limited	Basic Materials
SBRY	J Sainsbury	Consumer Services
SDR	Schroders	Financials
SGE	The Sage Group	Technology
SHP	Shire	Health Care
SMIN	Smiths Group	Industrials
SN	Smith & Nephew	Health Care
SSE	SSE	Utilities
STAN	Standard Chartered	Financials
SVT	Severn Trent	Utilities
TATE	Tate & Lyle	Consumer Goods
TPK	Travis Perkins	Industrials
TSCO	Tesco	Consumer Services
ULVR	Unilever	Consumer Goods
UU	United Utilities Group	Utilities
VOD	Vodafone Group	Technology
WEIR	The Weir Group	Industrials
WMH	William Hill	Consumer Services
WOS	Wolseley	Industrials
WPP	WPP ORD 10P	Consumer Services
WTB	Whitbread	Consumer Services

Bibliography

[1] E. P. Wigner, "Characteristic vectors of bordered matrices with infinite dimensions," in *The Annals of Mathematics, 62*, pp. 548–564, 1955.

[2] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," 1983, `http://leonidzhukov.net/hse/2013/stochmod/papers/KirkpatrickGelattVecchi83.pdf`.

[3] D. G. Luenberger, "Investment science," 1998.

[4] A. M. Sengupta and P. P. Mitra, "Distributions of singular values for some random matrices," *Phys. Rev. E*, vol. 60, pp. 3389–3392, 1999.

[5] V. Plerou, P. Gopikrishnan, B. Rosenow, L. A. N. Amaral, and H. E. Stanley, "Universal and non-universal properties of cross-correlations in financial time series," 1999, arXiv:cond-mat/9902283v1.

[6] I. T. Jolliffe, *Principal Component Analysis.* Springer, second ed., 2002.

[7] J.-P. Onnela, "Taxonomy of Financial Assets." `http://jponnela.com/`, 2002.

[8] J.-P. Onnela, A. Chakraborti, K. Kaski, and J. Kertesz, "Dynamic asset trees and portfolio analysis," 2002, arXiv:cond-mat/0208131v1.

[9] J.-P. Onnela, K. Kaski, and J. Kertesz, "Clustering and information in correlation based financial networks," 2003, arXiv:cond-mat/0312682v1.

[10] M. E. J. Newman, "The structure and function of complex networks," 2003, arXiv:cond-mat/0303516v1.

[11] A. Utsugi, K. Ino, and M. Oshikawa, "Random Matrix Theory Analysis of Cross Correlations in Financial Markets," 2003, arXiv:cond-mat/0312643v1.

[12] M. Meila, "Comparing clusterings by the variation of information," in *Learning Theory and Kernel Machines*, pp. 173–187, 2003.

[13] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," 2004, http://www.ece.unm.edu/ifis/papers/community-moore.pdf.

[14] R. Guimera, M. Sales-Pardo, and L. A. N. Amaral, "Modularity from fluctuations in random graphs and complex networks," 2004, http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2441765/pdf/nihms-34161.pdf.

[15] M. Potters, J. Bouchaud, and L. Laloux, "Financial Applications of Random Matrix Theory: Old Laces and New Pieces," 2005, arXiv:physics/0507111v1.

[16] J. Duch and A. Arenas, "Community detection in complex networks using Extremal Optimization," 2005, arXiv:cond-mat/0501368v1.

[17] M. B. Hastings, "Community Detection as an Inference Problem," 2006, arXiv:cond-mat/0604429v1.

[18] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," 2006, arXiv:physics/0605087v3.

[19] M. E. J. Newman, "Modularity and community structure in networks," 2006, arXiv:physics/0602124v1.

[20] U. von Luxburg, "A tutorial on spectral clustering," 2006, http://www.stanford.edu/class/ee378b/papers/luxburg.pdf.

[21] D. Spielman, "Spectral graph theory and its applications," in *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007)*, pp. 29–38, 2007.

[22] L. K. Chan, J. Lakonishok, and B. Swaminathan, "Industry Classifications and Return Comovement," in *Financial Analysts Journal, Vol. 63, No. 6*, 2007.

[23] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," 2008, arXiv:0803.0476v2.

[24] R. Lambiotte, J.-C. Delvenne, and M. Barahona, "Laplacian Dynamics and Multiscale Modular Structure in Networks," 2008, arXiv:0812.1770v3.

[25] J.-C. Delvenne, S. Yaliraki, and M. Barahona, "Stability of graph communities across time scales," 2008, arXiv:0812.1811v4.

[26] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," 2009, http://www.stanford.edu/~montanar/RESEARCH/FILEPAP/mpacs.pdf.

[27] M. J. Bommarito and A. Duran, "Spectral Analysis of Time-Dependent Market-Adjusted Return Correlation Matrix," 2010, http://ssrn.com/abstract=1672897.

[28] A. Maleki, D. L. Donoho, and A. Montanari, "Analysis of Approximate Message Passing Algorithm," 2010, http://www.ece.rice.edu/~mam15/CISS2010.pdf.

[29] S. Fortunato, "Community detection in graphs," 2010, arXiv:0906.0612v2.

[30] J. Leskovec, K. J. Lang, and M. W. Mahoney, "Empirical Comparison of Algorithms for Network Community Detection," 2010, arXiv:1004.3539v1.

[31] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," 2010, arXiv:1008.3926v1.

[32] R. Lambiotte, " Multi-scale Modularity in Complex Networks," 2010, arXiv:1004.4268v1.

[33] D. J. Fenn, M. A. Porter, P. J. Mucha, M. McDonald, S. Williams, N. F. Johnson, and N. S. Jones, "Dynamical Clustering of Exchange Rates," 2010, arXiv:0905.4912v2.

[34] B. H. Good, Y.-A. de Montjoye, and A. Clauset, "The performance of modularity maximization in practical contexts," 2010, arXiv:0910.0165v2.

[35] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela, "Community Structure in Time-Dependent, Multiscale, and Multiplex Networks," 2010, arXiv:0911.1824v3.

[36] A. Montanari, "Graphical Models Concepts in Compressed Sensing," 2011, arXiv:1011.4328.

[37] M. Bayati and A. Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," 2011, arXiv:1001.3448v4.

[38] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborova, "Phase transition in the detection of modules in sparse networks," 2011, arXiv:1102.1182v1.

[39] D. J. Fenn, M. A. Porter, S. Williams, M. McDonald, N. F. Johnson, and N. S. Jones, "Temporal Evolution of Financial Market Correlations," 2011, arXiv:1011.3225v2.

[40] P. D. Meo, E. Ferrara, G. Fiumara, and A. Provetti, "Generalized Louvain Method for Community Detection in Large Networks," 2011, arXiv:1108.1502v2.

[41] D. Kuhn, "The basic theory of interest," *Computational Finance: 422 Lecture Notes*, 2012.

[42] D. Kuhn, "Fixed-income securities," *Computational Finance: 422 Lecture Notes*, 2012.

[43] D. Kuhn, "Mean variance portfolio theory," *Computational Finance: 422 Lecture Notes*, 2012.

[44] D. Spielman, "Spectral graph theory and its applications," in *Combinatorial Scientific Computing. Chapman and Hall / CRC Press*, 2012.

[45] R. R. Nadakuditi and M. E. J. Newman, "Graph spectra and the detectability of community structure in networks," 2012, arXiv:1205.1813v1.

[46] S. Ertekin, H. Hirsh, and C. Rudin, "Learning to Predict the Wisdom of Crowds," 2012, arXiv:1204.3611v1.

[47] E. Mossel, J. Neeman, and A. Sly, "Stochastic Block Models and Reconstruction," 2012, arXiv:1202.1499v4.

[48] J. Yang and J. Leskovec, "Defining and Evaluating Network Communities based on Ground-truth," 2012, arXiv:1205.6233v3.

[49] M. Gomez-Rodriguez, J. Leskovec, and A. Krause, "Inferring Networks of Diffusion and Influence," 2012, arXiv:1006.0234v3.

[50] H. Hu, Y. van Gennip, B. Hunter, M. A. Porter, and A. L. Bertozzi, "Multislice Modularity Optimization in Community Detection and Image Segmentation," 2012, arXiv:1211.7180v1.

[51] D. S. Bassett, M. A. Porter, N. F. Wymbs, S. T. Grafton, J. M. Carlson, and P. J. Mucha, "Robust Detection of Dynamic Community Structure in Networks," 2013, arXiv:1206.4358v2.

[52] E. Mossel, J. Neeman, and A. Sly, "Belief Propagation, Robust Reconstruction, and Optimal Recovery of Block Models," 2013, arXiv:1309.1380v1.

[53] F. Krzakala, C. Moore, E. Mossel, J. Neeman, A. Sly, L. Zdeborova, and P. Zhang, "Spectral redemption: clustering sparse networks," 2013, arXiv:1306.5550v2.

[54] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborova, "Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications," 2013, arXiv:1109.3041v2.

[55] L. Massoulie, "Community detection thresholds and the weak Ramanujan property," 2013, arXiv:1311.3085v1.

[56] E. Mossel, J. Neeman, and A. Sly, "A Proof Of The Block Model Threshold Conjecture," 2013, arXiv:1311.4115v1.

[57] M. MacMahon and D. Garlaschelli, "Unbiased community detection for correlation matrices," 2013, arXiv:1311.1924v1.

[58] Y. Deshpande and A. Montanari, "Finding Hidden Cliques of Size sqrt(N/e) in Nearly Linear Time," 2013, arXiv:1304.7047v1.

[59] J. Barbier, F. Krzakala, and C. Schulke, "Compressed sensing and Approximate Message Passing with spatially-coupled Fourier and Hadamard operators," 2013, arXiv:1312.1740v2.

[60] Z. Bodie, A. Kane, and A. J. Marcus, "Essentials of investments ninth edition," 2013.

[61] "Talk at simons institute for the theory of computing: Finding small structures in large datasets." `http://simons.berkeley.edu/talks/andrea-montanari-2013-11-18`, cited in March 2014.

[62] "Wikipedia: Industry classifications benchmark." `http://en.wikipedia.org/wiki/Industry_Classification_Benchmark`, cited in March 2014.

[63] "Wikipedia: Great recession." `http://en.wikipedia.org/wiki/Great_Recession`, cited in March 2014.

[64] "London stock exchange website." `http://www.londonstockexchange.com/home/homepage.htm/`, cited in March 2014.

[65] "Yahoo finance website." `http://uk.finance.yahoo.com/`, cited in March 2014.

[66] "Quantivity wordpress blog." `http://quantivity.wordpress.com/2011/02/21/why-log-returns/`, cited in March 2014.

[67] "Community detection greedy algorithm code." `http://www.elemartelot.org/index.php/programming/cd-code`, cited in April 2014.

[68] "Andrea montanari webpage article." `http://www.stanford.edu/class/ee378b/hw4sol.pdf`, cited in April 2014.

[69] "C++ implementation of belief propagation algorithm for module detection in networks." `http://mode_net.krzakala.org/`, cited in May 2014.

[70] "Generalised louvain method for community detection in matlab." http://netwiki.amath.unc.edu/GenLouvain/GenLouvain, cited in May 2014.

[71] "Talk at netsci2010: Multislice community detection." http://www.jponnela.com/web_documents/netsci2010_multislice.pdf, cited in May 2014.

[72] "Youtube: Statistical mechanics lecture 9." http://www.youtube.com/watch?v=AT4_S9vQJgc, cited in May 2014.

[73] "Oxford centre for industrial and applied mathematics: Networks webpage." http://www.maths.ox.ac.uk/groups/ociam/research/networks, cited in June 2014.

[74] Y. Deshpande and A. Montanari, "Information-theoretically Optimal Sparse PCA," 2014, arXiv:1402.2238v1.

[75] G. Pruessner, "The canonical ensemble," *Mathematical Physics II Statistical Mechanics (M4A41): Lecture Notes*, 2014.

[76] G. Pruessner, "The ising chain," *Mathematical Physics II Statistical Mechanics (M4A41): Lecture Notes*, 2014.

[77] G. Pruessner, "Mean field theory," *Mathematical Physics II Statistical Mechanics (M4A41): Lecture Notes*, 2014.

[78] D. Gillies, "Introduction to probabilistic graphical models," *Intelligent Data Analysis and Probabilistic Inference: Lecture 12 Notes*, 2014.

[79] D. Gillies, "Principal component analysis," *Intelligent Data Analysis and Probabilistic Inference: Lecture 15 Notes*, 2014.