

8-6-2015

Social Network Community Detection

Guang Ouyang
guang.ouyang@uconn.edu

Follow this and additional works at: <https://opencommons.uconn.edu/dissertations>

Recommended Citation

Ouyang, Guang, "Social Network Community Detection" (2015). *Doctoral Dissertations*. 870.
<https://opencommons.uconn.edu/dissertations/870>

Social Network Community Detection

Guang Ouyang, Ph.D.
University of Connecticut, 2015

ABSTRACT

This dissertation has its main focus on the development of social network community detection algorithms. Real world social networks are usually found to divide naturally into small communities. In the big data age, effective and scalable algorithms detecting network community structures are in demand in a wide range of business applications, such as marketing segmentation, friend recommendation in online social networks, and product recommendation for online retailers such as Amazon. We aim to leverage the power of statistical inference over uncertainty to scalable community detection algorithms. We developed three novel community detection algorithms: the first is a statistical model-based clustering approach, the second is performing optimization on a global objective function, and the third is based on the optimization of a localized objective function. These three algorithms may service for different purposes of applications.

Social Network Community Detection

Guang Ouyang

B.S., Mathematics, Wuhan University, Wuhan, China, 2008

M.S., Mathematics, University of Miami, FL, USA, 2010

A Dissertation
Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy
at the
University of Connecticut

2015

Copyright by

Guang Ouyang

2015

APPROVAL PAGE

Doctor of Philosophy Dissertation

Social Network Community Detection

Presented by

Guang Ouyang, B.S. Mathematics, M.S. Mathematics

Major Advisor

Dipak K. Dey

Associate Advisor

Nitis Mukhopadhyay

Associate Advisor

Haim Bar

University of Connecticut

2015

Acknowledgements

First of all, I would like to thank my wife, Mengxuan Lu, for her companionship and encouragement during this long journey.

I would like to express my sincere appreciation to my Major Advisor, Professor Dipak K. Dey. He offered me lots of freedom on my research and always encouraged me to explore the far-frontiers of unknown areas and try new things. Without his encouragement and guidance, I could not finish my thesis in this exciting interdisciplinary area of between Statistics and Computer Science.

I would also like to thank my Associate Advisor, Professor Nitis Mukhopadhyay. His course of Mathematical Statistics and Statistical Inference was crucial for me to complete this thesis.

Sincere thanks to my Associate Advisor, Professor Haim Bar for his suggestions and guidance on the computational part of my research.

Lastly, thanks to all faculty members, staffs, and students. You made the Statistics Department so great a place to learn and to grow.

Contents

Acknowledgements	iii
1 Introduction	1
2 Statistical Community Detection	11
2.1 Introduction	11
2.2 Erdős-Rényi Random Graph Model	15
2.3 Stochastic Block Model	18
2.3.1 Parameter Estimation	21
2.3.2 Community Prediction	23
2.3.3 Additional Comments	24
2.4 Distance-based Mixed Membership Model	26
2.4.1 Model Description	28
2.4.2 Parameter Estimation	30
2.4.3 Example: Zachary Karate Club Data	34
2.4.4 Examples: Dolphin Network Data	36
2.4.5 Final Comments about Mixed Membership Model	39
3 Clique-based Community Detection	41
3.1 Introduction	41

3.2	Spectral Clustering for Network Data	43
3.3	Modularity Maximization	48
3.3.1	Modularity Score	48
3.3.2	Split the Network Into Two Communities	50
3.3.3	Split the Network Into More Than Two Communities	52
3.3.4	Resolution Limitation of Modularity Maximization Approach	53
3.4	Intuition of Clique-based Network Clustering	56
3.5	Clique-based Clustering Algorithm	58
3.5.1	Split the Network Into Two Communities	60
3.5.2	Split the Network Into More Than Two Communities	63
3.5.3	Summarize Clique-based Algorithm	66
3.6	Simulated Network Example	67
3.6.1	Simulation From Stochastic Block Model	67
3.6.2	Simulated Network 1	70
3.6.3	Simulated Network 2	73
3.6.4	Simulated Network 3	76
4	Localized Community Detection	78
4.1	Introduction	78
4.2	Effects of Threshold Parameter p	79
4.2.1	Karate Club Data Set	80

4.2.2	Dolphin Network Data	83
4.2.3	Simulated Random Network	86
4.3	How to Choose Threshold Parameter p	89
4.3.1	Control the Risk of Splitting an Erdős-Rényi Network	90
4.3.2	Control the Risk of Merging Significant Communities	94
4.4	Localized Clustering Algorithm	96
4.4.1	Modify Global Clustering Algorithm	97
4.4.2	Development of Localized Clustering Algorithm	103
4.5	Examples: Simulated Networks	106
5	Future Work	111
5.1	Conclusion	111
5.2	Music Web Application	114
5.3	Future Work Platform	116
A	Summary Tables	117
A.1	Tables of Chapter 2	117
A.2	Tables of Chapter 4	127
	Bibliography	133

List of Tables

3.1	Reward table for the clique-based clustering with parameter p	60
3.2	Parameters of $SBM1$ for simulation of random network 1.	70
3.3	Apply Algorithm 3.2 with $p = 0.11$ to 100 simulated networks from $SBM1$ with parameters in Table 3.2.	71
3.4	Parameters in $SBM2$ for simulation of random network 2.	73
3.5	Apply p -clique based Algorithm 3.2 with $p = 0.06$ to 100 simulated net- works from $SBM2$ with parameters in Table 3.4.	74
3.6	Parameter in $SBM3$ for simulation of random network 3	76
4.1	Karate Club network clustering summary by clique-based clustering Al- gorithm 3.2 with $p = 0.15$	81
4.2	Karate Club network clustering summary by clique-based clustering Al- gorithm 3.2 with $p = 0.1$	82
4.3	Dolphin network clustering summary by clique-based clustering Algo- rithm 3.2 with $p = 0.03$	85
4.4	Dolphin network clustering summary by clique-based clustering Algo- rithm 3.2 with $p = 0.20$	85
4.5	Parameters of $SBM4$ for simulation of random network 4	86

4.6	<i>NMI</i> estimate and standard error of Clique-based clustering out of 100 random network simulated from <i>SBM4</i> with parameters in Table 4.5.	89
4.7	Apply Algorithm 3.2 with $p = 0.09$ to 100 simulated networks from <i>SBM1</i> with parameters in Table 3.2	95
4.8	Parameters of <i>SBM5</i> for simulation of random network 5.	97
4.9	Apply Algorithm 3.2 with $p = 0.0886$ to 100 simulated networks from <i>SBM5</i> with parameters in Table 4.9.	100
4.10	Apply localized clique-based clustering algorithm to 100 simulated net- works from <i>SBM5</i> with parameters in Table 4.9.	102
4.11	Parameters of <i>SBM6</i>	107
4.12	Cluster size and internal link density for <i>SBM7</i>	109
4.13	Local Clique Index Maximization Clustering Algorithm 4.1	109
A.1	Summary of Posterior Distribution of community membership parameter $Z = (Z_1, \dots, Z_n)$ in the distance-based mixed membership model for the Karate Club data set.	118
A.2	Summary of Posterior Distribution of community membership parameter $Z = (Z_1, \dots, Z_n)$ in the distance-based mixed membership model for the Dolphine Network data set.	121
A.3	Summary of Type 1 error rate of splitting Erdős-Rényi Random Networks using p -clique Index Maximization algorithm 3.2.	128

List of Figures

1.1	Plot of a network with three communities colored in red, blue and green.	2
2.1	Plot of the split of Karate Club Network of Zachary (1977) after a conflict between the two instructors. The members tend to split into two groups following these two instructors.	35
2.2	Plot of the split of the Karate Club Network of Zachary (1977) by Distance-base Mixed Membership Model.	36
2.3	Split of the Dolphin Network of Lusseau et al. (2003) by betweenness-based network clustering algorithm.	37
2.4	Split of the Dolphin Network of Lusseau et al. (2003) by distance-based mixed membership model.	38
3.1	A random network simulated from Erdős-Rényi Random Graph model $G(40, 0.1)$. The maximized Modularity Score $Q = 0.3590$ when the network is splitted into 5 communities represented by 5 colors.	55
3.2	A random network simulated from $SBM1$ with parameters defined in Table 3.2. This network is splitted into three communities by our clique-based clustering algorithm. It is matching with the community structure defined in the stochastic block model. These three communities are displayed in three different colors: red, blue and green.	72

3.3	A random network simulated from <i>SBM2</i> with parameters defined in Table 3.3. This network is splitted into four communities by our clique-based clustering algorithm. It is matching with the community structure defined in the stochastic block model with minor errors. These four communities are displayed in four different colors: red, blue and green and pink.	75
4.1	Clique-based clustering Algorithm 3.2 with $p = 0.15$ is applied to Karate Club network. It splits the network into 4 communities denoted by four colors: yellow, blue, red, and green.	81
4.2	Clique-based clustering Algorithm 3.2 with $p = 0.1$ is applied to Karate Club network. It splits the network into 2 communities denoted by two colors: red and green.	82
4.3	Clique-based clustering Algorithm 3.2 with $p = 0.035$ is applied to Dolphin network. It splits the network into 4 communities denoted by two colors: red, and green.	84
4.4	Clique-based clustering Algorithm 3.2 with $p = 0.2$ is applied to Dolphin network. It splits the network into 6 major communities of size greater than 5.	84
4.5	Clique-based clustering Algorithm 3.2 with $p = 0.02$ applied on a random network simulated from <i>SBM4</i> with parameters in Table 4.5.	87

4.6	Clique-based clustering Algorithm 3.2 with $p = 0.05$ applied on a random network simulated from $SBM4$ with parameters in Table 4.5.	88
4.7	Clique-based clustering Algorithm 3.2 with $p = 0.1$ applied on a random network simulated from $SBM4$ with parameters in Table 4.5.	88
4.8	Clique-based clustering Algorithm 3.2 with $p = 0.0886$ applied on a random network simulated from $SBM5$ with parameters in Table 4.8.	99
4.9	Local Clique-based clustering on random network simulated from $SBM5$ with parameters in Table 4.8.	101
4.10	Localized Clique-based Hierarchical Clustering Scheme	104
4.11	Local Clique-based clustering on random network simulated from $SBM6$ with parameter in Table 4.11.	108

List of Algorithms

- 2.1 Gibbs Sampling Algorithm for Distance-based Mixed Membership Model 33
- 3.1 Community Detection via Modularity Maximization 52
- 3.2 Community Detection via p -clique Index Maximization 64
- 4.1 Community Detection via Local Clique Index Maximization 106

Chapter 1

Introduction

The main topic of this thesis is about social network community detection in the background of big data age. Human beings are generally social animals, and the entire human race forms a giant social network on this planet. In modern times, almost everyone's life relies on this social network. The concept of social network was first introduced by Émile Durkheim and Ferdinand Tönnies on late 1890s. Major developments in this field can be seen in 1930s by several groups in psychology, anthropology, and mathematics working independently.

Social network is a theoretical abstraction, useful in social sciences to study the relationships between individuals, groups, or organizations. Real world social networks are usually found to divide naturally into small communities. Generally speaking, a community in a social network is a set of nodes that are densely connected internally, but loosely connected to the rest of this network. The Figure [1.1](#) is a plot of network

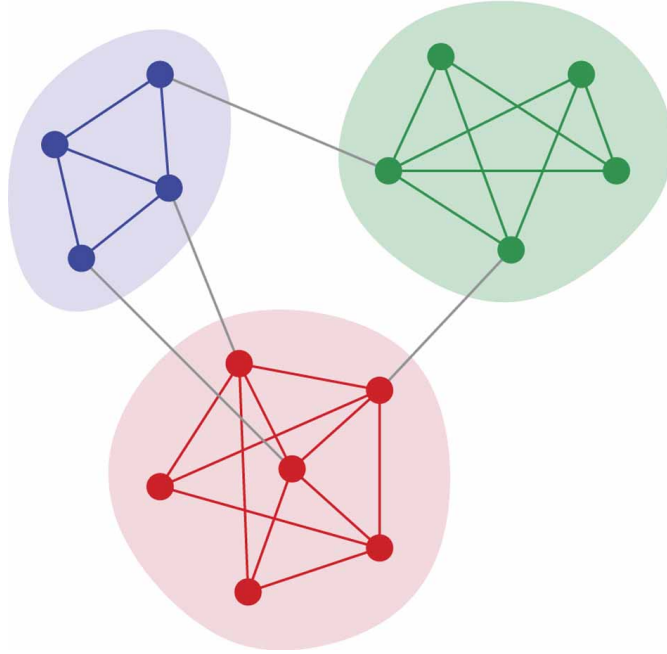


Figure 1.1: Plot of a network with three communities colored in red, blue and green.

with three communities colored in red, blue and green. In this thesis, we explore different methodologies to detect community structures from social networks effectively and efficiently.

Past research on community detection from network data generally divides into two categories: the statistical model-based method, and the optimization-based method. In the former category, we try to model the statistical distribution of relationships between nodes in a social network from the community structure, and any other auxiliary variables as predictors. The community structure can be parametrized as predictors in the statistical model. This category of method is very powerful, because there is a solid theoretical foundation to perform statistical inference on the parameter estimates about the community structure in the social network. In the latter category, we set up an

objective function that quantifies how good is a division to a network, and then optimize this objective function among all possible divisions. An important advantage of this category of methods is efficiency and scalability, which make them very popular in many real world applications.

The first and simplest statistical model of social network is Erdős-Rényi random graph model introduced in [Erdős and Rényi \(1959\)](#). It assumes that the relationship between every pair of two nodes is binary and distributed as $\text{Bernoulli}(p)$, and independent of the existence of an edge between any other pair of two nodes, where the parameter p is called the link density. This can be viewed as the null model without any community structure.

The first statistical model for network community detection was stochastic block model (SBM) by [Snijders and Nowicki \(1997\)](#). In this model, every node i in the network of size n is assumed to have an unobserved single community membership value $c(i)$, simulated from $\text{Multinomial}(\theta)$ distribution, where $i = 1, \dots, n$, and θ is the parameter of multinomial distribution. The distribution of relationships between two nodes i and j from community $c(i)$, and $c(j)$ in a social network is independently distributed as $\text{Bernoulli}(B_{c(i),c(j)})$, where $i = 1, \dots, n$, $j = 1, \dots, n$, $B_{c(i),c(j)}$ is a parameter about the link probability between nodes in community $c(i)$, and $c(j)$. The parameter estimates and inference are obtained from Bayesian method via Gibbs sampling algorithm.

Another important statistical method for network community detection is the latent position model introduced by [Handcock et al. \(2007\)](#). In this model, every node i in a

network of size n is assumed to have a latent social position z_i . The distribution of relationships between two individuals i and j is independently distributed as Bernoulli(p_{ij}), given z_i , z_j , and auxiliary variables $x_{i,j}$, where p_{ij} satisfying:

$$\log(p_{ij}/(1 - p_{ij})) = \beta_0^T x_{i,j} - \beta_1 |z_i - z_j|, \quad (1.1)$$

where z_i is distributed as a mixture of h multivariate normal such that

$$z_i \sim \sum_{k=1}^h \lambda_k \text{MVN}_d(\mu_k, \sigma_k^2 I_d), \quad (1.2)$$

with restriction:

$$\sqrt{\frac{1}{n} \sum_{i=1}^n \|z_i\|_2^2} = 1,$$

where $1 \leq i, j \leq n$, h denotes the number of candidate communities, d denotes the dimension of the latent social position vector z_i . The community membership of any particular individual can be inferred from their social position z_i using similar methods as Mixture Normal clustering algorithm. The parameter estimates and inference of this approach is also done by Bayesian methods with Gibbs sampling algorithm.

In all previous statistical models, we assume every node i in the social network only has a single community membership $c(i)$. However, in real world, a particular individual in a social network may belong to multiple communities with different strength of affiliation. For example, someone may be a member of both a dancing and a rock

climbing club. In the paper of [Airoldi et al. \(2008\)](#), a mixed membership stochastic block model was introduced to accommodate this situation. In this model, every node i , may belong to multiple communities. To be specific, now the community membership $c(i)$ of node i is not a constant, but a Multinomial random variable with distribution:

$$c(i) \sim \text{Multinomial}(\pi_i), \quad (1.3)$$

where

$$\pi_i \sim \text{Dirichlet}(\alpha), \quad (1.4)$$

where α is a hyper-parameter of Dirichlet distribution. The distribution of the relationships between two node i and j in a social network is independently distributed as Bernoulli($p_{c(i) \rightarrow j, c(j) \rightarrow i}$), where $c(i)$, and $c(j)$ are defined as in the expression (1.3) and (1.4), $c(i) \rightarrow j$ denotes the simulated value from distribution $c(i)$, with respect to the relationship with node j , $c(j) \rightarrow i$ denotes the simulated value from distribution $c(j)$, with respect to the relationship with node i . Another thing worth noting is that the Variational EM algorithm is used for parameter estimation, since there are too many variables to simulate and the MCMC sampling approach becomes in-feasible.

By the turning to the 21st century, most studies in social network focus on relatively small group of individuals, since large scale social network data is seldomly available.

Things changed in the big data revolution in recent years, especially after the emergence of online social network platforms, such as Facebook, LinkedIn, Twitter, Pinterest, Google+, etc. Nowadays, obtaining online social network data with millions or even billions of nodes is possible. This imposes a big challenge to statistical modeling of social network approach for community detection purpose. In practice with large scale network dataset, the efficiency and scalability of a community detection algorithm is crucial for its popularity. Due to this reason, two of the most popular network clustering algorithms nowadays are Spectral Clustering method, and Modularity Maximization method.

The main tool for spectral clustering method is the unnormalized graph Laplacian matrix or normalized graph Laplacian matrix. Standard reference for graph Laplacian matrix is [Chung \(1997\)](#). The two most common objective functions are Ratio Cut in [Hagen and Kahng \(1992\)](#) and Normalized cut (Ncut) in [Shi and Malik \(2000\)](#) and in [Ng et al. \(2001\)](#). The Ratio Cut function can be derived from unnormalized graph Laplacian matrix, and the Ncut function can be derived from normalized graph Laplacian matrix. The minimization of Ratio Cut and Ncut can be done by computing a few eigenvectors of the corresponding graph Laplacian matrix with smallest eigenvalues. One disadvantage for spectral clustering in social network is that it usually divides a network into communities of similar size. However, in real world application, community size may vary from very small to very large.

A remedy for this problem in spectral clustering in social network is the Modularity Maximization method introduced in [Newman \(2006\)](#). Like Ratio Cut and Ncut, the

Modularity score is also an objective function that quantifies how good is a division of a network into communities. The maximization of the modularity score over all possible division of the network can be done in a couple of algorithms. One common way is a eigenvector-based method by [Newman \(2006\)](#). Another way of maximizing Modularity score is using simulated annealing, independently described by [Kirkpatrick et al. \(1983\)](#) and [Černý \(1985\)](#). Although widely used in application, the Modularity maximization method is also known for its resolution limitation. It may split large communities, or merge two weakly connected communities as explained in [Fortunato and Barthelemy \(2007\)](#).

In order to evaluate the quality of a clustering algorithm over a network we will use the Normalized Mutual Information (NMI) introduced in [Fred and Jain \(2003\)](#) as a measure to validate our clustering algorithm with the initial cluster assignment in the stochastic block model. The validity of NMI has been confirmed by Information Theory.

Definition 1.1. *The NMI is defined as:*

$$I(T, S) = \frac{-2 \sum_{k=1}^{C_T} \sum_{l=1}^{C_S} N_{kl} \log\left(\frac{N_{kl} N_{..}}{N_{k.} N_{.l}}\right)}{\sum_{k=1}^{C_T} N_{k.} \log\left(\frac{N_{k.}}{N_{..}}\right) + \sum_{l=1}^{C_S} N_{.l} \log\left(\frac{N_{.l}}{N_{..}}\right)}, \quad (1.5)$$

where T stands for ground truth of the community structure, and S stands for the community structure found by our algorithm, C_T is the number of real clusters, C_S is the number of found clusters, N is the $C_T \times C_S$ confusion matrix with entries N_{kl} denotes the number of nodes in real cluster k that appear in found cluster l , $N_{k.}$ is the sum over

row k , $N_{.l}$ is the sum over column l , $N_{..}$ is the sum over all elements of N .

To summarize, the problem of network community detection is far from being solved in this big data age. There are two possible directions for progress. One way is still using current available statistical models, but develop novel algorithms to make them more efficient and scalable to massive network. In [Gopalan and Blei \(2013\)](#), an efficient algorithm was introduced that makes the mixed membership stochastic model in [Airoldi et al. \(2008\)](#) applicable to large network of size up to 3.7 million. The key idea is to use stochastic gradient descent method to perform the optimization of variational objective function. This paper opens the door of using sophisticated statistical model to analyze massive networks. Another way is try to add statistical inference to optimization-based community detection algorithm, such as spectral clustering, and modularity maximization algorithm. For these category of methods, efficient and scalable algorithms are usually available, but they can't tell us if the communities they find are statistically significant. To the best of our knowledge, there is still no big success in this direction, and it is our main interest in this thesis.

In this thesis, we developed a novel statistical model-based community detection algorithms that provides insights between the community structure and the underlying distance function. We also developed two novel optimization-based community detection algorithms. One guarantees internal link density of every community it detects is higher than some user controlled threshold. The other one provides control in the statistical significance of every community it detects, so we know these communities are meaningful,

not formed in random. Both of these two optimization-based community detection algorithm are efficient and scalable to massive networks. Our major passion and incentive for this research is to leverage the power of statistics to the emerging interdisciplinary area of data science, especially the area of large scale social network community detection in this big data age.

The rest of this thesis is organized as follow:

1. In chapter 2, we review the Erdős–Rényi random graph model and stochastic block model (SBM), and developed our novel statistical model for community detection purpose. For our model, we emphasize the relationship between the observed adjacency matrix of a network and the underlying distance function we use for clustering.
2. In chapter 3, we review the algorithms for spectral clustering and modularity maximization and discuss their advantages and disadvantages. We also developed a clique-based community detection algorithm with user-defined global parameter p . It guarantees that every community it detects has an internal link density higher than p .
3. In chapter 4, we explore some limitations of our global clique-based community detection algorithm, and extend it to a localized auto clique-based community detection algorithm. There is no need for a user to choose a threshold. It provides statistical control on the significance in every community it detects.

4. In chapter 5, we summarize results in this thesis, and describe a web application we developed as a platform for future work of this research.

Chapter 2

Statistical Community Detection

2.1 Introduction

A social network is usually a graphical mapping and measuring of relationships and flows between people, groups, organizations, computers, URLs, and other connected information/knowledge entities. The nodes in the network are the people and groups while the links show relationships or flows between the nodes. Social network provides both a visual and a mathematical analysis of human relationships. Management consultants use this methodology with their business clients and call it “Organizational Network Analysis”. Social network analysis also views social relationships in terms of well-established network theory, consisting of nodes, representing individual actors within the network and ties, which represent relationships between the individuals, such as acquaintance, friendship, organizational position, collaboration etc. These networks are often depicted

in a social network diagram, where nodes are represented as points and ties are represented as lines.

A social network can be represented mathematically by its adjacency matrix

$$A = ((a_{ij})), \quad (2.1)$$

in which $a_{ij} = 1$, if node i and node j has a “connection”, and $a_{ij} = 0$, if node i and node j does not has a “connection”. This “connection” is an abstraction of any social ties, such as friendship or collaboration in a social network. Mathematical random graph theory had been applied to the study of social network since the classical paper of [Erdős and Rényi \(1959\)](#). In the Erdős-Rényi model, the tie a_{ij} between any pair of node (i, j) is assumed to be independent and identically distributed Bernoulli with success parameter p .

However, in real world social network, the distribution of social tie a_{ij} are usually not independent of each other. For example, two people with some common friends are much more likely to be friends themselves than two people without common friend. This transitivity effects are properly handled in the Exponential Random Graph model by [Holland and Leinhardt \(1981\)](#). Further more, [Hoff et al. \(2002\)](#) proposed a Latent Space approach to model the adjacency matrix of a social network. This statistical model imposes conditional independence of social ties a_{ij} on their latent state.

Another complexity of real world social network is its community structure, the main

topic of this chapter. For example, in Facebook, students from the same department of a college form a community, in which almost everybody is connected with each other, while students with different education backgrounds are much less likely to be connected. The efforts of detecting such communities help us better understand the internal structure of a network. Network community detection has a wide range of business applications such as marketing segmentation, organizing large computer networks, introducing new friends for people in online social networks, and recommending new products to customers for online retailers, etc.

Detecting community structures from network is essentially a clustering algorithm, but designed specially for network data. So we will use the term network clustering and network community detection interchangeably. For a typical clustering algorithm, such as k-means developed by [MacQueen \(1967\)](#), we apply them to the distance matrix of a collection of data points. In a social network g of n nodes, the distance matrix is simply

$$dist(g) = J^{(n)} - A^{(n)}, \quad (2.2)$$

where $J(n)$ is an n by n matrix with all entries equal to 1, and $A(n)$ is its adjacency matrix. The unique difficulty for network clustering is that the distance matrix $dist(g)$ is usually binary or only consist of a small number of discrete values. For example, the distance or relationship between two Facebook users is usually binary, such as “Friends” or “Not Friends” . It does not provide as much information as a real number distance

value. Because of this, the commonly used clustering algorithm such as k-means or mixture normal model usually work very poorly in a binary or discrete ordinal distance matrix.

This difficulty brings about the necessity of developing clustering algorithms specialized for network data. To the best of our knowledge, current network clustering methods usually fall into two categories. One category is model-based approach, in which a parametric probabilistic model with clustering structure is assumed to generate the observed network data. The algorithm either maximizes the likelihood of the observed data under the model or simulate the posterior distribution of the parameters given the observed data. The clustering results can be represented by the estimation of parameters. Past work of this category includes stochastic block model by [Nowicki and Snijders \(2001\)](#), latent position cluster model by [Handcock et al. \(2007\)](#), mixed membership block model by [Airoldi et al. \(2008\)](#). Another category of approaches specifies an objective function which evaluates the quality of a network clustering, and the algorithm tries to optimize the objective function. Famous examples include normalized spectral clustering by [Ng et al. \(2001\)](#), and modularity method by [Newman \(2006\)](#). The former category of approaches is more expressive. It explains not only the clustering structure, but also the underlying distribution, from which the observed network is formed. The latter category of approaches is more direct and specialized in the clustering problem, and can usually be more computationally effective.

In this chapter, we focus on model based network clustering algorithms. We will first

introduce a new model-based network clustering algorithm, and a new optimization-based network clustering, and explore their relationships. The rest of this chapter is organized as follow: In section 2.2, we describe the Erdős-Rényi model in more details. In section 2.3, we introduce the stochastic block model, and describe its application in random network generation. In section 2.4, we introduce our new statistical model for network clustering.

2.2 Erdős-Rényi Random Graph Model

Due to the random nature of real world social networks, it is advantageous to use a statistical model to fit the observed adjacency matrix. A statistical model is a useful tool for better understanding internal structures of a social network, such as community or clustering structure. Here, we describe the first statistical model of network data: Erdős-Rényi random graph model. It was named after Paul Erdős and Alfred Rényi who first introduced this model at the year 1959. This is the simplest random graph model. Although it has no built-in clustering structures, it is still useful because it provides a good null model for network clustering algorithms we will describe later. There are two closely related variants of this model, the $G(n, M)$ model and the $G(n, p)$ model.

In the $G(n, M)$ model, a graph is randomly selected from all possible graphs of n nodes with M edges. In the $G(n, p)$ model, a graph is constructed by connecting

nodes randomly. Every pair of two nodes are connected by a fixed probability of p , and independent of the existence of an edge between any other pair of two nodes. The parameter p is called the link density. Mathematically, it can be expressed as follows:

$$P(A_{ij} = 1) = p, \quad (2.3)$$

where A_{ij} is the i th row and j th column of the adjacency matrix of the network, and $i \neq j$.

In this thesis, we will only consider the $G(n, p)$ model for its ease of analysis allowed by the independence of edges. If we observed a network of n nodes with M edges, the MLE estimator of p is :

$$\hat{p} = \frac{M}{n(n-1)}. \quad (2.4)$$

Definition 2.1. *The number of edges in the network with one end at v is defined as the degree of node v , and denoted as $\deg(v)$.*

The degree distribution of any particular node in a network is binomial. To be specific:

$$P(\deg(v) = k) = \binom{n-1}{k} p^k (1-p)^{n-k-1}. \quad (2.5)$$

If we let the network size n to be very large and keep the product np to be a constant, we have:

$$P(\deg(v) = k) \rightarrow \frac{(np)^k e^{-np}}{k!}. \quad (2.6)$$

Therefore, for a large network generated from $G(n, p)$ model with fixed np , its degree distribution of a particular node is approximately $\text{Poisson}(np)$. However, [Barabasi and Albert \(1999\)](#) claimed that many real world social network, including the world wide web have power-law degree distribution:

$$P(\deg(v) = k) \sim k^{-\gamma}, \quad (2.7)$$

where γ is a parameter whose value is typically between 2.1 and 4. The power-law distribution has heavier tail than the Poisson distribution. So we can see that real world social network is fundamentally different from the Erdős-Rényi network in the degree distribution.

It is also worth noting that the Erdős-Rényi random graph model can not incorporate any community structure, since there is equal probability of a link between any pair of two nodes. However, this simply null model is actually very important for network community detection. No matter what approach we use to detect community structure from a real world network, such as Facebook, we always need to define what constitute a community or cluster in a large network.

Definition 2.2. *A community in a network is defined as a subset of nodes which are closely connected with each other, but relatively loosely connected with the rest of the network.*

The keyword “relatively” is crucial. Intuitive, the probability of a link between every

pair of nodes in a community of a network should be roughly equal or at least similar. Otherwise, those nodes relatively more closely connected should be viewed as a small sub-communities inside this large community. This contradicts to the definition of a community in a network. In other words, a community should not contain deeper sub-community structure. Therefore, Erdős-Rényi random graph model can be used to fit a community or a cluster, the smallest building block of a much more complicated large network. This idea leads to a natural extension of Erdős-Rényi random graph model: the stochastic random block model, which was introduced in [Snijders and Nowicki \(1997\)](#). This will be the topic of the next section.

2.3 Stochastic Block Model

Real world social network such as the internet and Facebook usually contain community structure. As mentioned in the end of section 2.2, the link distribution among nodes within a particular community in a large network can be modeled by Erdős-Rényi random graph model. However, we also need to add additional structure and parameters to estimate the link distribution between nodes from different communities. This is the basic idea of Stochastic Block model which was introduced in [Snijders and Nowicki \(1997\)](#).

Definition 2.3. *Stochastic Block Model is a statistical model for a network with nodes $1, 2, \dots, n$ and communities $1, 2, \dots, h$ defined as follow:*

1. Every node v belongs to one of the h communities denoted by $c(v)$, where $c(v)$ has *i.i.d.* Multinomial distribution with $c(v) \sim \text{Multinomial}(\theta)$, $\theta = (\theta_1, \theta_2, \dots, \theta_h)$.

To be specific:

$$P(c(v) = k) = \theta_k, \quad (2.8)$$

where v is a node and k is a community, $1 \leq v \leq n$ and $1 \leq k \leq h$.

2. B is a $h \times h$ symmetric matrix of probabilities, such that conditional on the nodes community $c(1), c(2), \dots, c(n)$. The link distribution A_{ij} are independent with $A_{ij} \sim \text{Bernoulli}(B_{c(i), c(j)})$. To be specific:

$$P(A_{ij} = 1 | c(i), c(j)) = B_{ij}, \quad (2.9)$$

where A_{ij} is the link distribution between node i and j , $1 \leq i, j \leq n$.

From the definition above, we can see that stochastic blockmodel assume the network under study to be symmetric, that is, assuming the observed adjacency matrix A , to be symmetric: $A_{ij} = A_{ji}$, for $1 \leq i, j \leq n$. In fact, the triangle inequality of distance function require the distance matrix to be symmetric. The adjacency matrix and distance matrix are related in expression (2.2) for clustering purpose. Hence, it is reasonable to assume the adjacency matrix to be symmetric. In real world social network, there exists situations when the adjacency matrix tends to be asymmetric. In that case, we can

symmetrize the adjacency matrix by making more assumptions such as:

$$A_{ij}^{(new)} = \max(A_{ij}, A_{ji}), \forall 1 \leq i, j \leq n.$$

Further more, if two nodes i and j belong to the same community k , that is $c(i) = c(j) = k$, the probability of a link between them is B_{kk} , the k th diagonal element of the parameter matrix B . Therefore, the link distribution among nodes in k th community is Bernoulli(B_{kk}), just like the Erdős-Rényi Random Graph model described. If two nodes i and j belong to different communities, that is $c(i) \neq c(j)$, the probability of a link between them is $B_{c(i),c(j)}$, the $c(i)$ th row, $c(j)$ th column element in parameter matrix B . This is the additional modeling structure we needed to model real world social network with community structures.

Given the parameter (θ, B) , the complete likelihood function of a network with adjacency matrix A and community membership c can be written as

$$P(A, c; \theta, B) = \theta_1^{n_1} \dots \theta_h^{n_h} \prod_{1 \leq k \leq l \leq h} B_{kl}^{e_{kl}} (1 - B_{kl})^{n_{kl} - e_{kl}}, \quad (2.10)$$

where

$$e_{kl} = \frac{1}{1 + \delta_{kl}} \sum_{1 \leq i \neq j \leq n} a_{ij} I(c(i) = k) I(c(j) = l) \quad (2.11)$$

denotes the number of links with one node in community k and another node in community l , and

$$n_{kl} = \begin{cases} n_k n_l & , \text{ if } k \neq l \\ \binom{n_k}{2} & , \text{ if } k = l \end{cases} \quad (2.12)$$

denotes the maximal number of links between community k and community l , and

$$n_k = \sum_{i=1}^n I(c(i) = k) \quad (2.13)$$

denotes the number of nodes in community k , and

$$\delta_{kl} = \begin{cases} 1 & , \text{ if } k = l \\ 0 & , \text{ if } k \neq l \end{cases} \quad (2.14)$$

denotes an indicator of two equivalent communities.

2.3.1 Parameter Estimation

Here our objective is to estimate parameter θ and B . The adjacency matrix A is observed, but the community structure c is unknown. The likelihood function of the observed adjacency matrix A given the parameters θ and B is:

$$P(A; \theta, B) = \sum_c P(A, c; \theta, B). \quad (2.15)$$

Snijders and Nowicki (1997) have tried both direct maximizing the likelihood function of (2.15) and the EM algorithm described in Dempster et al. (1977) to get the parameter estimate for θ and B . Unfortunately, the authors have found out that these maximization approaches are only computationally feasible for small network of size up to 20.

An alternative approach to get parameter estimation is the Bayesian method. We treat the parameter (θ, B) as a random vector with some prior distribution $f(\theta, B)$. Now we need to simulate the posterior distribution: $(\theta, B, c(1), \dots, c(n))$ given the observed adjacency matrix A . This can be done by the Gibbs sampling method in Geman and Geman (1983) and further explained in Gelfand and Smith (1990) and Casella and George (1992). Gibbs sampling is an iterative sampling scheme to simulate complicated multivariate distributions. Given the current values $\theta^{(k)}$, $B^{(k)}$ and $c^{(k)}$, the next round of $\theta^{(k+1)}$, $B^{(k+1)}$, and $c^{(k+1)}$ can be determined by following procedure:

1. $\theta^{(k+1)}$ and $B^{(k+1)}$ can be simulated from the posterior joint distribution of (θ, B) , given complete data $(c^{(k)}, A)$.
2. $c^{(k+1)}(1)$ can be simulated from the posterior distribution of $c(1)$, given $\theta^{(k+1)}$, $B^{(k+1)}$, A , and $c^{(k)}(2), \dots, c^{(k)}(n)$.
3. For $i = 2, \dots, (n - 1)$ in turn, $c^{(k+1)}(i)$ can be simulated from the posterior distribution of $c(i)$, given $\theta^{(k+1)}$, $B^{(k+1)}$, A , $c^{(k+1)}(1), \dots, c^{(k+1)}(i - 1)$, and $c^{(k)}(i + 1), \dots, c^{(k)}(n)$.
4. $c^{(k+1)}(n)$ can be simulated from the posterior distribution of $c(n)$, given $\theta^{(k+1)}$,

$B^{(k+1)}$, A , and $c^{(k+1)}(1), \dots, c^{(k+1)}(n-1)$.

Please refer to the original paper of [Snijders and Nowicki \(1997\)](#) for more details on the calculations of all above conditional distribution. The convergence theorem of [Geman and Geman \(1983\)](#) claimed that, regardless of the starting values, the distribution of $(\theta^{(k)}, B^{(k)})$ will converge to the posterior distribution of (θ, B) given observed adjacency matrix A . This simulated sample can provide any type of inference for parameter $(\theta^{(k)}, B^{(k)})$. The distribution of $c^{(k)} = (c_1^{(k)}, \dots, c_n^{(k)})$ will converge to the posterior distribution of community structure c of the network, given observed adjacency matrix A . This simulated sample can provide inference for the community structure of the social network, which is the topic of the next section.

2.3.2 Community Prediction

In this section, our objective is to predict the community structure of a social network. Suppose that by using the Gibbs sampling technique, we have collected a simulated sample of the posterior distribution of the community parameter c :

$$P(c|A) \propto \int P(A, c|\theta, B) f(\theta, B) d\theta dB. \quad (2.16)$$

Above expression is the conditional distribution of community parameter c , given the observed adjacency matrix A , when the parameter (θ, B) has prior distribution $f(\theta, B)$. Therefore, the relative frequency of $\{r; c(r)(i) = k, r > \text{burn in iteration}, i = 1, \dots, n\}$ in

the simulated sample is the Bayes estimator of $P(c(i) = k)$, the posterior probability of node i belong to the community k . In practice, the community prediction of a particular node i will be the community with largest posterior probability:

$$\operatorname{argmax}_{1 \leq k \leq h} P(c(i) = k), \quad (2.17)$$

where $i = 1, \dots, n$. It has been showed in [Snijders and Nowicki \(1997\)](#) that if the assumption contained in Definition 2.3 is valid, and one uses an appropriate statistical procedure to predict the community of every nodes in the network, the prediction will be correct with probability approaching to 1 as $n \rightarrow \infty$. Empirical results show that if $n > 30$, the prediction of community is quite good. So the approach of using stochastic block model for network community detection or clustering is reliable.

2.3.3 Additional Comments

So far, we have illustrated the basic idea of using stochastic block model to discover community structure from a social network. This is a starting point of more advanced network clustering approach. We will summarize both the advantage and disadvantage of using this method.

For the advantage:

1. It provides not only parameter estimates, but also the statistical inference on those estimates. We how confident we are on the communities we found from the

network.

2. The Gibbs sampling procedure is relatively easy to implement, and it works for large data sets.
3. We can fix a value of the parameter (θ, B) , or just fix community parameter c , and use them to simulate random network. The randomly generated network with built in clusters is convenient for evaluating the performance of any other network clustering algorithms.

For the disadvantage:

1. In real world social network, we may have additional information for each node, in addition to the adjacency matrix. Stochastic block model can not incorporate this information directly. [Handcock et al. \(2007\)](#) has proposed a latent position cluster model, which is capable of handling auxiliary variables. A potential problem for this approach is that the effects of auxiliary variables and clustering may compete with each other, which may jeopardize the clustering results.
2. In real world social network, a node may belong to multiple communities instead of a single community. [Airoldi et al. \(2008\)](#) introduced the Mixed membership block model, that naturally extends stochastic block model to allow a mixture of community membership for any particular node in the social network. [Airoldi et al. \(2008\)](#) also adopted Bayesian approach, but used Variational Bayes method as opposed to Gibbs sampling to make the computation more efficient.

3. Any clustering is based on a particular type of distance function explicitly or implicitly. The stochastic block model does not provide any insight about the distance function behind the clustering results. In the next section, we will describe a novel statistical model of network data, which illustrates the relationship between the observed adjacency matrix and cosine similarity matrix of the mixed community membership of nodes.
4. Model-based clustering approach is usually much less efficient than an optimization based clustering approach due to the MCMC sampling procedure. Possible remedies including using the Variational Bayes approach to transform it into an optimization problem.

2.4 Distance-based Mixed Membership Model

As mentioned in section 2.3.3, for every clustering algorithm, some distance function is assumed, either explicitly or implicitly. In fact, a distance function is a measure that quantifies how far away two points lie in the social network space. It is a necessary part for any clustering and community detection task. To build a statistical model with major purpose of network clustering, it is beneficial to include the distance function in the model explicitly. However, the stochastic block model does not provide any insight of the distance function it implies underlying the clustering results. In this section, our purpose is to develop a novel statistical model for network data, that explores the relationship

between the observed adjacency matrix and the underlying distance function we use for clustering.

In essence, the probability of a link between two nodes in the social network is proportional to their “similarity”, and inverse proportional to their “distance”. To build a novel statistical model for network clustering purpose, there are following couple of things to consider:

1. How to define a space that every node in a social network has a location represented by some coordinate system?
2. What distance function to use to quantify the distance or similarity of two nodes in our defined social network space?
3. How to use the distance or similarity between two nodes in a social network to predict the probability of a link between them?

For question 1, since our purpose is network clustering or community detection, we will use the mixed community membership vector to represent the location of a node in social network. For example, a social network has 3 communities $\{1, 2, 3\}$. A node v has a community membership vector $(0.2, 0.3, 0.5)^T$ means that v has 20% of membership at community 1, 30% of membership at community 2, and 50% of membership at community 3. This approach is advantageous to the stochastic block model, because it is capable of handling the situations when a node belong to more than one communities, which is very common in real world social network.

For question 2, we decide to use the cosine similarity:

$$\begin{aligned} \text{similarity} &= \cos(x, y) = \frac{x^T y}{\|x\|_2 \|y\|_2} \\ &= \frac{\sum_{i=1}^r x_i y_i}{\sqrt{\sum_{i=1}^r x_i^2} \sqrt{\sum_{i=1}^r y_i^2}} \end{aligned}$$

where x and y are r -dimensional vectors, $\|\cdot\|_2$ stands for ℓ^2 norm. The cosine distance is defined as $1 - \text{cosine similarity}$. We choose the cosine similarity mainly because of its successful applications of high-dimensional data such as information retrieval and text mining. We believe that social network data is high-dimensional by its nature. Another advantage of using cosine similarity is its relation to Pearson Correlation Coefficient. When data are centered by their mean, the cosine similarity and Pearson correlation coefficient are equivalent.

For question 3, since cosine similarity ranges from 0 to 1 which is identical to the range of probability, we assume the probability of a link between two nodes in a social network equals their cosine similarity.

In the next section, we will introduce our distance-based mixed membership model formally.

2.4.1 Model Description

In this section, we formally introduce the distance-based mixed membership model. Assume we have a social network of n nodes, and we want to detect h communities. For

each node i , we assume it has an unobserved location:

$$Z_i = (Z_{i1}, Z_{i2}, \dots, Z_{ih})^T, \quad (2.18)$$

with restriction $\sum_{k=1}^h Z_{ik} = 1$, where $i = 1, 2, \dots, n$, and $0 \leq Z_{ik} \leq 1$ for $1 \leq k \leq h$.

Here the unobserved location Z_i represent the mixed membership of node i among all h candidate communities. To be specific, Z_{ik} is a percentage of membership of node i in community k . For example, if we have $Z_{ik} = 0.7$, and $Z_{il} = 0.3$, then node i has 70% of membership in community k and 30% of membership in community l .

The link distribution A_{ij} between node i and j given their community membership vector Z_i, Z_j is independent of any other link distribution, and can be expressed as:

$$P(A_{ij} = a_{ij} | Z_i, Z_j) = \left(\frac{Z_i^T Z_j}{\|Z_i\|_2 \|Z_j\|_2} \right)^{a_{ij}} \left(1 - \frac{Z_i^T Z_j}{\|Z_i\|_2 \|Z_j\|_2} \right)^{1-a_{ij}} \quad (2.19)$$

or

$$A_{ij} \sim \text{Bernoulli}\left(\frac{Z_i^T Z_j}{\|Z_i\|_2 \|Z_j\|_2}\right), \text{ given } Z_i, Z_j, \quad (2.20)$$

where

$$\|Z_i\|_2 = \sqrt{\sum_{k=1}^h Z_{ik}^2}, \text{ for } 1 \leq i \leq n$$

denotes the ℓ^2 norm of mixed community membership vector Z_i , and

$$a_{ij} = \begin{cases} 1, & \text{if node } i \text{ and } j \text{ are connected} \\ 0, & \text{if node } i \text{ and } j \text{ are not connected} \end{cases}.$$

The complete likelihood function of the observed adjacency matrix $A = ((a_{ij}))$ is

$$P(A = ((a_{ij}))|Z) = \prod_{1 \leq i < j \leq n} \left(\frac{Z_i^T Z_j}{\|Z_i\|_2 \|Z_j\|_2} \right)^{a_{ij}} \left(1 - \frac{Z_i^T Z_j}{\|Z_i\|_2 \|Z_j\|_2} \right)^{1-a_{ij}}, \quad (2.21)$$

where $Z = (Z_1, Z_2, \dots, Z_n)$ is an $h \times n$ matrix, that represents the community membership of all nodes in the social network.

Please note that once we have an estimate of the community membership matrix parameter Z , the probability of a link between any pair of two nodes i and j are just the cosine similarity of their community membership vector Z_i and Z_j . The cosine similarity is very similar to the Pearson Correlation Coefficient. Our distance-based mixed membership model illustrate the relationship between the observed adjacency matrix and the underlying cosine distance very intuitively.

2.4.2 Parameter Estimation

There is no simple way to get the Maximum Likelihood Estimator of community membership parameter Z . So we will use the Bayesian approach just like section 2.3.2. We assume the prior distribution of the mixed membership parameter Z_i of each node i to

be independent and identically distributed as:

$$Z_i \stackrel{i.i.d.}{\sim} \text{Dirichlet}(\alpha), \text{ for } 1 \leq i \leq n, \quad (2.22)$$

where α is an h dimensional vector of the hyper-parameter of prior Dirichlet distribution.

Our objective is to obtain the posterior distribution of each Z_i given the observed adjacency matrix $A = ((a_{ij}))$. We will use the Gibbs sampling technique introduced by [Geman and Geman \(1983\)](#) and further explained by [Gelfand and Smith \(1990\)](#) and [Casella and George \(1992\)](#). Gibbs sampling is especially suitable for the simulation of high-dimensional posterior distribution of the parameter vector we want to study.

Gibbs sampling is an iterative sampling scheme to simulate the target posterior distribution of Z , given adjacency matrix A . Given current iteration of values $Z^{(k)} = (Z_1^{(k)}, \dots, Z_n^{(k)})$, the next iteration of values $Z^{(k+1)} = (Z_1^{(k+1)}, \dots, Z_n^{(k+1)})$ can be simulated in following way:

1. $Z_1^{(k+1)}$ is simulated from the posterior distribution of Z_1 , given $Z_2^{(k)}, \dots, Z_n^{(k)}, \alpha, A$.
2. For $i = 2, 3, \dots, n-1$, $Z_i^{(k+1)}$ is simulated from the posterior distribution of Z_i , given $Z_1^{(k+1)}, \dots, Z_{i-1}^{(k+1)}, Z_{i+1}^{(k)}, \dots, Z_n^{(k)}, \alpha, A$.
3. $Z_n^{(k+1)}$ is simulated from the posterior distribution of Z_n , given $Z_1^{(k+1)}, \dots, Z_{n-1}^{(k+1)}, \alpha, A$.

The convergence theorem of [Geman and Geman \(1983\)](#) claimed that, regardless of the starting values, the distribution of $Z^{(k)}$ will converges to the posterior distribution of community membership parameter Z , given the observed adjacency matrix A . In practical use of Gibbs sampling, we will only keep the simulated values of $Z^{(k)}$ after the simulated distribution has converged. The threshold of iteration, after which we will start keep the simulated values is called the burn in number. For the convenience of notation, we will denote

$$Z_{-i} = (Z_1, \dots, Z_{i-1}, Z_{i+1}, \dots, Z_n). \quad (2.23)$$

In order to complete the Gibbs sampling procedure, we need to derive the expression of the posterior distribution of Z_i , given Z_{-i} , A , and α . The results are as follow:

$$\begin{aligned} P(Z_i|A, Z_{-i}, \alpha) &= \frac{P(Z, A|\alpha)}{P(Z_{-i}, A|\alpha)} \\ &\propto P(A|Z, \alpha)P(Z|\alpha) \\ &\propto \prod_{j=1, j \neq i}^n \left(\left(\frac{Z_i^T Z_j}{\|Z_i\|_2 \|Z_j\|_2} \right)^{a_{ij}} \left(1 - \frac{Z_i^T Z_j}{\|Z_i\|_2 \|Z_j\|_2} \right)^{1-a_{ij}} \right) \prod_{k=1}^h Z_{ik}^{\alpha_k - 1}, \end{aligned} \quad (2.24)$$

where the last step of expression (2.24) can be derived from expression (2.21), and the density function of Dirichlet distribution with hyper-parameter α .

Since there is no well-known distribution proportional to the posterior density function in expression (2.24), we will use Metropolis – Hastings sampling to simulate it at

each step of Gibbs sampling. The complete sampling procedure can be organized in Algorithm 2.1, in which *burninNum* is the number of burn in MCMC sampling iterations

Algorithm 2.1 Gibbs Sampling Algorithm for Distance-based Mixed Membership Model

Inputs: *burninNum* = 5000, *size* = 10000, empty set *posteriorSample*

- 1: Initialize $Z_{ik} \leftarrow \frac{1}{h}$, for all $i = 1, \dots, n$ and $k = 1, \dots, h$
- 2: Initialize iteration number *iterNum* $\leftarrow 1$
- 3: **repeat**
- 4: **for** i from 1 to n **do**
- 5: Simulate $T_i \sim \text{Dirichlet}(\alpha)$
- 6: **if** $\frac{(\prod_{j=1, j \neq i}^n (\frac{T_i^T Z_j}{\|T_i\|_2 \|Z_j\|_2})^{a_{ij}} (1 - \frac{T_i^T Z_j}{\|T_i\|_2 \|Z_j\|_2})^{1-a_{ij}} \prod_{k=1}^h T_{ik}^{\alpha_k-1}}{(\prod_{j=1, j \neq i}^n (\frac{Z_i^T Z_j}{\|Z_i\|_2 \|Z_j\|_2})^{a_{ij}} (1 - \frac{Z_i^T Z_j}{\|Z_i\|_2 \|Z_j\|_2})^{1-a_{ij}} \prod_{k=1}^h Z_{ik}^{\alpha_k-1}} = r \geq 1$ **then**
- 7: Set $Z_i \leftarrow T_i$
- 8: **else**
- 9: Set $Z_i \leftarrow T_i$ with probability r
- 10: **end if**
- 11: **end for**
- 12: **if** *iterNum* > *burninNum* **then**
- 13: Add $Z = (Z_1, \dots, Z_n)$ to *posteriorSample*
- 14: **end if**
- 15: Set *iterNum* \leftarrow *iterNum* + 1
- 16: **until** *iterNum* > *size* + *burninNum*

Output: *posteriorSample*

we will skip, before we start to collect the simulated values, and *size* is sample size we need for *posteriorSample*. The burn in procedure is crucial because it helps make sure the simulated distribution of Z has converged.

After we obtained the *posteriorSample* of the mixed community membership parameter Z , we choose the sample mean \bar{Z}_i as the Bayes estimator of Z_i , for $i = 1, 2, \dots, n$. In case we need to do hard clustering on a graph, we can simply classify node i into community $\underset{k}{\operatorname{argmax}}(\bar{Z}_{ik})$, which is the dominate membership.

2.4.3 Example: Zachary Karate Club Data

The best way to understand a statistical model for social network is to apply it to some real world social networks, and see how it works. In this section, we will study the well-known Zachary karate club data prepared by Wayne Zachary in 1977. Each node represents a member of the club, and each edge represents a tie between two members of the club. This is a classical social network dataset from the literature. The network is very small: it has 34 vertices and 78 indirect edges. [Zachary \(1977\)](#) used these data and an information flow model of network conflict resolution to explain the split-up of this group following disputes among the members.

This data set is a good example for testing network clustering algorithm because we know the ground truth of the split in real world after the network conflict. The [Figure 2.1](#) illustrates this split.

Here we apply our distance-based mixed membership model to above data set using the Gibbs sampling method described in section 2.4.2. Since there are two obvious clusters in this network, it is a good idea to let the number of communities parameter $h = 2$. The results are summarized in [Table A.1](#) of the Appendix, in which, the mean, median, 2.5th percentile, and 97.5th percentile of posterior distribution of Z_{ik} are provided, for $i = 1, \dots, n$, and $k = 1, 2$. We will use the posterior mean \bar{Z}_{ik} as the estimator of parameter Z_{ik} . For hard clustering, a particular node i will be assigned to community

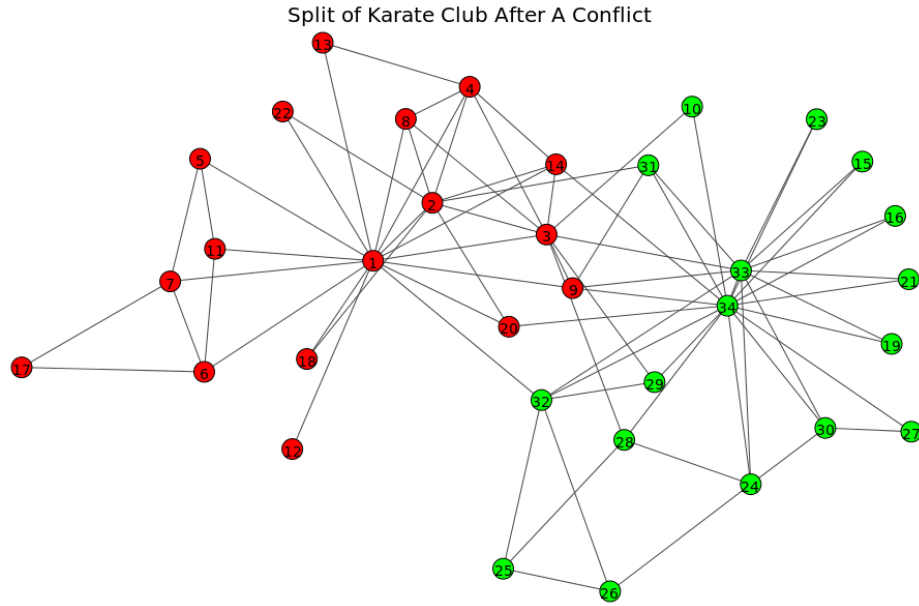


Figure 2.1: Plot of the split of Karate Club Network of [Zachary \(1977\)](#) after a conflict between the two instructors. The members tend to split into two groups following these two instructors.

of its leading membership:

$$c(i) = \max_{k=1,2} Z_{ik}, \quad (2.25)$$

where $i = 1, \dots, n$. The hard clustering results are summarized in [Figure 2.2](#).

Both [Figure 2.1](#) and [2.2](#) split the Karate Club Network into two communities colored in red and green. By comparison, we can see that node 9 is the only node that is misclassified from the ground truth in [Figure 2.1](#). In fact, node 9 is connected with three nodes from community 1 (white node 31, 33, and 34, in [Figure 2.1](#)), and with only two nodes from community 2 (red node 1 and 3 in [Figure 2.1](#)). Therefore, it is still “reasonable” to classify node 9 to community 1 in [Figure 2.2](#), although in reality, node 9 joined community 2 after the split of the karate club.

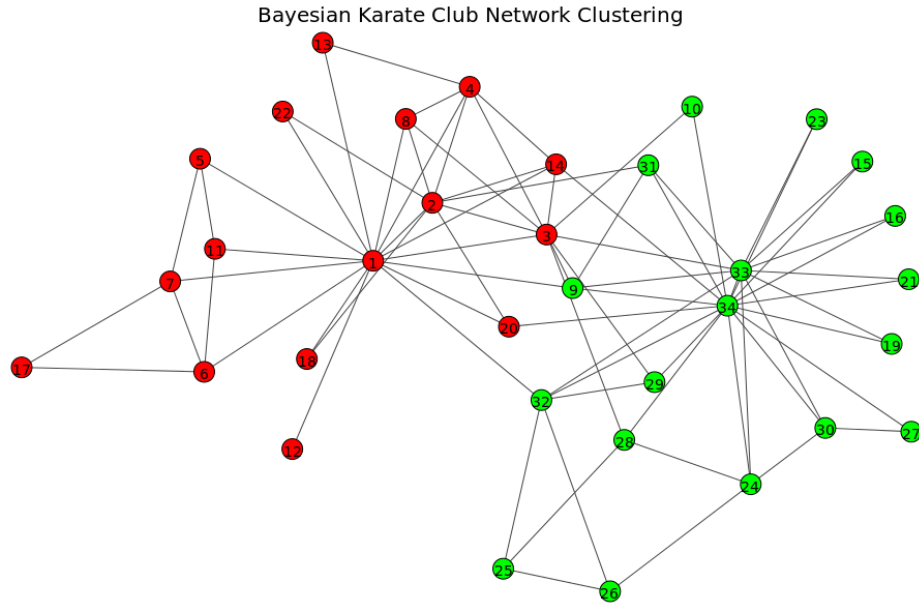


Figure 2.2: Plot of the split of the Karate Club Network of [Zachary \(1977\)](#) by Distance-base Mixed Membership Model.

2.4.4 Examples: Dolphin Network Data

The dolphin social network data was constructed from observations of a group of 62 bottlenose dolphins living in Doubtful Sound, New Zealand, over a period of seven years from 1994 to 2001 by [Lusseau et al. \(2003\)](#). This social network consists of 62 nodes of dolphins with 318 edges representing dolphin connections.

The purpose of [Lusseau et al. \(2003\)](#) paper was to find communities structure of dolphin social network, and study how these divisions arise. Lusseau adopted the betweenness-based network clustering algorithm by [Girvan and Newman \(2002\)](#). This method finds natural division of network into tightly connected groups by looking for edges between groups. These edges are identified by a “betweenness” measure, which is

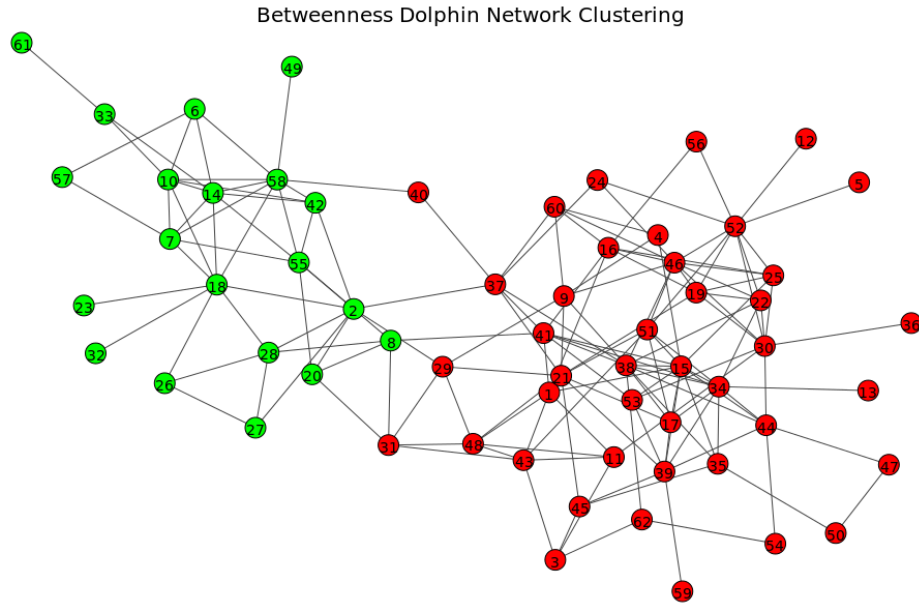


Figure 2.3: Split of the Dolphin Network of [Lusseau et al. \(2003\)](#) by betweenness-based network clustering algorithm.

a generalization of the edges of node betweenness measure by [Freeman \(1977\)](#). Edges with highest score are removed from the network, leaving behind the groups themselves.

Despite of the inefficiency of the betweenness based network clustering algorithm, it has been proved to be accurate and sensitive in real world application. Here in this section, we will use the clustering results by betweenness measure as a reference, and compare it with our model-based network clustering algorithm described in section 2.3.1. through 2.3.3. The clustering result of the betweenness-based algorithm on the Dolphin Network is shown in Figure 2.3.

In this application, we choose the number of communities in the network to be $h = 2$. The MCMC posterior sample summary of the 62 mixed membership parameter Z_i is organized in Table A.2 of the Appendix, where $i = 1, \dots, 62$. In order to compare

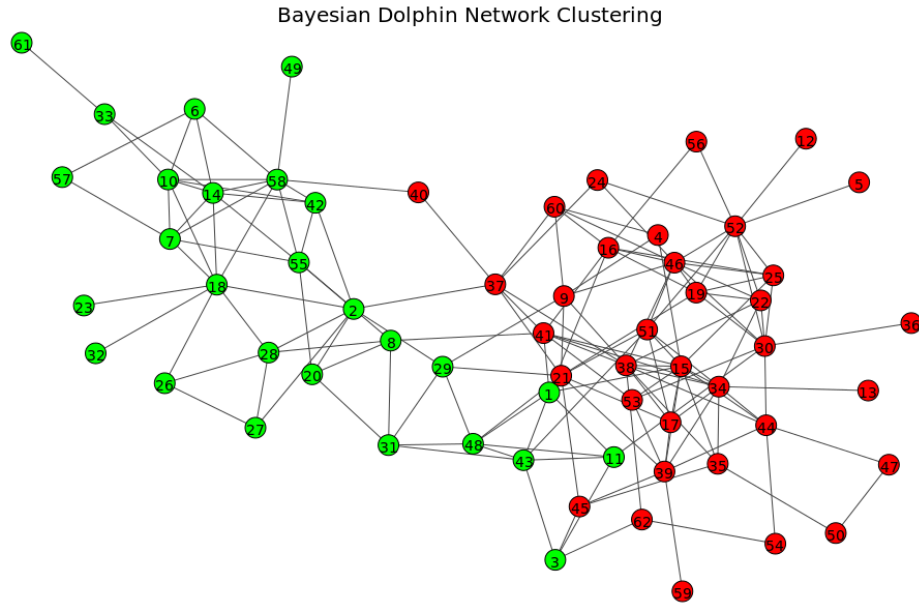


Figure 2.4: Split of the Dolphin Network of [Lusseau et al. \(2003\)](#) by distance-based mixed membership model.

with the hard clustering of betweenness-based algorithm, we also use expression 2.25 to find the leading community membership for every of these 62 dolphins. The clustering results from our distance-based mixed membership model is shown in the Figure 2.4 for comparison with Figure 2.3.

Both plots split the Dolphin Network into two communities colored in red and green. we can see that the division of the dolphin network from these two algorithms are very similar. They match on all but 7 nodes with id: 1, 3, 11, 29, 31, 43, 48. These 7 nodes lie on the boundary between the two communities. In the paper of [Lusseau et al. \(2003\)](#), these 7 nodes form a sub-community inside the bigger red community, so we can see our model-based clustering is also providing reasonable results.

2.4.5 Final Comments about Mixed Membership Model

In this section, we will summarize some advantages and disadvantages of our novel distance-based mixed membership model.

For advantages:

1. The cosine similarity function is used directly to predict the link probability between any pair of two nodes. This provides great insights into the relationship between the clustering and the underlying distance function.
2. The Bayesian MCMC method provides a posterior sample of the community membership parameter. So we know the distribution of the parameter, and can perform any inference.
3. A model-based clustering algorithm provide not only the community structure of a network, but also the insight about how the observed network is formed.

For disadvantages:

1. The major disadvantages of our mixed membership model is that it is much slower than some optimization-based algorithm that we will introduce in Chapter 3. In fact, this is the major reason that the most commonly used network clustering algorithms are usually optimization-based algorithm.
2. It can not take advantages of additional information we have for every node in the social network to improve clustering results.

The ultimate goal of social network clustering is that it can be as efficient as a typical optimization-based algorithm, but could also perform some statistical inference as a model-based algorithm.

Chapter 3

Clique-based Community Detection

3.1 Introduction

In Chapter 2, we have explored the basic concepts about network clustering, reviewed the Erdős-Rényi Random Graph model, Stochastic Block model, and developed a novel mixed membership model for social network community detection. The major advantages of this category of model is to perform statistical inference in addition to the clustering results. The major disadvantage is that the number of parameters is proportional to the size of the network. If the network is large, it can be computationally expensive to simulate the posterior distribution of the large parameter space. In practice, model-based network clustering are usually much less efficient than their counterpart optimization-based network clustering algorithms described in this chapter. Since social network community detection is a big data problem by its nature, the scalability of a

clustering algorithm is crucial for its popularity in real world application.

The major topic of this chapter is optimization-based network community detection algorithm. This category of algorithm usually start with an objective function that quantifies the quality of a division of a network, and then strategically optimize that objective function. This objective functions usually closely related to the observed adjacency matrix of the network. Commonly used objective function include the normalized minimum cut by [Shi and Malik \(2000\)](#) and [Ng et al. \(2001\)](#), and the Modularity Score by [Newman \(2006\)](#). We will create a new objective criterion called “Clique Score”, and develop a clustering algorithm based on it. An advantage of our approach is that it provides user control of what kinds of community will be detected.

The rest of this Chapter is organized as follows. In section 3.2, we review the spectral clustering algorithm that minimizes the normalized minimum cut. In section 3.3, we review the Modularity Maximization algorithm, and discuss the resolution limit of Modularity-based clustering algorithm. In section 3.4, we discuss intuitions for developing Clique-based algorithm. In section 3.5, we formally describe the clique-based algorithm. In section 3.6, we apply our Clique-based clustering algorithm to random network simulated from stochastic block model.

3.2 Spectral Clustering for Network Data

This section will briefly describe the spectral clustering algorithm for network data, introduced by [Chung \(1997\)](#), and further explained by [Ng et al. \(2001\)](#). We follow the notations developed in chapter 1 and 2. Define $A = ((a_{ij}))$ to be the adjacency matrix of a social network G of n nodes $\{1, 2, \dots, n\}$. This adjacency matrix A is symmetric, and all diagonal elements $a_{ii} = 0$, for $i = 1, 2, \dots, n$. According to Definition 2.1, $deg(i)$ is defined as the degree of node i . Also, the degree matrix R of this network is defined as

$$R = \begin{pmatrix} deg(1) & 0 & \cdots & 0 \\ 0 & deg(2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & deg(n) \end{pmatrix}. \quad (3.1)$$

The volume of this network is defined as

$$Vol(G) = \sum_{i=1}^n deg(i). \quad (3.2)$$

The number of edges between two sub-network G_1 and G_2 is defined as

$$Cut(G_1, G_2) = \frac{1}{2} \sum_{i \in G_1, j \in G_2} a_{ij}. \quad (3.3)$$

The unnormalized Laplacian matrix L is defined as

$$L = R - A. \quad (3.4)$$

The normalized Laplacian matrix L_{sym} is defined as

$$L_{sym} := R^{-1/2} L R^{-1/2} = I - R^{-1/2} A R^{-1/2}. \quad (3.5)$$

The following proposition develops the desired objective function.

Proposition 3.1. *The Laplacian matrix L satisfy following properties:*

1. *For every vector $c \in \mathbb{R}^n$ we have*

$$c^T L_{sym} c = \frac{1}{2} \sum_{i,j=1}^n a_{ij} \left(\frac{c_i}{\sqrt{\deg(i)}} - \frac{c_j}{\sqrt{\deg(j)}} \right)^2. \quad (3.6)$$

2. *0 is an eigenvalue of L_{sym} with eigenvector $R^{1/2} \mathbb{1}_n$, where $\mathbb{1}_n$ denotes the n -dimensional column vector with all entries equal to 1.*

3. *L_{sym} is semi-positive definite and have n non-negative real eigenvalues*

$$0 = \lambda_1, \dots, \lambda_n.$$

The expression (3.6) is the objective function we would like to minimize, with restriction $c^T c = 1$ and $c^T \mathbb{1}_n = 0$. Intuitively, in order to minimize expression (3.6), we

need all connected pairs of nodes i and j , in which $a_{ij} = 1$, to have similar values, such that

$$\frac{c_i}{\sqrt{\deg(i)}} \approx \frac{c_j}{\sqrt{\deg(j)}}.$$

Therefore, we can treat vector c as a community indicator for the division of the network. For example, If we hope to split the network into two parts, then nodes with positive c_i tend to form one community, and nodes with negative c_j tend to form another community. We also see that the restriction $c^T c = 1$ and $c^T \mathbb{1}_n = 0$ is important, otherwise, the minimum value of expression (3.6) will be achieved when $c = \frac{\mathbb{1}_n}{\sqrt{n}}$. In this case, all nodes are in the same group, and there is actually no division.

In the general case, when we would like to split the network into h communities: $\{G_1, \dots, G_h\}$. Intuitively, we want to minimize the number of edges across different communities, adjusted by the number of edges in each community. We call this objective function the Normalized Cut, defined as follows:

$$\begin{aligned} Ncut(G_1, \dots, G_h) &= \sum_{k=1}^h \frac{Cut(G_k, \overline{G_k})}{Vol(G_k)} \\ &= \text{Tr}(P^T L P), \end{aligned} \tag{3.7}$$

where $\overline{G_k}$ is the complement of G_k , and P is an $n \times h$ matrix of community indicator

for all n nodes, such that

$$P_{ij} = \begin{cases} 1/\sqrt{\text{Vol}(G_j)} & , \text{ if } i \in G_j \\ 0 & , \text{ if } i \notin G_j \end{cases} . \quad (3.8)$$

It can be easily verified that $P^T R P = I_h$, where R is defined in expression (3.1). If we allow P to be any real value $n \times h$ matrix satisfying $P^T R P = I_h$, then minimizing objective function (3.7) can be done approximately by

$$\begin{aligned} & \min_{G_1, \dots, G_h} \text{Tr}(P^T L P) , \text{ under (3.8)} \\ & \approx \min_{P \in \mathbb{R}^{n \times h}} \text{Tr}(P^T L P) , \text{ under } P^T R P = I_h \\ & = \min_{C \in \mathbb{R}^{n \times h}} \text{Tr}(C^T L_{sym} C) , \text{ under } C^T C = I_h, \end{aligned} \quad (3.9)$$

where $C = R^{1/2} P$, R , L , and L_{sym} are defined in (3.1), (3.4), and (3.5) respectively. Expression (3.9) is a standard form of trace minimization problem. A version of Rayleigh-Ritz Theorem in section 8.2 of [Hogben \(2013\)](#) claims that the solution is choosing C to be the first h orthogonal eigenvectors of L_{sym} as columns. In theory, every row of matrix C represents the feature vector of a node in the network, so we can perform the k-mean algorithm on these n rows of feature vector, and get the clustering results.

In practice, however, we need to first normalize every row of matrix C to be norm 1 before we apply the k-mean algorithm. In fact, if there are k disconnected part of

a network, the first k orthogonal eigenvectors of L_{sym} will be indicator matrix $C(k) = [R^{1/2}\mathbb{1}_{G_1}, \dots, R^{1/2}\mathbb{1}_{G_k}]$. So, in this ideal case, every row i of $C(k)$ is consist of only one nonzero element: the indicator of the community membership of node i . The effectiveness of this spectral clustering algorithm relies on that all these nonzero indicator entries in the eigenvectors should be far away from 0, especially in non-ideal case when the network does not have k disconnected parts. Otherwise, these indicator vector won't be significant enough for k-mean algorithm to discover the community structure in the network.

In summary, the spectral clustering algorithm for network data minimize the Normalized Cut on edges across different communities. We want to emphasize the importance of “Normalized” here. If we simply count the number of cuts on edges across different communities without adjusting to the community size as we do in expression (3.7), we may end up with every biased split of a network. For example, one community may contains only one node, and another community may contains all the rest. This kind of division is obviously not useful for our study to the structure of a network. Therefore, the concept of minimizing the “Normalized Cut” is advantageous than a trivial objective function that simply minimizes the total number of cuts. This spectral clustering algorithm has been proven to be useful in many real world applications.

However, this algorithm also have some disadvantages.

1. Just as most model-based network clustering algorithm, we need to specify the number of clusters, which turn out to be not an easy task.

2. It usually divides a network into communities of similar sizes. However, large scale real world social network may include communities of very different sizes.
3. No statistical inference is included in every community we find.

In the rest of this chapter and Chapter 4, we will explore solutions to overcome these disadvantages. The idea of using adjacency matrix to construct an objective function will be the main approach of all network clustering algorithms described in this chapter.

3.3 Modularity Maximization

Network clustering algorithms via modularity maximization is one of the most commonly used approaches nowadays. It was first introduced by [Newman \(2006\)](#), and it overcomes the first two disadvantages of spectral clustering. The concept of modularity comes from this idea: in a reasonably good division of a network into communities, the number of edges between communities is significantly less than we expect by chance, and equivalently the number within communities is significantly more. We need to quantify this idea by a measure of how good a division is. This measure is known as "Modularity".

3.3.1 Modularity Score

Given the degree, $\deg(i)$ of each node i in a random network G of n nodes, and performing uniformly random pairing of these total of $\text{Vol}(G)$ edges. It can be shown that the expected number of edges between nodes i and j is $\frac{\deg(i)\deg(j)}{\text{Vol}(G)}$.

Definition 3.1. *The modularity of a network clustering is defined as*

$$Q = \frac{1}{Vol(G)} \sum_{1 \leq i, j \leq n} (a_{ij} - \frac{deg(i)deg(j)}{Vol(G)}) \delta(c_i, c_j), \quad (3.10)$$

where c_i denotes the community membership of node i of the clustering, and

$$\delta(c_i, c_j) = \begin{cases} 1 & , \text{ if } c_i = c_j \\ 0 & , \text{ if } c_i \neq c_j \end{cases}. \quad (3.11)$$

Intuitively, we can see that modularity of a network clustering is a measure of the difference on the link density among nodes of same community membership between the observed network and a random network with same degree for each nodes. Modularity is a quantity between 0 and 1. A higher modularity score, indicates a better network clustering. Therefore, our objective is to maximize the modularity score over all possible clustering of a network.

3.3.2 Split the Network Into Two Communities

In this section, we consider the case to split a network into two communities $\{1, 2\}$. For an arbitrary node i in a network G of n nodes, we denote

$$s_i = \begin{cases} 1 & , \text{ if } i \in \text{Community 1} \\ -1 & , \text{ if } i \in \text{Community 2} \end{cases}. \quad (3.12)$$

From expression (3.12), observing that

$$\frac{s_i s_j + 1}{2} = \begin{cases} 1 & , \text{ if } i, j \text{ are in the same community} \\ 0 & , \text{ if } i, j \text{ are in different community} \end{cases}.$$

Therefore, we can rewrite the Modularity Score in expression (3.10) as

$$\begin{aligned} Q &= \frac{1}{Vol(G)} \sum_{1 \leq i, j \leq n} \left(a_{ij} - \frac{deg(i)deg(j)}{Vol(G)} \right) \frac{s_i s_j + 1}{2} \\ &= \frac{1}{2Vol(G)} \sum_{1 \leq i, j \leq n} \left(a_{ij} - \frac{deg(i)deg(j)}{Vol(G)} \right) s_i s_j. \end{aligned} \quad (3.13)$$

The second equality follows from the fact that

$$\sum_{1 \leq i, j \leq n} a_{ij} = \sum_{i=1}^n deg(i) = Vol(G).$$

Definition 3.2. We define a real $n \times n$ symmetric matrix $B = ((b_{ij}))$ by letting $b_{ij} =$

$a_{ij} = \frac{\deg(i)\deg(j)}{\text{Vol}(G)}$, which is conventionally called "Modularity Matrix".

The Modularity Score Q in (3.13) can be rewritten as

$$Q = \frac{1}{2\text{Vol}(G)} s^T B s. \quad (3.14)$$

We can proceed by writing s as a linear combination of the normalized eigenvectors u_i of B such that

$$s = \sum_{i=1}^n (u_i^T s) u_i$$

The Modularity Score Q can be derived from (3.14) as

$$\begin{aligned} Q &= \frac{1}{2\text{Vol}(G)} \left(\sum_{i=1}^n (u_i^T s) u_i^T \right) B \left(\sum_{i=1}^n (u_i^T s) u_i \right) \\ &= \frac{1}{2\text{Vol}(G)} \sum_{i=1}^n (u_i^T s)^2 \beta_i, \end{aligned} \quad (3.15)$$

where β_i is the eigenvalue of B corresponding to eigenvector u_i .

Without loss of generality, we assume $\beta_1 \geq \beta_2 \geq \dots \beta_n$. In order to maximize the Modularity Q , we should concentrate all the weights on the largest eigenvalue β_1 . That is we need to choose s to be parallel to u_1 . However, each element in s has to be either 1 or -1. So an approximate solution to the maximization of Modularity Q is to let $s_i = 1$ if $u_{1,i} > 0$, and $s_i = -1$ if $u_{1,i} \leq 0$.

Note that if the largest eigenvalue $\beta_1 < 0$, any split will lead to a negative modularity. We will just keep the entire network as a community instead of doing any split.

3.3.3 Split the Network Into More Than Two Communities

The matrix-based method for finding a good division is described in preceding section. Many networks, however, contains more than two communities. The standard approach is recursive division into two in which, we first divide the network into two parts using the method described in previous section, then divide these two parts, and so forth. This procedure can be summarized by following recursive algorithm:

Algorithm 3.1 Community Detection via Modularity Maximization

Input: Network G with Modularity matrix B in Definition 3.2
Community list K

```

1: procedure BiPARTITION( $B$ )
2:   if  $G$  needs to be divided into  $G_1$  and  $G_2$  then
3:     Denote  $B^{(G_1)}$  to be sub-matrix of  $B$  with index in  $G_1$ 
4:     Denote  $B^{(G_2)}$  to be sub-matrix of  $B$  with index in  $G_2$ 
5:     BiPARTITION( $B^{(G_1)}$ )
6:     BiPARTITION( $B^{(G_2)}$ )
7:   else
8:     Add  $G$  into community list  $K$ 
9:   end if
10: end procedure
Output:  $K = [G_1, \dots, G_h]$ .

```

The details about how to decide if a community G of size n_G can be further divided is explained here: Before we split the community G , the overall modularity is $Q_{original}$, and after we split G , while keeping all the other communities unchanged, the modularity becomes Q_{split} . Our strategy is to maximize $Q_{split} - Q_{original}$ by matrix-based algorithm. The division is justified only if $Q_{split} > Q_{original}$. Please note that in the very beginning when there is no division on the entire network, $Q_{original} = 0$. So when splitting a network into two communities, we simply maximize $Q_{split} - 0 = Q_{split}$. It is exactly the

method we described in section 3.3.2. Now we can express additional contribution ΔQ of a further split as

$$\begin{aligned}
\Delta Q &= Q_{split} - Q_{original} \\
&= \frac{1}{Vol(G)} \sum_{i,j \in G} \left(\frac{b_{ij}(s_i s_j + 1)}{2} - b_{ij} \right) \\
&= \frac{1}{2Vol(G)} s^T B^{(G)} s,
\end{aligned} \tag{3.16}$$

where $B^{(G)} = ((b_{ij}^{(G)}))$ is a $n_G \times n_G$ diagonal matrix with elements indexed by the labels of i, j of nodes in community G and having values

$$b_{ij}^{(G)} = b_{ij} - \delta(i, j) \sum_{j \in G} b_{ij}, \tag{3.17}$$

where δ is defined as expression (3.11). If $B^{(G)}$ has any normalized eigenvector β , whose corresponding eigenvalue is positive and also the largest in magnitude, we further split community G by the signs of elements in β . Otherwise, we add G into the community list K .

3.3.4 Resolution Limitation of Modularity Maximization Approach

Although modularity method is very effective in many networks, it has a resolution limit if the network is large. The modularity compares the observed number of in-community

links, and the expected number of in-community links of a random network in which each node has the same degree as the observed network. The random network comes from a global null model, and it implicitly assumes that each node can be connected to any other node. Such assumption is however unreasonable if a network is very large, as the horizon of a node includes only a small part of the network, and ignoring most of it.

In a random network of given degrees for each node, the expected number of links between each pair of nodes i and j is estimated as $\frac{deg(i)deg(j)}{Vol(G)}$, which can become very small when $Vol(G)$, the total number of edges of a network is large. Thus, when the network is large enough, the expected number of links between two groups of nodes may be smaller than 1 under the modularity null model. In that case, even two cliques will be merged as a single cluster by the modularity maximization algorithm, as long as there is a single link between them. However, intuitively, two cliques with only a single link between them should not be treated as a single cluster or community. [Fortunato and Barthélemy \(2007\)](#) proposed some examples in which the resolution limitation problems occurred in modularity method.

In addition to the under-splitting problem described above, the modularity maximization algorithm also has over-splitting problem in some cases. Here is an example. We simulate a random network of 40 nodes from the Erdős-Rényi Random Graph model $G(40, 0.1)$ described in section 2.2. There are no built-in community structure in this simulated network, so an effective network community detection algorithm should not

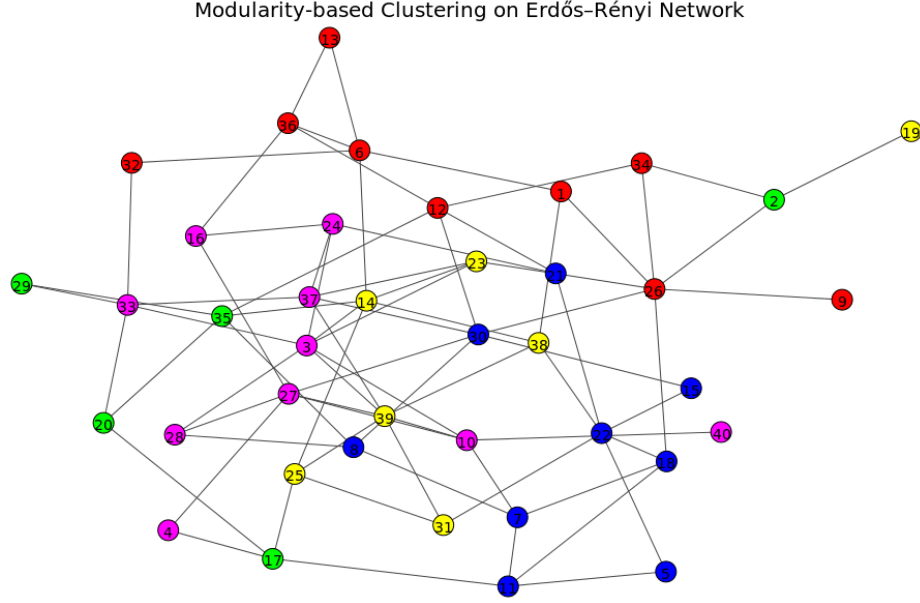


Figure 3.1: A random network simulated from Erdős-Rényi Random Graph model $G(40, 0.1)$. The maximized Modularity Score $Q = 0.3590$ when the network is splitted into 5 communities represented by 5 colors.

split it. However, when we apply the Modularity Maximization algorithm to this completely random network, it does split it into 5 communities. The results is shown in Figure 3.1.

In order to solve the resolution limitation problem, [Reichardt and Bornholdt \(2006\)](#) proposed a multi-resolution method by introducing generalized modularity

$$Q_\lambda = \frac{1}{Vol(G)} \sum_{1 \leq i, j \leq n} (a_{ij} - \lambda \frac{deg(i)deg(j)}{Vol(G)}) \delta(c_i, c_j), \quad (3.18)$$

where works as a resolution parameter, and all other notations are the same as standard modularity in expression (3.10). Unfortunately, [Lancichinetti and Fortunato \(2011\)](#) discovered that the problem for modularity maximization is that it not only inclined to

merge small clusters, but also to break large clusters. It seems basically impossible to avoid both errors simultaneously just by adding a tuning parameter to the modularity measure. The major incentive of our research is to provide a better understanding and possible solution to this problem from statistical point of view. We developed a new network clustering algorithm that enables us to detect the ‘type’ of clusters we exactly want. We also extended this approach to a localized clustering algorithm in chapter 4 that enables us to quantify and minimize the errors of both over-splitting and under-splitting. We stress that if there was statistically significant clusters, our algorithm usually is able to detect them with very small error.

3.4 Intuition of Clique-based Network Clustering

Our strategy to split a network is by creating a measure evaluating how good a clustering is, and then maximizing the quantity on this measure. Researchers have explored many different measures to quantify the quality of a clustering, including the Ratio-Cut, and the Normalized Cut in spectral clustering, and modularity, etc. A clustering algorithm is essentially an optimization procedure over an objective function based on a measure.

People usually have different options on criteria that ensure a good clustering. We start our thinking from an extreme point of view. In Graph Theory, a Clique is a subset of nodes in which any pair of two nodes is connected. A maximal Clique is a Clique that cannot be extended by including one more adjacent node. In some extreme cases, we

may need to require every cluster in a network to be a maximal clique. In other words, within each cluster, every pair of two nodes is connected, and outside each cluster, every node is disconnected with some node inside this cluster. Suppose we extend this idea into more general cases by introducing a new concept called Clique Score for a cluster in the network.

Definition 3.3. *The Clique Score of a cluster or sub-network is defined as the ratio of its internal link count and the internal link count of a clique of same size. Clique Score can be viewed as the internal link density of a cluster or sub-network inside the whole network.*

For example, suppose we have a cluster G of 10 nodes with 18 internal links. The number of links in a clique of 10 nodes is $\frac{10 \times 9}{2} = 45$. Then the Clique Score of G is computed as $\frac{18}{45} = 0.4$. If we treat the entire network as a cluster, its Clique Score will be just its link density. The Clique Score of a clique in a network is $\frac{1}{1} = 1$, which is very intuitive. Our objective here is to design a Clique-based algorithm, such that each cluster has Clique Score above some threshold p , where $0 \leq p \leq 1$, and the weighted average Clique Score of all clusters are maximized.

Now we reexamine an example of the resolution limit problem in modularity method and explain how it can be concurred by our approach. As explained in section 3.3.4, it is possible that two cliques of equal size with only one link connecting them may be merged as one cluster by modularity maximization algorithm as long as the network is

large enough. In our clique-based clustering algorithm, this situation will be guaranteed to be prevented from happening as long as we set the threshold p to be greater than 0.5. In fact, a sub-network of two cliques of size n with only one link between them has $(n-1)n + 1$ links. If we treat such a sub-network as a cluster, its Clique Score will be $\frac{(n-1)n+1}{(2n-1)n} < 0.5$, and for sure this will never be allowed in our clustering algorithm.

The choice of p provides us with the flexibility to define the minimum requirement to form a cluster in a network. In an extreme case if we set $p = 1$, then, all the detected clusters will be a maximal clique. This may not be our desired case, but we can see from this example that we have some control on what type of cluster we really want in our specific application.

3.5 Clique-based Clustering Algorithm

In this section, we introduce our novel Clique-based Clustering Algorithm for community detection.

Definition 3.4. *Suppose we have a social network G of n nodes with adjacency matrix $A = ((a_{ij}))$ defined as in expression (2.1). We define a p -clique in a network to be a random sub-network in which any two nodes are connected with probability p , where $0 \leq p \leq 1$.*

So the expect internal link count in a p -clique of size n_c is $p \binom{n_c}{2}$. In a special case of $p = 1$, any two nodes in the sub-network are connected with probability 1, that is

a 100% clique. So we expect the internal link count to be the largest possible number, which is $\binom{n_c}{2}$.

Suppose a clustering algorithm splits a network G into a list of h clusters G_1, \dots, G_h with cluster size n_1, n_2, \dots, n_h , and interval link counts to be $Vol(G_1), \dots, Vol(G_h)$ respectively. The count of links between two different clusters G_k and G_l are denoted as $Vol(G_k, G_l)$. In order to evaluate this clustering, we compare it with a random network G of n nodes with link density p . The expected link count of G is $Vol(G) = pn(n-1)$. Ideally, we want the internal link count of the h clusters in our clustering arrangement to beat h p-cliques of corresponding size as much as possible, and the number of links between any two clusters to be minimized. This idea can be quantified as maximizing following expression called p -clique index.

Definition 3.5. *Let G be an arbitrary network of n nodes. A clustering algorithm split this network into a list of communities $K = [G_1, \dots, G_h]$. The p -clique Index is a measure of how each of these communities can beat a p -clique of corresponding size in terms of internal link count. It can be represented as follows:*

$$\begin{aligned} D(c, p) &= \frac{1}{n(n-1)} \left(\sum_{k=1}^h \left(Vol(G_k) - \frac{p(n_k-1)n_k}{2} \right) + \sum_{1 \leq k \neq l \leq h} (pn_k n_l - Vol(G_k, G_l)) \right) \\ &= \frac{1}{n(n-1)} \sum_{1 \leq i \neq j \leq n} ((a_{ij} - p)\delta(c_i, c_j) + (p - a_{ij})(1 - \delta(c_i, c_j))), \end{aligned} \quad (3.19)$$

where c_i denotes the community of node i , and δ denotes the Kronecker delta symbol defined in expression (3.11).

Table 3.1: Reward table for the clique-based clustering with parameter p

Reward Table	Connected Pair of Nodes	Disconnected Pair of Nodes
Pair of Nodes in Same Cluster	$1 - p$	$-p$
Pair of Nodes in Different Clusters	$-1 + p$	p

Also note that the cluster size n_k may change as the clustering change. Intuitively, we have set up an objective for our clustering called p -clique index. The contribution of each pair of node to the p -clique index can be summarized in the reward Table 3.1

Above objective function, although similar to that of fitting a stochastic block model, is actually very different. We are not fitting our data to a model with p -clique as a cluster; instead we are trying to beat it in our clustering arrangement. In fact, it is possible that in our clustering, the internal link count is higher than those p -cliques.

Note that if $p = 1$, D will never be positive, because a network with each cluster to be a clique has the highest possible density of internal link.

Similar to Modularity approach, we will design a hierarchical clustering algorithm, and returns a clustered network in which each cluster has a Clique Score higher than user-specified threshold p .

3.5.1 Split the Network Into Two Communities

We first consider splitting the network into two communities. Let s be an n -dimensional vector indicating the community membership of each node in the network. Let $s_i = 1$

if node i belongs to community 1, and $s_i = -1$ if node i belongs to community 2 as expression (3.12). The p -Clique Index in (3.19) can be expressed as

$$\begin{aligned} D(s, p) &= \frac{1}{n(n-1)} \sum_{1 \leq i \neq j \leq n} (a_{ij} - p) s_i s_j \\ &= \frac{1}{n(n-1)} s^T C(p) s, \end{aligned} \quad (3.20)$$

where

$$C(p) = A - p(J_n - I_n) \quad (3.21)$$

is called as p -clique matrix, J_n is an $n \times n$ matrix, in which every element equals 1, and I_n is n -dimensional identity matrix.

We hope to maximize the p -clique Index $D(s, p)$ over s . This is equivalent to maximizing

$$\begin{aligned} \max_s (D(s, p) - D(\mathbb{1}_n, p)) &= \max_s (s^T C(p) s - \mathbb{1}_n^T C(p) \mathbb{1}_n) \\ &= \frac{1}{n(n-1)} \max_s (s^T (C(p) - (\frac{1}{n} \sum_{i,j=1}^n C(p)_{ij}) I_n) s) \\ &= \frac{1}{n(n-1)} \max_s (s^T C(p)^{(0)} s), \end{aligned} \quad (3.22)$$

where $C(p)^{(0)} = C(p) - (\frac{1}{n} \sum_{i,j=1}^n C(p)_{ij}) I_n$.

Unfortunately, this is an NP hard problem. However, if we relax the problem to allow s to be any normalized real value vector, the maximum value of D can be achieved

when s equals the eigenvector β of $C(p)^{(0)}$ with the largest eigenvalue λ . Therefore,

$$s_i = \begin{cases} 1, & \text{if } \beta_i \geq 0 \\ -1, & \text{if } \beta_i < 0 \end{cases}, \forall i = 1, \dots, n$$

is a natural approximation to the solution of this clustering problem.

1. If $\lambda \leq 0$, and $D(\mathbb{1}_n, p) \geq 0$, that is $D(\mathbb{1}_n, p) \geq \max(D(s, p), 0)$, we cannot make any further improvement by the split, but the entire network itself is already a good cluster. In fact if $D(\mathbb{1}_n) = \frac{1}{n(n-1)}(\sum_{1 \leq i \neq j \leq n} (a_{ij} - p(J_n - I_n)_{ij})) \geq 0$, the link density of the entire network is higher than p , that is how we quantify what is a good cluster, so we do not need to do any further clustering on this case.
2. If $\lambda > 0$, or $D(\mathbb{1}_n, p) < 0$, that is $D(\mathbb{1}_n, p) \leq \max(D(s, p), 0)$, a split of the entire network is needed. It is easy to understand that we can make improvement in p -clique index by the split induced by s , if $D(\mathbb{1}_n, p) \leq D(s, p)$. However, it is worth noting that we should still make the split even if $D(\mathbb{1}_n, p) \geq D(s, p)$ as long as $D(\mathbb{1}_n, p) \leq 0$. In fact, if $D(\mathbb{1}_n, p) \leq 0$, the entire network has link density lower than p . In this specific case, an extreme split c , which treats every single node a

cluster will make

$$\begin{aligned} D(c, p) &= \frac{1}{n(n-1)} \sum_{1 \leq i, j \leq n} (-a_{ij} + p) \\ &= -D(1_n, p) \geq 0 \geq D(1_n, p). \end{aligned}$$

In other words, as long as we keep making further splits in each cluster of s , we will end up with some clustering s_{final} , such that $D(s_{final}) > D(1_n)$. So $D(1_n)$ is not the maximum of p -clique index, and we should make the splits. This property is important, we can see it in next section 3.5.2 that every cluster will have a Clique Score higher than p . Therefore, we can have a control of the “quality” of clusters.

3.5.2 Split the Network Into More Than Two Communities

Similar to 3.3.3, some networks may contain more than two clusters. Our strategy here is still recursive division into two: we first divide the network into two parts using the method described in section 3.5.1, then divide these two parts, and so forth. Similar to (3.19), we denote

$$C(p)^{(G)} = a - p(J_n - I_n),$$

to be the p -clique matrix of network G , in which A is its adjacency matrix. This procedure can be summarized by following recursive Algorithm 3.2.

The details about applying BiPartition procedure to a sub-network G of size n is

Algorithm 3.2 Community Detection via p -clique Index Maximization

Input: p -clique matrix $C(p)^{(G)}$ of Network G ,
Community List K

```

1: procedure BiPARTITION( $G, C(p)^{(G)}$ )
2:   Compute eigenvector  $s$  of  $C(p)^{(G)}$  with largest eigenvalue
3:   Split  $G$  into  $G_1$  and  $G_2$  by the sign of  $s$ 
4:   Compute additional contribution  $\Delta D(p)$  to  $p$ -clique Index from split of  $G$ 
5:   if  $\Delta D(p) > 0$  or  $\sum_{i,j \in G} C(p)_{ij}^{(G)} < 0$  then
6:     BiPARTITION( $G_1, C(p)^{(G_1)}$ )
7:     BiPARTITION( $G_2, C(p)^{(G_2)}$ )
8:   else
9:     Add  $G$  into community list  $K$ 
10:  end if
11: end procedure
Output: Community list  $K = [G_1, \dots, G_h]$ .

```

explained as follows.

1. For the step (3) of the Algorithm 3.2, before we split the sub-network G , the current clustering arrangement is called $c_{current}$, and the overall p -clique index is $D(c_{current}, p)$ in expression (3.19). After we split G into G_1 and G_2 , while keeping all the other clusters unchanged, the clustering arrangement is called c_{split} , and the corresponding p -clique index becomes $D(c_{split}, p)$ in expression (3.19). Our strategy is to maximize $D(c_{split}, p) - D(c_{current}, p)$ by matrix-based algorithm. We

can express additional contribution ΔD of a further split as

$$\begin{aligned}
\Delta D(p) &= D(c_{split}, p) - D(c_{current}, p) \\
&= \frac{1}{n(n-1)} \sum_{i,j \in G} ((a_{ij} - p)\delta(c_i, c_j) + (p - a_{ij})(1 - \delta(c_i, c_j))) \\
&\quad - \frac{\sum_{i,j \in G} (a_{ij} - p)}{n(n-1)} \\
&= \frac{1}{n(n-1)} \sum_{i,j \in G} (s_i C(p)_{ij}^{(G)} s_j - C(p)_{ij}^{(G)}), \tag{3.23}
\end{aligned}$$

where $s_i = 1$ if node i belongs to G_1 , $s_i = -1$ if node i belongs to G_2 , $C(p)^{(G)}$ is the p -clique matrix of G . The maximization of D is equivalent to maximization of $s^T C(p)^{(G)} s$, which is an NP hard problem. If we relax the problem to let s be any real vector of ℓ^2 norm of 1, the solution is $s = \beta^{(G)}$, where $\beta^{(G)}$ is the eigenvector of $C(p)^{(G)}$ with largest eigenvalue. This leading eigenvector can be obtained by the Implicitly Restarted Lancos Method introduced in [Calvetti et al. \(1994\)](#). Therefore, an approximate solution to our maximization problem is let

$$s_{is} = \begin{cases} 1 & , \text{ if } \beta_i^{(G)} \geq 0 \\ 0 & , \text{ if } \beta_i^{(G)} < 0 \end{cases}.$$

2. For the step (5) of the Algorithm [3.2](#), If $\delta D(p) > 0$ or $\sum_{i,j \in G} C(p)_{ij}^{(G)} < 0$, we need to use the sign s to split G . It is straightforward that if $\Delta D(p) > 0$, we can increase the p -clique index $D(p)$, so a split of G is needed. Similar to section 3.5.1,

it is worth noting that if $\sum_{i,j \in G} C(p)_{ij}^{(G)} < 0$, we should still make a split to G even when $\Delta D(p) \leq 0$. In this case, it is true that we will not increase the p -clique index $D(p)$ at this step, but eventually, the contribution of G to p -clique Index will be at least $\frac{1}{n(n-1)} \sum_{i,j \in G} (-C(p)_{ij}^{(G)}) > 0$, as long as we keep further dividing it into more clusters. This can be seen from two extreme cases. If G is totally divided, every single node in G is a cluster by itself, the contribution of G to p -clique index will be exactly $\frac{1}{n(n-1)} \sum_{i,j \in G} (-a_{ij} + p) > 0$. On the other hand, however, if we leave G undivided, the contribution of G to p -clique Index will be $\frac{1}{n(n-1)} \sum_{i,j \in G} (a_{ij} - p) < 0$. Therefore, it is clear that we should not leave G undivided if $\sum_{i,j \in G} (a_{ij} - p) < 0$.

3. For the step (8) of the algorithm, If $\Delta D(p) \leq 0$ and $\sum_{i,j \in G} (a_{ij} - p) \geq 0$, we cannot increase the contribution of G to the p -clique index by splitting it into two. At this time, the Clique Score of G is higher than p , and we should be satisfied with it.

3.5.3 Summarize Clique-based Algorithm

The most important feature of our algorithm is the guarantee that all clusters will have a higher Clique Score than the user defined parameter p . This feature is very useful in applications of finding closest friends for people in a network. Other algorithms may also find good clusters, but they usually have no control on the ‘quality’ of the clusters, and therefore the clusters may have unsatisfactory low Clique Score. Sometimes, we

are not interested in very small clusters such as size less than 5, and we can simply remove them from the cluster list. When using our algorithm, we should also be aware that clusters of Clique Score lower than parameter p maybe splitted. In applications without specific requirements for Clique Score of clusters, this is called the error of over-splitting. In Chapter 4, we provide detailed instructions about how to choose parameter p to minimize this error and the error of under-splitting.

3.6 Simulated Network Example

In this section, we will use stochastic block model from [Nowicki and Snijders \(2001\)](#) to simulate different random networks. This model was described in detail in section 2.3. Since the clustering structure in these networks are clear, they provide us the ground truth from which, we can use them to test the validity of our clustering algorithm.

3.6.1 Simulation From Stochastic Block Model

Our objective is to simulate a random network of n nodes with h communities with pre-specified structure and link density. The Stochastic Block Model described in section 2.3 may be used for this purpose. The only difference is that the size of each of the h communities are fixed as $[n_1, \dots, n_h]$, and we assign every node i randomly into one of these h communities, and denote its membership as $c(i)$. We also specify an $h \times h$ symmetric probability matrix B . Given the specified community membership $c =$

$[c(1), \dots, c(h)]$, and probability matrix B , every off-diagonal upper triangular entry a_{ij} of the adjacency matrix A can be simulated independently from $\text{Bernoulli}(B_{c(i),c(j)})$, where $i < j$. We fill out the rest of entries of matrix A by let

$$a_{ij} = \begin{cases} a_{ji} & , \text{ if } j < i \\ 1 & , \text{ if } j = i \end{cases}.$$

To be specific, the diagonal entries of B represents the probability of two nodes to be connected in that corresponding cluster, which is also the desired Clique Score; while the off-diagonal entries of B represents the probability of two nodes to be connected in corresponding different two clusters, which is also the inter-cluster link density. Furthermore, we need to require that $B_{kl} \leq \min(B_{kk}, B_{ll})$, otherwise, either k or l can not be called a cluster any more.

We can use simulated random networks from Stochastic Block Model to test the performance of our Clique-based clustering algorithm. The performance can be measured by the Normalized Mutual Information(NMI) of [Fred and Jain \(2003\)](#) introduced in expression (1.5).

In order to better examine the type of errors our algorithm may make, we also define our own measure in this way: for a sampled network of size n , create a $n \times n$ matrix $E^{(1)}$ of binary entries, such that $E_{ij}^{(1)} = 1$ if node i and j are grouped into the same community in our algorithm and $E_{ij}^{(1)} = 0$ vice versa. Similarly, we can also create

matrix $E^{(2)}$ from the initial community assignment in the stochastic block model. If $E^{(1)} = E^{(2)}$, our clustering result is match the correct answer exactly. If $E^{(1)} \neq E^{(2)}$, the proportion p of different off-diagonal entries between $E^{(1)}$ and $E^{(2)}$

$$error = \sum_{1 \leq i \neq j \leq n} \frac{|E_{ij}^{(1)} - E_{ij}^{(2)}|}{n(n-1)} \quad (3.24)$$

is the ‘distance’ we use to measure the closeness of our clustering with the correct answer.

We may also want to have more detailed evaluation of the clustering. For example, we can compare $E^{(1)}$ and $E^{(2)}$ part by part, instead of as a whole. In this case, indexes of the same initial cluster assignment in the stochastic block model are grouped into one block in both $E^{(1)}$ and $E^{(2)}$, and we will end up with two $h \times h$ block matrices. Therefore, we can compute the proportion of different off-diagonal entries between $E^{(1)}$ and $E^{(2)}$ block by block. The measure we use is an $h \times h$ proportion matrix P , with entries

$$P_{kl} = \sum_{i \neq j, i \in k, j \in l} \frac{|E_{ij}^{(1)} - E_{ij}^{(2)}|}{f_{kl}}, \quad (3.25)$$

where

$$f_{kl} = \begin{cases} n_k n_l & , \text{ if } k \neq l \\ n_k(n_k - 1) & , \text{ if } k = l \end{cases}$$

represents the number of different pair (i, j) such that $i \neq j, i \in k, j \in l, k, l$ represents for index of initial cluster assignment, i, j represents for index of nodes in the network.

Note that our clustering algorithm only has one global parameter p , which is the minimum requirement for the Clique Score of all clusters. However, when this recursive algorithm is working on a small sub-network, it simply focus on the structure of this local sub-network, and the clustering results are always the same no matter what the rest of the entire network looks likes. Therefore, our algorithm is more localized than the Modularity methods, in which the clustering on each sub-network always depends on the entire network link count. Therefore, even some of the following sample networks are small and simple, the clustering results would still be valid, if the small and simple network exists as a sub-network of a much larger and more complex network.

3.6.2 Simulated Network 1

The first stochastic block model we want to consider has three built-in clusters with size 100, 10, and 10, and the within/between cluster link probability matrix B in Table 3.2. It is denoted as *SBM1*.

Table 3.2: Parameters of *SBM1* for simulation of random network 1.

cluster ID: size	<i>no.1</i> : 100	<i>no.2</i> : 10	<i>no.3</i> : 10
<i>no.1</i> : 100	0.2	0.05	0.05
<i>no.2</i> : 10	0.05	0.5	0.05
<i>no.3</i> : 10	0.05	0.05	0.5

Table 3.2 shows all the information we need to simulate this sample network from stochastic block model. For example, the internal link probability in cluster 1 is 0.2, in cluster 2 and 3 are both 0.5 , and the inter-cluster link probability between cluster

2, and cluster 3 is 0.05. Here we set the global threshold parameter $p = 0.11$, which is significantly lower than the internal link probability of each cluster, but significantly higher than the inter-cluster link probability of each pair of cluster. We will go over in great details in chapter 4 about how to choose the parameter p , and what does the word ‘significantly’ mean in the previous sentence. In this section, we want to get an idea of the performance of our algorithm in simulated networks.

We simulate 100 random networks by stochastic block model using above parameters. For each of the 100 simulated networks, we use our clustering algorithm to get a cluster list, and get the average error rate estimates in expression (3.24) and (3.25), and their corresponding standard error. The estimate of NMI in expression (1.5) is $\widehat{NMI} = 0.9194$ with $s.e.(\widehat{NMI}) = 0.0065$, and the overall error rate in (3.24) is $\widehat{error} = 1.83\%$, with $s.e.(\widehat{error}) = 0.16\%$. So, in general, our clustering results are fairly close to the correct answer.

The estimate of block-wise error matrix P in (3.25) with standard error is summarized in table 3.3. From table 3.3, we can see that the clustering of pair of nodes within initial

Table 3.3: Apply Algorithm 3.2 with $p = 0.11$ to 100 simulated networks from $SBM1$ with parameters in Table 3.2.

cluster ID: size	$no.1 : 100$	$no.2 : 10$	$no.3 : 10$
$no.1 : 100$	$\hat{P}_{11} = 2.01\%$, $s.e.(\hat{P}_{11}) = 0.20\%$	$\hat{P}_{12} = 0.83\%$, $s.e.(\hat{P}_{12}) = 0.29\%$	$\hat{P}_{13} = 1.14\%$, $s.e.(\hat{P}_{13}) = 0.28\%$
$no.2 : 10$	$\hat{P}_{21} = 0.85\%$, $s.e.(\hat{P}_{21}) = 0.29\%$	$\hat{P}_{22} = 6.44\%$, $s.e.(\hat{P}_{22}) = 1.26\%$	$\hat{P}_{23} = 6.14\%$, $s.e.(\hat{P}_{23}) = 1.72\%$
$no.3 : 10$	$\hat{P}_{31} = 1.14\%$, $s.e.(\hat{P}_{31}) = 0.28\%$	$\hat{P}_{32} = 6.14\%$, $s.e.(\hat{P}_{32}) = 1.72\%$	$\hat{P}_{33} = 4.42\%$, $s.e.(\hat{P}_{33}) = 1.06\%$

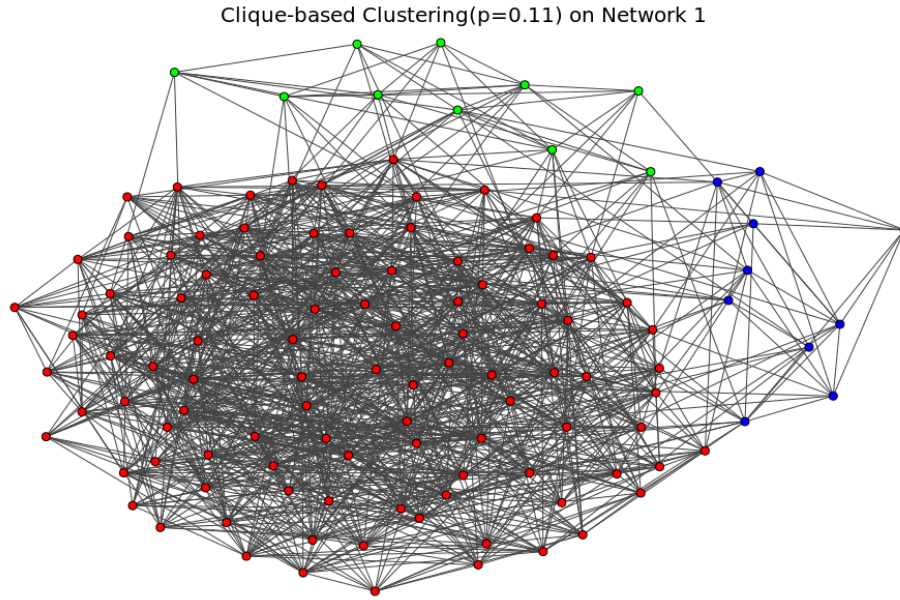


Figure 3.2: A random network simulated from $SBM1$ with parameters defined in Table 3.2. This network is splitted into three communities by our clique-based clustering algorithm. It is matching with the community structure defined in the stochastic block model. These three communities are displayed in three different colors: red, blue and green.

cluster 1, between cluster 1 and cluster 2, or between cluster 1 and cluster 3 are very good with less than 2.5% error rate on average. However, For pair of nodes within cluster 2, or within cluster 3, or between cluster 2, and cluster 3, we do have a higher error rate up to 7%. In other words, sometimes, our clustering algorithm will miss-classify nodes in cluster 2 with nodes in cluster 3. All these estimates telling us that the clustering in small cluster 2 or cluster 3 are not as good nor stable as in a big cluster like cluster 1. This is making sense, because the smaller the size of the cluster, the less information we have about it, and it will be more difficult to detect the correct boundary of the true cluster because of the random variation of the network.

For visualization purpose, we also plot the clustering results for one of the network simulated from stochastic block model in Figure 3.2. In this particular simulated random network, our clique-based algorithm correctly ‘discovered’ all the built-in three communities: cluster 1 in red, cluster 2 in green, and cluster 3 in blue.

3.6.3 Simulated Network 2

The second stochastic block model we want to consider has four built-in clusters with size 800, 400, 50 and 20, and the within/between cluster link probability matrix B in Table 3.4. We denote it as $SBM2$

Table 3.4: Parameters in $SBM2$ for simulation of random network 2.

cluster ID:size	<i>no.1</i> : 800	<i>no.2</i> : 400	<i>no.3</i> : 50	<i>no.4</i> : 20
<i>no.1</i> : 800	0.1	0.04	0.01	0.01
<i>no.2</i> : 400	0.04	0.15	0.01	0.01
<i>no.3</i> : 50	0.01	0.01	0.4	0.02
<i>no.4</i> : 20	0.01	0.01	0.02	0.6

We choose the within/between cluster link probability that best mimic real network. In real life network, the average count of link(the degree) of a node can not increase to a very large number, but the maximum total number of links in a network may increase in order of $O(n^2)$. Therefore, in stochastic block model, we set the internal link of cluster 1 of size 800 to be as low as 0.1, but cluster 4 of size 20 to be as high as 0.6. Further more, the inter-cluster link probability between a large cluster like cluster 1 and a small cluster like cluster 4 are set to be low. In fact, even if this probability is set to be as low as 0.01, we may expect each node in cluster 4 are connected with 12 nodes in cluster 1 and

2, which is more than the expected internal link count of 11.4 in cluster 4. Therefore, detecting cluster 4 from a random network sample may not be easy.

As in section 3.6.1, we simulate 100 sample of network from stochastic block model with above parameters, and apply our clustering algorithm with $p = 0.06$. The estimate of NMI in expression (1.5) is $\widehat{NMI} = 0.9988$ with $s.e.(\widehat{NMI}) = 0.0002$, and estimate of overall error rate in expression (3.24) is $\widehat{error} = 6.40610 \times 10^{-3}\%$ with $s.e.(\widehat{error}) = 2.22610 \times 10^{-3}\%$. In general, we can say that the found clusters and real clusters are almost matching exactly. The estimate of block wise error rate P in expression (3.25) with standard error is summarized in Table 3.5.

Table 3.5: Apply p -clique based Algorithm 3.2 with $p = 0.06$ to 100 simulated networks from $SBM2$ with parameters in Table 3.4.

cluster ID:size	$no.1 : 800$	$no.2 : 400$	$no.3 : 50$	$no.4 : 20$
$no.1 : 800$	\hat{P}_{11} $= 1 \times 10^{-2}\%$, $s.e.(\hat{P}_{11})$ $= 4.9 \times 10^{-3}\%$,	\hat{P}_{12} $= 1.25 \times 10^{-3}\%$, $s.e.(\hat{P}_{12})$ $= 1.24 \times 10^{-3}\%$,	\hat{P}_{13} $= 0\%$ $s.e.(\hat{P}_{13})$ $= 0\%$	\hat{P}_{14} $= 0\%$ $s.e.(\hat{P}_{14})$ $= 0\%$
$no.2 : 400$	\hat{P}_{21} $= 1.25 \times 10^{-3}\%$, $s.e.(\hat{P}_{21})$ $= 1.24 \times 10^{-3}\%$,	\hat{P}_{22} $= 0\%$, $s.e.(\hat{P}_{22})$ $= 0\%$,	\hat{P}_{23} $= 0\%$ $s.e.(\hat{P}_{23})$ $= 0\%$	\hat{P}_{24} $= 0\%$ $s.e.(\hat{P}_{24})$ $= 0\%$
$no.3 : 50$	\hat{P}_{31} $= 0\%$, $s.e.(\hat{P}_{31})$ $= 0\%$,	\hat{P}_{32} $= 0\%$, $s.e.(\hat{P}_{32})$ $= 0\%$,	\hat{P}_{33} $= 0.71\%$ $s.e.(\hat{P}_{33})$ $= 0.21\%$	\hat{P}_{34} $= 0.15\%$ $s.e.(\hat{P}_{34})$ $= 0.11\%$
$no.4 : 20$	\hat{P}_{41} $= 0\%$, $s.e.(\hat{P}_{41})$ $= 0\%$,	\hat{P}_{42} $= 0\%$, $s.e.(\hat{P}_{42})$ $= 0\%$,	\hat{P}_{43} $= 0.15\%$ $s.e.(\hat{P}_{43})$ $= 0.11\%$	\hat{P}_{44} $= 0.85\%$ $s.e.(\hat{P}_{44})$ $= 0.55\%$

From summary Table 3.5, our clustering algorithm makes almost no mistakes, in 100

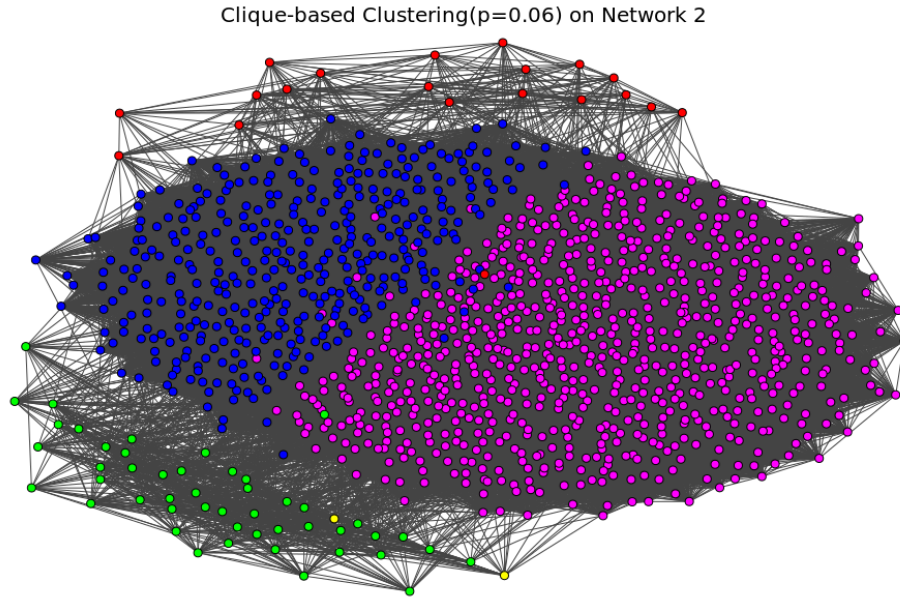


Figure 3.3: A random network simulated from $SBM2$ with parameters defined in Table 3.3. This network is splitted into four communities by our clique-based clustering algorithm. It is matching with the community structure defined in the stochastic block model with minor errors. These four communities are displayed in four different colors: red, blue and green and pink.

simulated random network samples in each block of cluster pair. However, we should still be note that, the error rate in small clusters like cluster 3 or cluster 4 are slightly higher than in large clusters like cluster 1 or cluster 2. This observation is similar as our first example.

For visualization purpose, we also plot the clustering results for one of the network simulated from stochastic block model in Figure 3.3. Our clique-based clustering algorithm discovered the 4 built-in clusters: cluster 1 in pink, cluster 2 in blue, cluster 3 in green, and cluster 4 in red.

Table 3.6: Parameter in *SBM3* for simulation of random network 3

cluster ID:size	<i>no.1</i> : 3000	<i>no.2</i> : 2000	<i>no.3</i> : 1000	<i>no.4</i> : 400	<i>no.5</i> : 200	<i>no.6</i> : 100	<i>no.7</i> : 100	<i>no.8</i> : 100	<i>no.9</i> : 80	<i>no.10</i> : 20
<i>no.1</i> : 3000	0.08	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005
<i>no.2</i> : 2000	0.005	0.09	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005
<i>no.3</i> : 400	0.005	0.005	0.1	0.005	0.005	0.005	0.005	0.005	0.005	0.005
<i>no.4</i> : 200	0.005	0.005	0.005	0.15	0.005	0.005	0.005	0.005	0.005	0.005
<i>no.5</i> : 100	0.005	0.005	0.005	0.005	0.2	0.005	0.005	0.005	0.005	0.005
<i>no.6</i> : 100	0.005	0.005	0.005	0.005	0.005	0.25	0.005	0.005	0.005	0.005
<i>no.7</i> : 100	0.005	0.005	0.005	0.005	0.005	0.005	0.25	0.005	0.005	0.005
<i>no.8</i> : 100	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.25	0.005	0.005
<i>no.9</i> : 3000	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.3	0.005
<i>no.10</i> : 3000	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.005	0.7

3.6.4 Simulated Network 3

The third stochastic block model we want to consider is relatively larger with 7000 nodes and 10 clusters of size 3000, 2000, 1000, 400, 200, 100, 100, 100, 80, 20. The within/between cluster link probability matrix B is represented in Table 3.6. It is denoted as *SBM3*.

Because of the large size of above network, we only simulate one sample, and apply our clustering algorithm to it with parameter $p = 0.06$. It correctly detected cluster 1 through 9 exactly, but split cluster 10 into 2 clusters of size 13 and 3, and 4 single nodes. It has been investigated that the reason for this mis-clustering is due to the approximation we use in optimization of p -clique index. In other words, the maximum of p -clique index is not achieved because we approximate the eigenvector of largest

eigenvalues by a vector of entries of only 1's and -1 's. The NMI in expression 1.5 is $NMI = 0.9988$, and the overall error rate in expression (3.24) is $error = 4.4510 \times 10^{-4}\%$. This results shows the general validity of our clustering algorithm. Since it is hard to plot a network with 7000 nodes, we will not include a visualization of the clustering result.

Chapter 4

Localized Community Detection

4.1 Introduction

In Chapter 3, we reviewed two commonly used optimization-based network clustering algorithms, the spectral clustering method and Modularity Maximization method, and discussed their advantages and disadvantages. We also introduced a novel Clique-based clustering algorithm that provides guarantee on the internal link density of each community we detect. The clustering results on simulated random network is satisfactory as described in section 3.6. This algorithm is especially useful in applications when we have a specific requirements what every community should look like, such as finding out friendship communities in Facebook with internal link density higher than 20%.

In essence, the minimum link density threshold parameter p is a user controlled tool for a balance between “over-splitting” and “under-splitting”. In some applications, we

have no specific requirements on the link density of every community. In this situation, we need to understand how to choose an optimal value for parameter p . It is similar to the generalized Modularity method introduced by [Reichardt and Bornholdt \(2006\)](#). It is possible that an optimal value of parameter p may not exist at all, just as [Lancichinetti and Fortunato \(2011\)](#) explained in their paper about the limitation of Modularity-based method. A better solution is to adopt a localized clustering strategy to modify our clique-based clustering algorithm. We may consider choosing different values for parameter p in different parts of the network.

The rest of this chapter is organized as follows: In section 4.2, we apply our clique-based algorithm on both real world and simulated data sets with different values for parameter p , and explain the relationship between p and clustering results. In section 4.3, we explain how to choose parameter p to minimize the risk of “over-splitting”, and “under-splitting”. In section 4.4, we develop a fully localized algorithm for large scale social network clustering. In section 4.5 we apply our localized algorithm on some simulated random network and discuss the results.

4.2 Effects of Threshold Parameter p

In order to figure out a strategy about how to choose the threshold parameter p for our clique-based clustering algorithm introduced in chapter 3, we need to run some experiments to explore the effects of threshold parameter p on clustering results. In this

section, we will apply our clique-based clustering algorithm on both real world network data and simulated data sets.

4.2.1 Karate Club Data Set

In this chapter, we will use the well-known Zachary karate club dataset to test our Clique-based clustering algorithm using different values of threshold parameter p . This dataset was prepared by [Zachary \(1977\)](#), and we have used it to test our model-based clustering algorithm in section 2.4.3. There are 34 members and 78 links among them in this Karate Club. An advantage of using this dataset is that this karate club was splitted into two groups after a conflict, so we can compare our clustering results with the “ground truth”.

The link density of the network is $\frac{156}{34 \times 33} = 0.139$. If we set parameter $p > 0.139$, the algorithm will automatically looking for clusters with Clique Scores higher than p . In case if this network is a completely random network without any cluster structure of Clique Score higher than 0.139, it will be finally splitted into single node clusters or very small clusters, which is easy to form even in a random network setting. If we set parameter $p < 0.139$, in theory, we will be able to avoid splitting the network, when it is a completely random(Erdős-Rényi Network). Furthermore, we will also be able to detect any cluster structure with Clique Score higher than p .

We will apply our clique-based clustering algorithm to this dataset for threshold parameter $p = 0.15$ versus $p = 0.1$, and plot both results in [Figure 4.1](#) and [Figure 4.2](#)

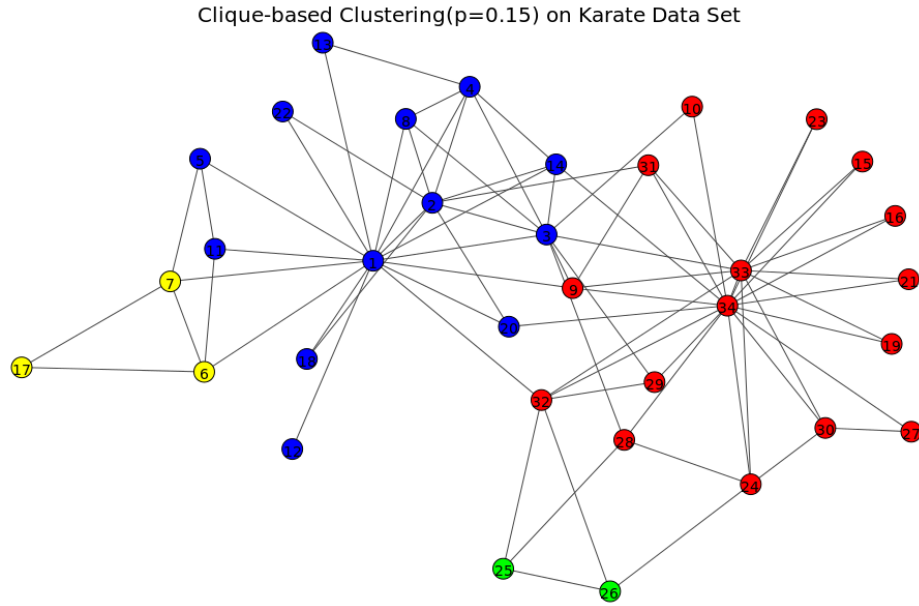


Figure 4.1: Clique-based clustering Algorithm 3.2 with $p = 0.15$ is applied to Karate Club network. It splits the network into 4 communities denoted by four colors: yellow, blue, red, and green.

for comparison.

We summarize the clustering results for $p = 0.15$ in Table 4.1. We can see that

Table 4.1: Karate Club network clustering summary by clique-based clustering Algorithm 3.2 with $p = 0.15$

Cluster ID	Karate Club Member ID	Cluser Clique Score
1	25,26	1.0
2	9, 10, 15,16, 19,21, 23, 24, 27, 28, 29, 30, 31, 32, 33, 34	0.25
3	6,7,17	1.0
4	1, 2, 3, 4, 5, 8, 11, 12, 13, 14, 18, 20, 22	0.33

four clusters are detected, and all of them have Clique Score much higher than the parameter $p = 0.15$. Since cluster 1, and cluster 3 are too small, we may simply treat them as isolated members from the bulk of the club. The cluster 2, and cluster 4,

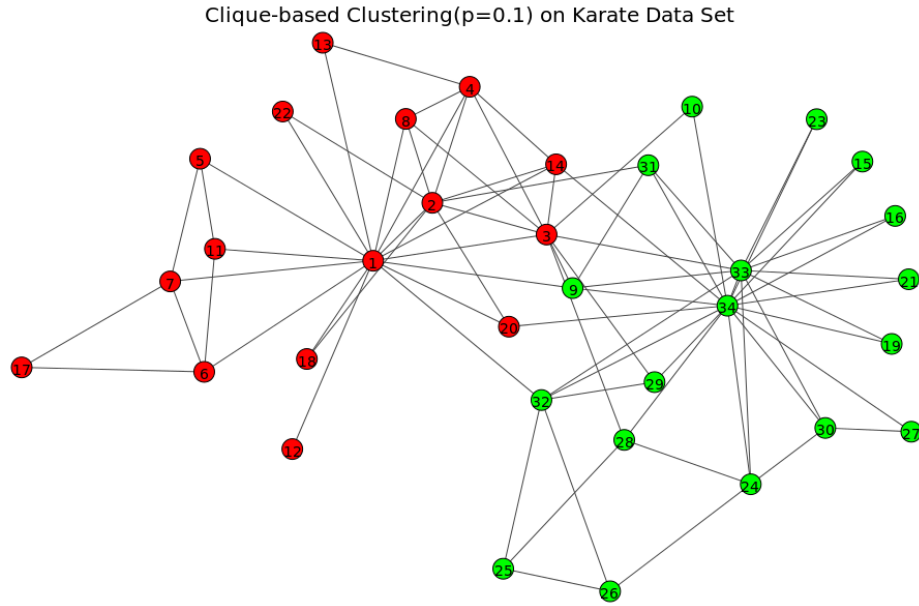


Figure 4.2: Clique-based clustering Algorithm 3.2 with $p = 0.1$ is applied to Karate Club network. It splits the network into 2 communities denoted by two colors: red and green.

however, are large enough to be a significant cluster. They illustrates the fact that the club are splitted into two groups after a conflict.

We summarize the clustering results for $p = 0.1$ in Table 4.2. The clustering result is

Table 4.2: Karate Club network clustering summary by clique-based clustering Algorithm 3.2 with $p = 0.1$

Cluster ID	Karate Club Member ID	Cluster Clique Score
1	9, 10, 15, 16, 19, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34	0.2288
2	1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 17, 18, 20, 22	0.2750

matching the real split of club very well except for that node 9 was mis-classified. This result is exactly the same as our model-based clustering algorithm described in section 2.4.3. Furthermore, both clusters also have a much higher Clique Score than p . We may

also compare this result with the result of last trail when $p = 0.15$. The cluster 1 for $p = 0.10$ is the merge of cluster 1 and cluster 2 for $p = 0.15$. The cluster 2 for $p = 0.10$ is the merge of cluster 3 and cluster 4 for $p = 0.15$.

4.2.2 Dolphin Network Data

In this chapter, we will use the Dolphin network dataset to test our clique-based clustering algorithm using different values of threshold parameter p . This dataset was constructed from observations of a group of 62 bottlenose dolphins living in Doubtful Sound, New Zealand, over a period of seven years from 1994 to 2001 by [Lusseau et al. \(2003\)](#). We have used it to test our model-based clustering algorithm in section 2.4.4. There are 62 members and 318 links among them in this Dolphin network. The link density of the entire network is $\frac{318}{62 \times 61} = 0.084$.

First, we want to test if this network has any community structure, and we can set $p = 0.03 < 0.084$. If the network with link density 0.084 has no community structure, such as Erdős-Rényi network, our algorithm with $p = 0.03$ should not split it. We may also want to look closer to the network, and detect clusters with higher Clique Scores. In this case, we set $p = 0.20$, and apply our algorithm to this dolphin network. We plot the clustering results using both $p = 0.03$ and $p = 0.20$ in [Figure 4.3](#) and [Figure 4.4](#) for comparison.

We summarize the clustering results for $p = 0.03$ in [Table 4.3](#). We can see from above results that the dolphin network do have two cluster structures, both have Clique Score

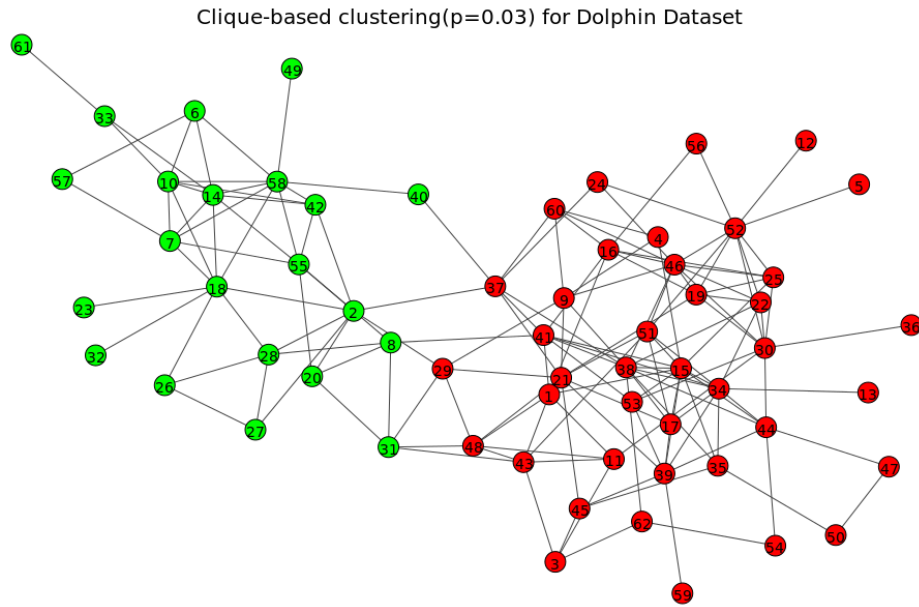


Figure 4.3: Clique-based clustering Algorithm 3.2 with $p = 0.035$ is applied to Dolphin network. It splits the network into 4 communities denoted by two colors: red, and green.

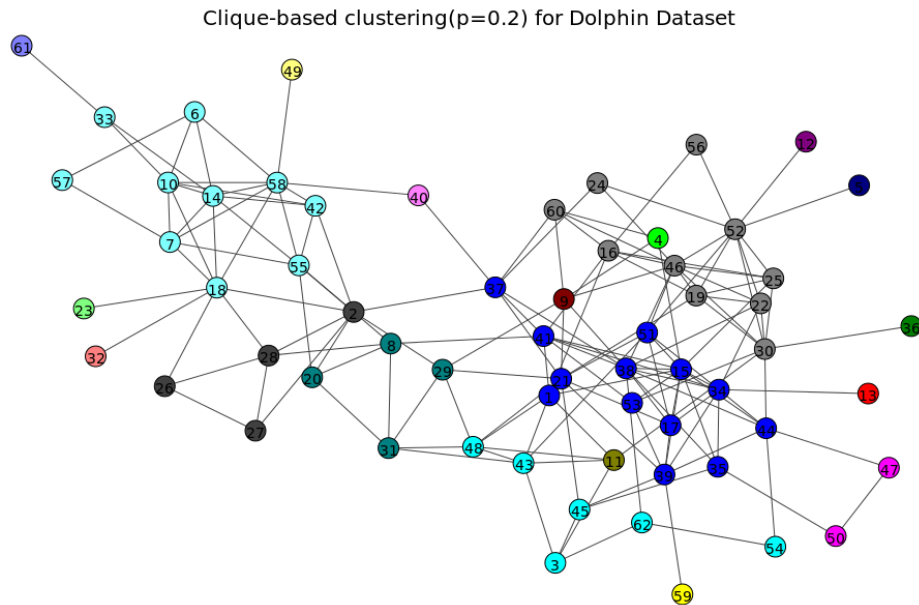


Figure 4.4: Clique-based clustering Algorithm 3.2 with $p = 0.2$ is applied to Dolphin network. It splits the network into 6 major communities of size greater than 5.

Table 4.3: Dolphin network clustering summary by clique-based clustering Algorithm 3.2 with $p = 0.03$

Cluster ID	Dolphin ID	Cluster Clique Score
1	1, 3, 4, 5, 9, 11, 12, 13, 15, 16, 17, 19, 21, 22, 24, 25, 29, 30, 34, 35, 36, 37, 38, 39, 41, 43, 44, 45, 46, 47, 48, 50, 51, 52, 53, 54, 56, 59, 60, 62	0.1359
2	2, 6, 7, 8, 10, 14, 18, 20, 23, 26, 27, 28, 31, 32, 33, 40, 42, 49, 55, 57, 58, 61	0.1991

higher than 0.12. The above clustering is very close to the bi-partition of betweenness-based algorithm by Girvan and Newman (2002), except that dolphin 40 and 31 are grouped into cluster 1 in their algorithm.

We summarize the clustering results for $p=0.03$ in the Table 4.4. We can compare

Table 4.4: Dolphin network clustering summary by clique-based clustering Algorithm 3.2 with $p = 0.20$

Cluster ID	Dolphin ID	Cluster Clique Score
1	2, 26, 27, 28	0.8333
2	8, 20, 29, 31	0.6667
3	6, 7, 8, 10, 14, 18, 33, 42, 55, 57, 58	0.5333
4	16, 19, 22, 24, 25, 30, 46, 52, 56, 60	0.5111
5	3, 43, 45, 48	0.5000
6	1, 15, 17, 21, 34, 35, 37, 38, 39, 41, 44, 51, 53	0.4230

Table 4.4 with Table 4.3. The clusters 1, 2, 3 in Table 4.4 combined are almost identical to cluster 2 in Table 4.3 except for dolphin with id 61. The cluster 4, 5, 6 in Table 4.4 are all sub-clusters from the cluster 1 in Table 4.3.

Except for dolphins in above clusters, there are some dolphins in small clusters of size 1 or 2. Their ids are 23, 32, 61, 40, 49, 9, 11, 36, 5, 12, 59, 47, 50, 54, 62, 4, 13. Usually, people are not interested in these small clusters, since they are easy to found even in a completely random network. These dolphins usually stays next to the boundary of larger clusters in above table.

4.2.3 Simulated Random Network

In this chapter, we will explore the effects of choice of parameter p on the clustering of randomly simulated network from stochastic block model. We will use the same method to simulate random network as we did in section 3.6. Here, we introduce the fourth stochastic block model. It has three built-in clusters of size 100, 20, and 20 respectively, and the within/between cluster link probability matrix B in Table 4.5. We denote this model as $SBM4$.

Table 4.5: Parameters of $SBM4$ for simulation of random network 4

cluster ID: size	$no.1 : 40$	$no.2 : 40$	$no.3 : 40$
$no.1 : 100$	0.15	0.01	0.01
$no.2 : 20$	0.01	0.8	0.02
$no.3 : 20$	0.01	0.02	0.8

First, we try the threshold parameter $p = 0.02$, and the clustering result is plotted in Figure 4.5. We can see clearly that the two smaller clusters are merged together in color green. Since the value for p is relatively small, so it is easy to merge small clusters.

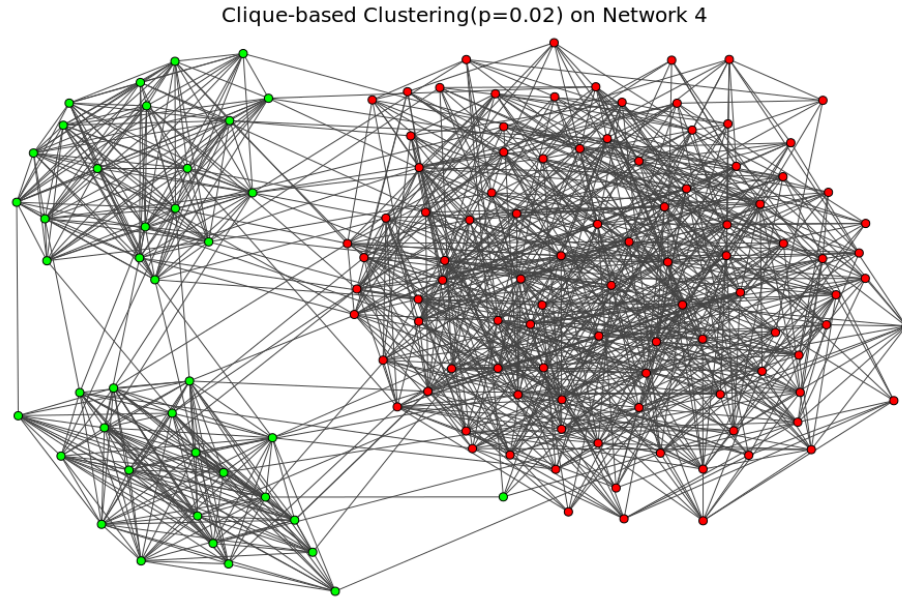


Figure 4.5: Clique-based clustering Algorithm 3.2 with $p = 0.02$ applied on a random network simulated from $SBM4$ with parameters in Table 4.5.

Now we will increase the value of p to be $p = 0.05$, and see what is going to change in the communities our algorithm detects. The result is plotted in Figure 4.6.

Now we can see that as we increase the parameter p from 0.02 to 0.05, our clique-based clustering algorithm does recover the three communities we built in the simulation model. We also notice the pattern that as we increase the value of parameter p , the communities our algorithm detects will become smaller. Now we further increase parameter p to be $p = 0.1$, and apply our clustering algorithm on the same random network again. The result is plotted in Figure 4.7.

To explore the optimal value for parameter p , we use the stochastic block model 4 with parameters defined in Table 4.5 to simulate 100 random networks. Then we perform network clustering on these 100 randomly simulated networks with parameter

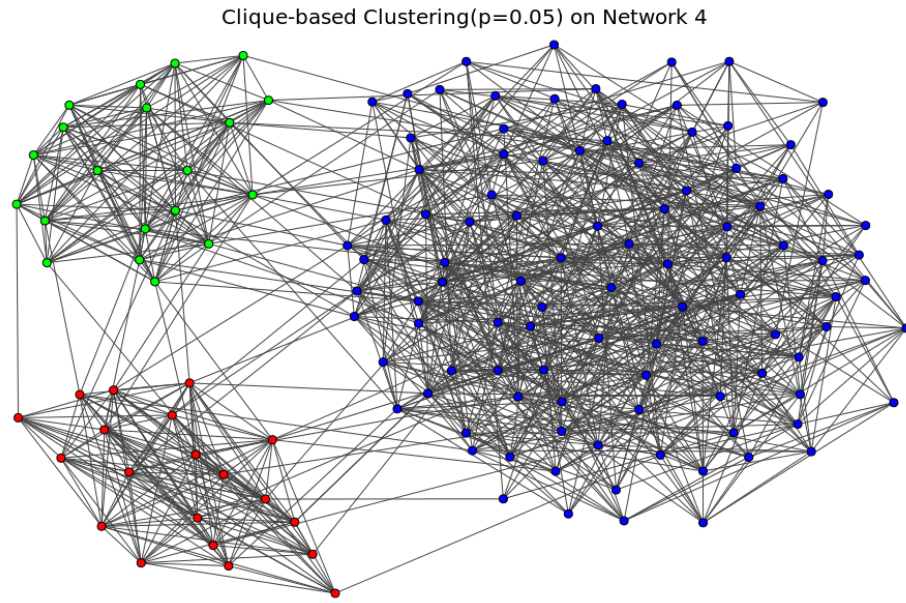


Figure 4.6: Clique-based clustering Algorithm 3.2 with $p = 0.05$ applied on a random network simulated from $SBM4$ with parameters in Table 4.5.

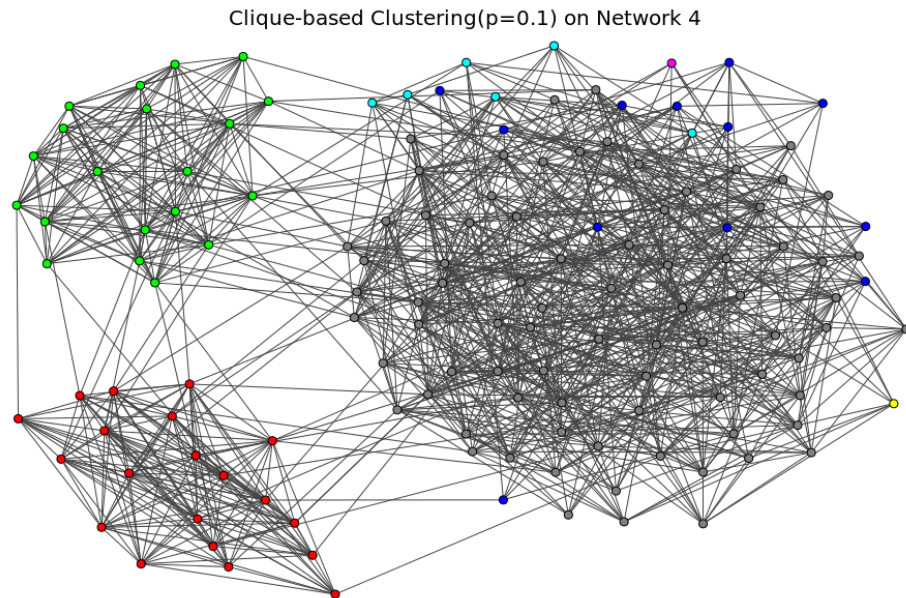


Figure 4.7: Clique-based clustering Algorithm 3.2 with $p = 0.1$ applied on a random network simulated from $SBM4$ with parameters in Table 4.5.

p equals to 0.02, 0.05, and 0.1 respectively, and summarize the results by Normalized Mutual Information (NMI) in Table 4.6.

Table 4.6: NMI estimate and standard error of Clique-based clustering out of 100 random network simulated from $SBM4$ with parameters in Table 4.5.

Clique-based Clustering Algorithm 3.2	\widehat{NMI}	$s.e.(\widehat{NMI})$
$p = 0.02$	0.5795	0.0394
$p = 0.05$	0.9186	0.0195
$p = 0.1$	0.8710	0.0105

In Figure 4.7, our clique-based clustering algorithm 3.2 with $p = 0.1$ detects the two smaller communities successfully, but it over-split the large community.

From table 4.6, we can see that $p = 0.05$ is indeed the optimal value for the threshold parameter p . In the next section, we will explain how to choose parameter p in a general case.

4.3 How to Choose Threshold Parameter p

From examples in section 4.2, we can see that adjusting parameter p is similar to adjusting the zoom of a camera, and it is a trade-off situation between over-splitting and under-splitting. Increasing p is like zooming in to examining the network more closely. The detected clusters are usually smaller with higher Clique Score, and therefore looks more like real clusters. However, there is higher risk of splitting a big completely random network of low Clique Score. In this case, we may zoom in too much at a local area of the network, and ignore the whole picture. This error of over-splitting is called type 1

error. On the other hand, decreasing p is like zooming out to exam the entire network, and therefore less likely to split a big completely random network. However we may risk merging weakly connected smaller clusters. This error of under-splitting is called type 2 error. In this section, we will discuss how to choose global parameter p to minimize both of these two types of error.

4.3.1 Control the Risk of Splitting an Erdős-Rényi Network

The Erdős-Rényi model is introduced by [Erdős and Rényi \(1959\)](#) and [Erdős and Rényi \(1960\)](#) to generate random networks. It sets a link between any pair of two nodes with equal probability, independently of the other links. We want to stress that a sub-network generated by Erdős-Rényi model is completely random without any cluster structure and an effective network clustering algorithm should minimize the error of splitting such sub-networks. From experiments, we found that there usually exists a split of Erdős-Rényi network into two similar size clusters that leads to higher Modularity Score, so a Modularity-based algorithm may choose to split an Erdős-Rényi network. This could also be an error of other network clustering algorithms. We want to control the risk of making this error in our algorithm.

Now suppose we have an Erdős-Rényi network of size n , and link probability p_0 . In order to avoid the type 1 error of splitting it, we definitely need to set the parameter $p < p_0$. However, $p < p_0$ is still not enough. Parameter p has to be smaller than p_0 significantly. In fact, a Erdős-Rényi network is completely random, in which there might

be some area with average link density higher than p_0 , and another area with average link density lower than p_0 . This random variation on local average link density may cause our algorithm to split the network if $p = p_0$. Intuitively, we need the parameter p to be small enough, such that there are very little chance to find two clusters, between which, the link probability is smaller than p .

Here, we want to find the upper bound of parameter p , such that the risk of type 1 error of over-splitting is less than a small positive value . For example, in an Erdős-Rényi network of size n and link probability p_0 , we can set $\alpha = 2.5\%$. We need to find the largest value for p , such that at most 2.5% of nodes are stripped from the majority of the network by our algorithm. In theory, each of the 2.5% stripped nodes has less than $0.975(n - 1)p$ links with the rest 97.5% of nodes in the network. Let X to denote the distribution of the observed link density between a stripped node and the rest 97.5% nodes. The upper bound of parameter p can be chosen as the 2.5 th percentile of distribution X , so that there will be only up to 2.5% of nodes to be stripped in theory. The 2.5% stripped node are more closely connected with each other than the rest 97.5% nodes. Therefore, the distribution X can be approximated by a Normal distribution after the top 2.5% truncated, with mean

$$E(X, \alpha) = p_0 - \frac{\phi(z_\alpha)}{1 - \alpha} \sqrt{\frac{p_0(1 - p_0)}{(1 - \alpha)(n - 1)}} \\ \stackrel{\alpha=0.025}{=} p_0 - 0.06 \sqrt{\frac{p_0(1 - p_0)}{0.975(n - 1)}}, \quad (4.1)$$

with variance

$$\begin{aligned} Var(X, \alpha) &= \frac{p_0(1-p_0)}{(1-\alpha)(n-1)} \\ &\stackrel{\alpha=0.025}{=} \frac{p_0(1-p_0)}{0.975(n-1)}, \end{aligned} \quad (4.2)$$

where $\phi(\cdot)$ is the *PDF* function of standard Normal distribution, and z_α is the $(1-\alpha)th$ percentile of standard Normal distribution, and $\alpha = 0.025$. In theory, given value α , we can set parameter p to be:

$$\begin{aligned} p(\alpha) &= \max(0, E(X) - z_\alpha \sqrt{Var(X)}) \\ &\stackrel{\alpha=0.025}{=} \max(0, p_0 - 2.02 \sqrt{\frac{p_0(1-p_0)}{0.975(n-1)}}), \end{aligned} \quad (4.3)$$

and there will be at most $\alpha = 2.5\%$ of nodes to be stripped from an Erdős-Rényi random network. In practice, there may be slightly more than 2.5% of nodes to be stripped, for the sake of simplicity, we used an approximation to the real distribution of. To test the effectiveness of formula (4.1), we simulate 100 random networks from Erdős-Rényi Random Graph Model, apply our clustering algorithm with p chosen from (4.3). Estimate of Type 1 error of over-splitting this Erdős-Rényi network can be obtained by averaging the percentage of stripped nodes over these 100 simulations. We have tried Erdős-Rényi model with 102 different network size n and link probability p_0 , and most have Type 1 error less than 4%.

The results are summarized in Table A.3 in Appendix. We will go over one example here. For an Erdős-Rényi random network of size $n = 100$ with link probability $p_0 = 0.2$, we plug in formula (4.3) and get $p = 0.118$. Using this parameter p in our clustering algorithm, there are on average 3.05% of nodes are stripped from rest of the majority network.

Recall example in section 3.6.3, we have a random network from stochastic block model with 4 clusters. We hope to set the parameter p to be significantly smaller than the Clique Score of all these 4 clusters to bound the risk of splitting them by 5% in practice. Using formula (4.3), cluster 1 of size 800 with Clique Score 0.1 requires $p \leq 0.078$; cluster 2 of size 400 with Clique Score 0.15 requires $p \leq 0.113$; cluster 3 of size 50 with Clique Score 0.4 requires $p \leq 0.257$; and cluster 4 of size 20 with Clique Score 0.6 requires $p \leq 0.37$. In all, 0.078 is the upper bound for parameter p to avoid splitting any of the four designed clusters in the random network. Therefore, 0.078 is the threshold, below which, we can call parameter p to be significantly smaller than the Clique Score of all 4 clusters. In section 3.6.3, we actually set $p = 0.06$, and it is working pretty good.

In all, the expression (4.3) provides a clear guideline about how to choose parameter p to keep the risk of splitting a completely random network to a very low level.

4.3.2 Control the Risk of Merging Significant Communities

In this section, assume we have two Erdős-Rényi random cluster of size n_1 and n_2 with Clique Score of p_1 and p_2 . The probability of link between nodes from these two cluster is p_{12} . Using normal distribution as an approximation to binomial distribution, we can see that there are still up to 2.5% of chance that the observed link density between these two clusters may be larger than $p_{12} + z_{.025}\sqrt{\frac{p_{12}(1-p_{12})}{n_1n_2}}$, where $z_{.025} = 1.96$ is the 97.5th percentile of standard normal distribution. Therefore, we require

$$p_{12} + 1.96\sqrt{\frac{p_{12}(1-p_{12})}{n_1n_2}} < \min(p_1, p_2)$$

otherwise, at least one of the clusters will not be statistically significant. In order for our algorithm to correctly split the two clusters, we need to set parameter p such that:

$$p > p_{12} + 1.96\sqrt{\frac{p_{12}(1-p_{12})}{n_1n_2}}. \quad (4.4)$$

For two relatively small clusters, $1.96\sqrt{\frac{p_{12}(1-p_{12})}{n_1n_2}}$ can be relatively large, so it is easier to merge small clusters. Recall the example random network 2 in section 3.6.2, cluster 2 and cluster 3 are both of size 10, and the probability of link between these two clusters are 0.05. Therefore, we need to set the parameter $p > 0.05 + 1.96\sqrt{\frac{0.05 \times 0.95}{100}} = 0.0927$. In this example, p is set to be $p = 0.11 > 0.0927$, and it is working well in practice. If we set $p = 0.09 < 0.0927$, however, chances of merging cluster 2 and cluster 3 are much

higher. This can be seen from the block wise error estimate in Table 4.7.

Table 4.7: Apply Algorithm 3.2 with $p = 0.09$ to 100 simulated networks from *SBM1* with parameters in Table 3.2

cluster ID: size	<i>no.1</i> : 100	<i>no.2</i> : 10	<i>no.3</i> : 10
<i>no.1</i> : 100	$\hat{P}_{11} = 0.58\%$, $s.e.(\hat{P}_{11}) = 0.07\%$	$\hat{P}_{12} = 3.64\%$, $s.e.(\hat{P}_{12}) = 0.61\%$	$\hat{P}_{13} = 2.04\%$, $s.e.(\hat{P}_{13}) = 0.49\%$
<i>no.2</i> : 10	$\hat{P}_{21} = 3.64\%$, $s.e.(\hat{P}_{21}) = 0.61\%$	$\hat{P}_{22} = 9.49\%$, $s.e.(\hat{P}_{22}) = 1.32\%$	$\hat{P}_{23} = 22.18\%$, $s.e.(\hat{P}_{23}) = 3.67\%$
<i>no.2</i> : 10	$\hat{P}_{31} = 2.45\%$, $s.e.(\hat{P}_{31}) = 4.26\%$	$\hat{P}_{32} = 22.18\%$, $s.e.(\hat{P}_{32}) = 3.76\%$	$\hat{P}_{33} = 7.20\%$, $s.e.(\hat{P}_{33}) = 1.31\%$

The 3rd row and 4th column of the table 4.7 shows that about 22% of chance for our algorithm to merge clusters 2 and 3 if $p = 0.90$. However, this risk of merging cluster 2 and 3 is only about 6% in section 6.2 when we set $p = 0.11$. We can see that even parameter p slightly lower than the bound in 4.4 will increase the error of merging cluster 2 and 3 significantly.

We also recall the example random network 2 in section 3.6.3, where cluster 3 and cluster 4 are of size 50, and 20 respectively, and the probability of link between these two clusters are 0.02. We need to set the parameter $p > 0.02 + 1.92\sqrt{\frac{0.02 \times 0.98}{50 \times 20}} = 0.02868$. In this example, p is set to be $p = 0.06 > 0.02868$, and it indeed works well in practice.

However, in real application, we usually don't know the value of p_{12} before an appropriate clustering is performed. The best we can do is to make the parameter p as large as possible, since that will decrease the chance of merging two clusters. In all, after the parameter p is set, now we are clear about type of pair of clusters, that our algorithm can avoid merging.

4.4 Localized Clustering Algorithm

From section 4.2, we can see that the choice of parameter p is a trade-off situation. To control the risk of splitting an Erdős-Rényi network of no cluster structure, p needs to be as small as possible, while to control the risk of merging two weakly connected clusters, p needs to be set as large as possible. In this section we modify our algorithm slightly to control both risks simultaneously.

Suppose we want to detect clusters from an observed network of size n , and Clique Score p_0 . As mentioned in section 4.3, sometimes we don't have any specific requirements on the Clique Score of found clusters. A good strategy is to choose parameter p such that both the type 1 error of over-splitting and type 2 error of under-splitting will be minimized. Since we want to control the error of splitting a possibly Erdős-Rényi network by 2.5%, we need to follow formula in 4.3 and set $p = \max(0, p_0 - 2.02\sqrt{\frac{p_0(1-p_0)}{0.975(n-1)}})$.

There is, however, a potential problem for above strategy in choosing parameter p . For example, after we performed our first split, the entire network will be divided into two sub-networks of size n_1 and n_2 with Clique Score of p_1 and p_2 respectively. Since the nature of our clustering is to maximize p -Clique index of a network, this recursive algorithm will work on splitting smaller and smaller sub-networks, with higher and higher Clique Score. Therefore, very likely, we may find out that p is significantly smaller than $p \ll \max(0, p_1 - 2.02\sqrt{\frac{p_1(1-p_1)}{0.975(n-1)}})$ or $p \ll \max(0, p_2 - 2.02\sqrt{\frac{p_2(1-p_2)}{0.975(n-1)}})$. So we may have relatively high risk of failing to split these sub-network even when they include two

or more weakly connected clusters. Furthermore, in theory, It is also possible to have $p > \max(0, p_1 - 2.02\sqrt{\frac{p_1(1-p_1)}{0.975(n-1)}})$ or $p > \max(0, p_2 - 2.02\sqrt{\frac{p_2(1-p_2)}{0.975(n-1)}})$. So we may still have risk in splitting these sub-networks even when they are completely random from Erdős-Rényi model. This problem, however, naturally leads to an important modification to our current clustering algorithm by allowing parameter p to vary from case to case.

4.4.1 Modify Global Clustering Algorithm

Our current clustering algorithm is a global clustering algorithm since it has a global parameter p that defines the minimum requirement in the Clique Score of all found clusters. In this section, we will use an illustrative example to illustrate how to transform our global clustering algorithm into a localized clustering algorithm. The key point is to allow p to vary as the recursive algorithm working on splitting sub-networks of different size and Clique Score.

Similar to section 3.6, we will simulate random networks from a stochastic block model to illustrate this method. This model includes three clusters of size 100, 20, and 20, and the within/between cluster probability matrix B in Table 4.8. We denote this model as $SBM5$.

Table 4.8: Parameters of $SBM5$ for simulation of random network 5.

cluster ID: size	<i>no.1</i> : 100	<i>no.2</i> : 20	<i>no.3</i> : 20
<i>no.1</i> : 100	0.2	0.05	0.05
<i>no.2</i> : 20	0.05	0.6	0.12
<i>no.3</i> : 20	0.05	0.12	0.8

The entire network has $n = 140$ nodes, and we expect the Clique Score in a simulated network is about $p_0 = \frac{100 \times 99 \times 0.2 + 20 \times 19 \times 0.6 \times 2 + 4 \times 2000 \times 0.05 + 2 \times 400 \times 0.12}{140 \times 139}$. Before performing any modeling or clustering to an observed network, we usually don't know if it is an Erdős-Rényi network. Still, we need to bound the error of splitting an Erdős-Rényi network by 2.5%. Using formula (4.3) in section 4.3.1, we need to set $p = \max(0, p_0 - 2.02 \sqrt{\frac{p_0(1-p_0)}{0.975(n-1)}}) = 0.0886$.

If $p = 0.0886$, the algorithm may not have any problem to split cluster 1 from cluster 2 and 3. However, very likely, it will fail to split cluster 2 and 3, since the link probability between these two clusters is as high as $0.12 > p = 0.0886$. This can be seen from the following experimental results.

First, we simulate random network 5 from stochastic model with parameters in Table 4.8. and apply our clustering algorithm with $p = 0.0886$. The clustering result is plotted in Figure 4.8. We also simulate another 100 random networks from the same stochastic block model, apply our algorithm with $p = 0.0886$ and get estimates of NMI in expression (1.5) and errors of the clustering result.

From the plot of clique-based clustering on one simulated random network in Figure 4.8, we can see the two smaller communities 1, and 2, are merged as one cluster colored in red. This is consistent with our expectation. The estimate of NMI in (1.5) is $\widehat{NMI} = 0.8488$ with $s.e.(\widehat{NMI}) = 0.0025$. The estimate of block wise error p in (3.25) is summarized in Table 4.9.

In the 3rd row, 4th column of Table 4.9, the entry that represents nodes between

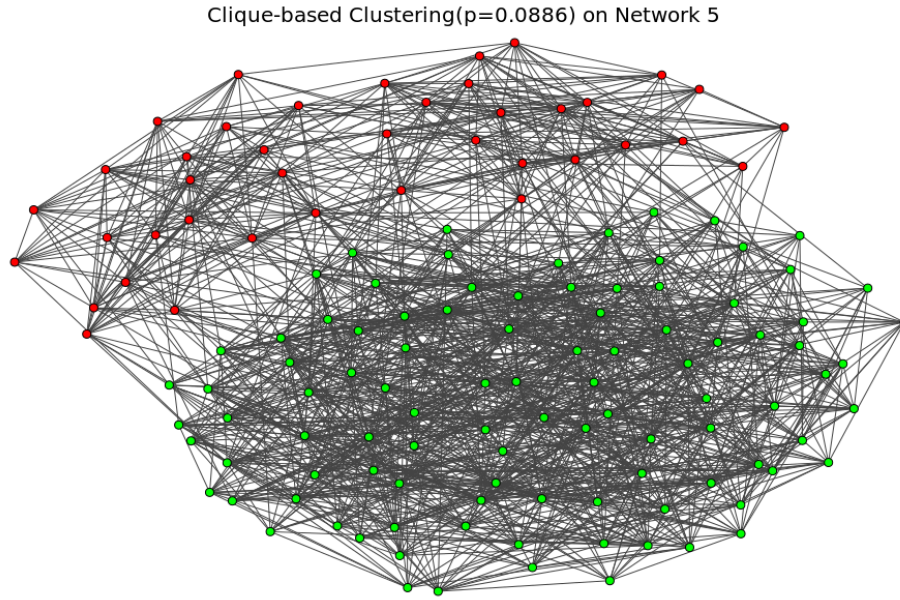


Figure 4.8: Clique-based clustering Algorithm 3.2 with $p = 0.0886$ applied on a random network simulated from $SBM5$ with parameters in Table 4.8.

cluster 2 and cluster 3, the estimate of error rate can be as high as 98.90% . In other words, the algorithm fails to split cluster 2 and cluster 3 almost for sure. All the other estimate of error rate in other entries of Table 4.9 are very small. So the algorithm indeed successfully split cluster 1 from cluster 2 and 3.

A very natural remedy to this problem is to increase the parameter p when it works on splitting the sub-network consists of cluster 2 and 3. The sub-network consists of clusters 2 and 3 having $n_1 = 40$ nodes, and the expected Clique Score is $p_1 = \frac{20 \times 19 \times 0.6 \times 2 + 2 \times 20 \times 20 \times 0.12}{40 \times 39} = 0.3538$. Using expression (4.3), we can set $p = \max(0, p_1 - 2.02\sqrt{\frac{p_1(1-p_1)}{0.975(n-1)}}) = 0.1972$. Using this new value for parameter p , we can not only avoid splitting this 40 nodes sub-network when it is an Erdős-Rényi random network, but also minimizing the chance of failing to split it when it does contain two or more clusters.

Table 4.9: Apply Algorithm 3.2 with $p = 0.0886$ to 100 simulated networks from *SBM5* with parameters in Table 4.9.

cluster ID: size	<i>no.1</i> : 100	<i>no.2</i> : 20	<i>no.3</i> : 20
<i>no.1</i> : 100	$\hat{P}_{11} = 0.60\%$, $s.e.(\hat{P}_{11}) = 0.11\%$	$\hat{P}_{12} = 0.12\%$, $s.e.(\hat{P}_{12}) = 0.30\%$	$\hat{P}_{13} = 0.12\%$, $s.e.(\hat{P}_{13}) = 0.03\%$
<i>no.2</i> : 20	$\hat{P}_{21} = 0.12\%$, $s.e.(\hat{P}_{21}) = 0.30\%$	$\hat{P}_{22} = 0.00\%$, $s.e.(\hat{P}_{22}) = 0.00\%$	$\hat{P}_{23} = 98.90\%$, $s.e.(\hat{P}_{23}) = 0.10\%$
<i>no.2</i> : 20	$\hat{P}_{31} = 0.12\%$, $s.e.(\hat{P}_{31}) = 0.03\%$	$\hat{P}_{32} = 98.90\%$, $s.e.(\hat{P}_{32}) = 0.10\%$	$\hat{P}_{33} = 0.10\%$, $s.e.(\hat{P}_{33}) = 0.09\%$

In this specific example, this 40 nodes sub-network contains two clusters: cluster 2 of size $n_{12} = 20$, and cluster 3 of size $n_{13} = 20$. The probability of link between nodes from these two clusters is $p_{123} = 0.12$. According to expression (4.4) in section 4.3.2, a lower bound for the value of parameter p to avoid merging cluster 2 and 3 is $p_{123} + 1.96\sqrt{\frac{p_{123}(1-p_{123})}{n_{12}n_{13}}} = 0.1518 > 0.1972 = p$. Therefore, our choice of $p = 0.1972$ is good enough.

Now, we can move on to another sub-network that consists of only cluster 1 of size $n_2 = 100$. Of course, in real application, we have no idea if this sub-network has only 1 or more clusters. Still, we need a balanced strategy that not only avoid splitting single cluster but also minimizing the risk of merging multiple clusters. The expected Clique Score of this sub-network is $p_2 = 0.2$. By formula (4.3), we can set a new value for parameter $p = \max(0, p_2 - 2.02\sqrt{\frac{p_2(1-p_2)}{0.975(n-1)}}) = 0.1178$ when working on the sub-network consists of real cluster 1.

For our hierarchical clustering framework, we still need to check if further bi-partition can be performed in detected sub-network of cluster 2, and sub-network of cluster 3.

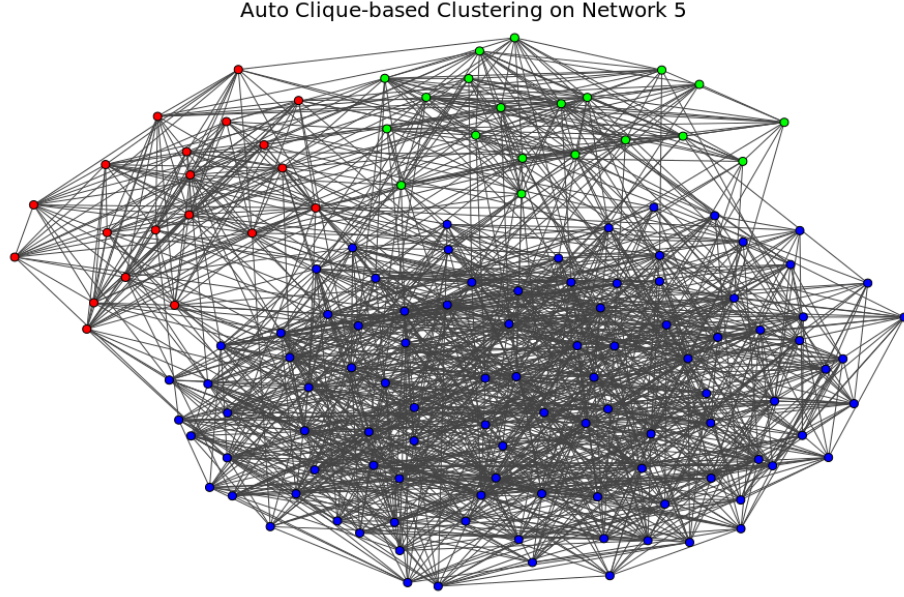


Figure 4.9: Local Clique-based clustering on random network simulated from $SBM5$ with parameters in Table 4.8.

For this purpose, we need to reset parameter p again and again using the strategy we described above.

We call this localized clustering algorithm described above auto clique-based clustering algorithm, since it may choose different threshold parameter p automatically for different sub-networks. We apply this algorithm to this random network simulated from stochastic model 5 with parameters defined in Table 4.8. and plot the clustering result in Figure 4.9. This algorithm is advantageous than the global clique-based algorithm with $p = 0.0886$, since it can split smaller community 1 and 2.

We also simulate 100 random networks from stochastic model 5 with parameters in Table 4.8 and apply our localized clique-based algorithm to them. The estimate of NMI in expression (1.5) is $\widehat{NMI}_{local} = 0.9677$ with $s.e.(\widehat{NMI}_{local}) = 0.0034$. This result is

much better than $\widehat{NMI} = 0.8488$ with $s.e.(\widehat{NMI}) = 0.0025$ from global p -clique Index Maximization algorithm 3.2 with $p = 0.0886$ we described before. The estimate of block wise error P in (3.25) is summarized in Table 4.10.

Table 4.10: Apply localized clique-based clustering algorithm to 100 simulated networks from $SBM5$ with parameters in Table 4.9.

cluster ID: size	<i>no.1</i> : 100	<i>no.2</i> : 20	<i>no.3</i> : 20
<i>no.1</i> : 100	$\hat{P}_{11} = 2.31\%$, $s.e.(\hat{P}_{11}) = 0.33\%$	$\hat{P}_{12} = 1.5 \times 10^{-3}\%$, $s.e.(\hat{P}_{12}) = 8.53 \times 10^{-4}\%$	$\hat{P}_{13} = 1.00 \times 10^{-2}\%$, $s.e.(\hat{P}_{13}) = 9.95 \times 10^{-3}\%$
<i>no.2</i> : 20	$\hat{P}_{21} = 1.50 \times 10^{-3}\%$, $s.e.(\hat{P}_{21}) = 8.53 \times 10^{-4}\%$	$\hat{P}_{22} = 3.34\%$, $s.e.(\hat{P}_{22}) = 0.57\%$	$\hat{P}_{23} = 0.19\%$, $s.e.(\hat{P}_{23}) = 0.19\%$
<i>no.2</i> : 20	$\hat{P}_{31} = 1.00 \times 10^{-2}\%$, $s.e.(\hat{P}_{31}) = 9.95 \times 10^{-3}\%$	$\hat{P}_{32} = 0.19\%$, $s.e.(\hat{P}_{32}) = 0.19\%$	$\hat{P}_{33} = 3.57\%$, $s.e.(\hat{P}_{33}) = 0.67\%$

We compared above Table 4.10 with Table 4.9, the chance of type 2 error of merging cluster 2 and 3 have been decreased from 98.9% to 0.19%. This simple example represents a usual bottleneck of performing clustering on real life large network. In real life large scale network, it is possible that the internal link density (Clique Score) of cluster 1 is not significantly higher than the link density between cluster 2 and 3 in another sub-network. There may be an inevitable conflict between avoid splitting cluster 1 and merging cluster 2 and 3 in our global algorithm in section 3.5. This conflict is easily removed in our localized algorithm by allowing parameter p to vary from one sub-network to another in the hierarchical clustering process. Our localized algorithm provides an example of how to minimize the errors of both over-splitting and under-splitting simultaneously.

4.4.2 Development of Localized Clustering Algorithm

In this section, we organize the idea and procedure in 4.4.1 into a local optimization framework. Still, we will perform hierarchical clustering in a network. The clustering procedure and results can be summarized by a binary tree. We start from the tree root, which represents the entire network. Based on the size, and observed link density (Clique Score) of the entire network, we can use formula (4.3) in section 4.3.1 to set a value for parameter p . Using this parameter p , we try to perform bi-partition on the entire network. If the network needs to be divided into two sub-networks, we create two child nodes for the root to represent these two sub-networks, and recursively repeat the same procedure in each of them. Finally, after the clustering is done, we end up with a binary tree, in which every leaf node represents a found community, and every non-leaf node represents an intermediate sub-network that we splitted in the hierarchical clustering procedure. Every tree node v represents a sub-network we tried bi-partition using parameter value p_v . The value of p_v depends on the sub-network size n_v , and observed Clique Score $p_v^{(0)}$. Using formula (4.3), we set $p_v = p_v^{(0)} - 2.02\sqrt{\frac{p_v^{(0)}(1-p_v^{(0)})}{0.975(n_v-1)}}$ to control the type 1 error of over splitting by $\alpha = 0.25\%$. We may also choose other values for threshold as long as it is small. The Figure 4.10 is an example of binary tree representation of hierarchical clustering.

Definition 4.1. *Let G be an arbitrary network of n nodes. A hierarchical clustering procedure on this network can be represented by a binary tree T , in which every node v*

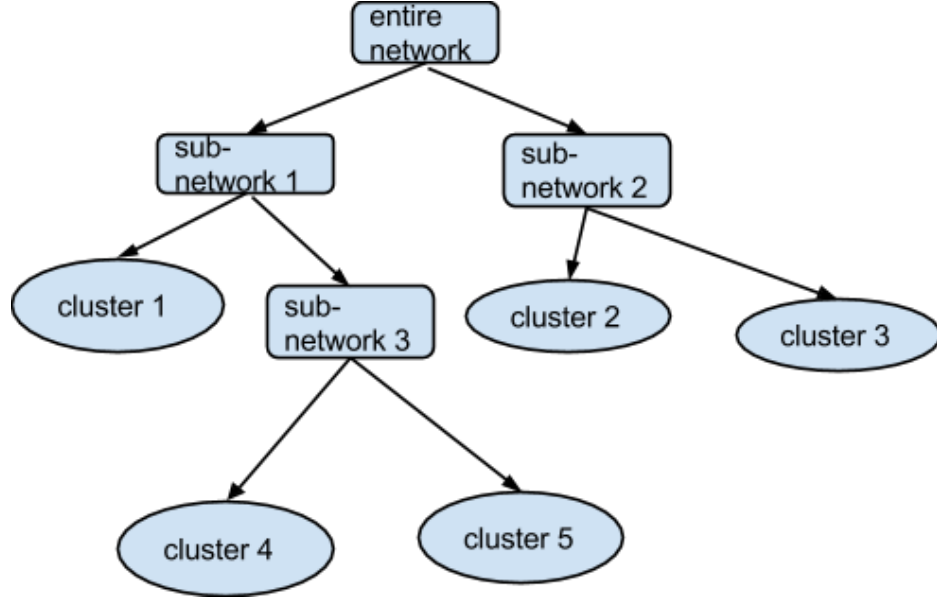


Figure 4.10: Localized Clique-based Hierarchical Clustering Scheme

represents a sub-network of G . Every internal node of T has two children, that represents the bi-partition of the sub-network it represents. Every leaf node of T represents a community found in G . The Local Clique Index is defined as follow:

$$LD(T) = \frac{1}{n(n-1)} \sum_{v \in T} \sum_{i, j \in v, i \neq j} ((a_{ij} - p_v) \delta_{ij}^{(v)} + (p_v - a_{ij})(1 - \delta_{ij}^{(v)})), \quad (4.5)$$

where $A = ((a_{ij}))$ is the adjacency matrix of the entire network; T is a binary tree, that represents the hierarchical clustering results; v is any node of T , that represents a sub-network our algorithm tried to divide; p_G is the value we use for parameter p when dividing sub-network represented by G ; $i, j \in G$ represents nodes i and j in the

sub-network G , and

$$\delta_{ij}^{(G)} = \begin{cases} 1 & , \text{ if } i \text{ and } j \text{ are kept together when dividing } G \\ 0 & , \text{ otherwise} \end{cases}$$

This objective function is very similar to expression (3.19) in Definition 3.3. In order to maximize $LD(T)$, we need to perform a bi-partition to every sub-network we encountered until the bi-partition causes negative contribution to the total of localized clique index. The bi-partition procedure are the same as described in section 3.5.1, except that now we compute the eigenvector of the local clique matrix

$$C^{(v)} = A^{(v)} - p_v(J_{n_v} - I_{n_v}), \quad (4.6)$$

where $A^{(v)}$ is the sub-matrix of A with index of nodes in sub-network v , parameter p_v may vary for different sub-networks, n_v is the size of sub-network v . Every bi-partition in sub-network v will bring contribution

$$\Delta LD = \frac{1}{n-1} \sum_{i \neq j, i, j \in v} ((a_{ij} - p_v)\delta_{ij}^{(v)} + (p_v - a_{ij})(1 - \delta_{ij}^{(v)})) \quad (4.7)$$

to total Local Clique Index LD . A bi-partition is needed only if this contribution $\Delta LD > 0$. This localized clustering algorithm can be summarized in Algorithm 4.1.

Algorithm 4.1 Community Detection via Local Clique Index Maximization

Input: Network G and binary tree T with single node G as root.

```

1: procedure BiPARTITION( $G$ )
2:   Compute size  $n_G$  and observed Clique Score  $p_G^{(0)}$  of  $G$ 
3:   Choose  $p_G := p_G^{(0)} - 2.02\sqrt{\frac{p_G^{(0)}(1-p_G^{(0)})}{0.975(n_G-1)}}$ 
4:   Compute Local Clique Matrix  $C^{(G)} = A^{(G)} - p_G(J_{n_G} - I_{n_G})$ 
5:   Compute the eigenvector  $s$  of  $C^{(G)}$  with largest eigenvalue
6:   Split  $G$  into  $G_1, G_2$  by the sign of  $s$ 
7:   Compute  $\Delta LD$  from above split of  $G$ 
8:   if  $\Delta LD > 0$  then
9:     Add  $G_1$  as left child of  $G$  in binary tree  $T$ 
10:    BiPARTITION( $G_1$ )
11:    Add  $G_2$  as left child of  $G$  in binary tree  $T$ 
12:    BiPARTITION( $G_2$ )
13:   end if
14: end procedure
Output: Binary Tree  $T$ .

```

We can start our algorithm by calling procedure $BiPartition(G)$, and the binary tree T shows the results of this hierarchical clustering. The found clusters are represented by the leaf nodes of binary tree T .

4.5 Examples: Simulated Networks

The localized Auto Clique-based network clustering algorithm is as efficient as the Modularity Maximization algorithm, but it brings in the statistical control of uncertainty when performing bi-partition. In this section, we will apply this novel algorithm to simulated random networks from all previous stochastic block model, plus two more models described below. Our clique-based algorithm is implemented in *Python*. The

performance will be estimated by *NMI* and running time.

First, we introduce one more stochastic block model, and denote it as *SBM6*. It has 3 built-in communities $\{1, 2, 3\}$, all of which is consist of 40 nodes. We include it to test our Local Clique Index maximization Algorithm 4.1 in network with equal size communities. The within/between community probability parameter matrix B is defined in Table 4.11. We use *SBM6* to simulate a sample random network 6, and apply our Local Clique Index Maximization Algorithm 4.1 to it. The clustering results is plotted in Figure 4.11. We can see that the three built-in clusters are correctly discovered and marked in color red, blue and green respectively.

Table 4.11: Parameters of *SBM6*

Cluster ID: size	<i>no.1</i> : 100	<i>no.2</i> : 20	<i>no.3</i> : 20
<i>no.1</i> : 100	0.2	0.01	0.01
<i>no.2</i> : 20	0.01	0.2	0.01
<i>no.3</i> : 20	0.01	0.01	0.2

Second, we will introduce one more stochastic block model that simulate large scale random networks. We denote it as *SBM7*. It has 25 built-in communities $\{1, 2, \dots, 25\}$ including 20000 nodes. The probability of a link between nodes from any two different communities always equals 0.005. The list of cluster size with internal link density is specified in Table 4.12.

We apply our Local Clique Index Maximization Clustering Algorithm 4.1 on random networks simulated from the 7 different stochastic block models we described previously, and summarize the results in Table 4.13. A description of the 8 columns of Table 4.13

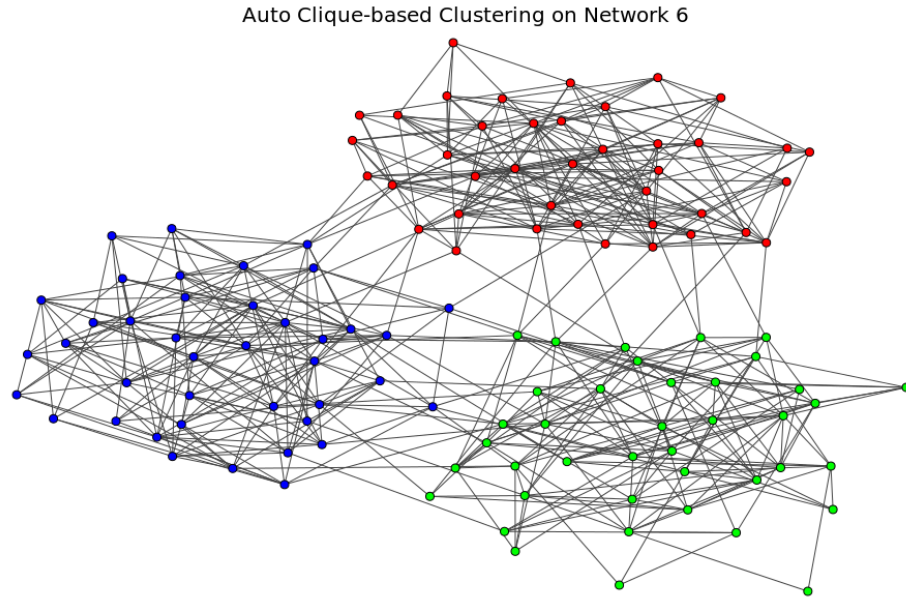


Figure 4.11: Local Clique-based clustering on random network simulated from *SBM6* with parameter in Table 4.11.

is here:

1. ID: Stochastic Block Model ID
2. B : The within/between community link density matrix B of Stochastic Block Model
3. n : Network size.
4. h : Number of built-in communities in the Stochastic Block Model
5. \widehat{NMI} : Average NMI of clustering on all simulated random networks
6. $s.e.(\widehat{NMI})$: Standard error of the estimate \widehat{NMI}

Table 4.12: Cluster size and internal link density for *SBM7*

Cluster ID	Cluster Size	Cluster Internal Link Density	Cluster ID	Cluster Size	Cluster Internal Link Density
1	3350	0.045	14	400	0.14
2	3000	0.05	15	400	0.14
3	2000	0.07	16	200	0.30
4	2000	0.07	17	200	0.30
5	2000	0.07	18	200	0.30
6	1000	0.09	19	100	0.40
7	1000	0.09	20	100	0.40
8	1000	0.09	21	50	0.80
9	1000	0.09	22	50	0.80
10	500	0.12	23	50	0.80
11	500	0.12	24	50	0.80
12	400	0.14	25	50	0.80
13	400	0.14			

7. Time: Average running time of *Python* implementation of the Local Clique Index

Maximization Clustering Algorithm 4.1.

8. M : The number of simulated networks for each Stochastic Block Model.

Table 4.13: Local Clique Index Maximization Clustering Algorithm 4.1

performance summary on 7 Stochastic Block Models

ID	B	n	h	\widehat{NMI}	$s.e.(\widehat{NMI})$	Time	M
<i>SBM1</i>	Table 3.2	120	3	0.8596	0.0138	92.7 ms	100
<i>SBM2</i>	Table 3.4	1270	4	0.9687	0.0088	94.5 ms	100
<i>SBM3</i>	Table 3.6	7000	10	0.9895	0.0022	1.66 s	20
<i>SBM4</i>	Table 4.5	140	3	0.9493	0.004	84.1 ms	100
<i>SBM5</i>	Table 4.8	140	3	0.9724	0.0032	97.1 ms	100
<i>SBM6</i>	Table 4.11	120	3	0.9008	0.0138	134 ms	100
<i>SBM7</i>	Table 4.12	20000	25	0.8960	0.0029	12.6 s	20

From above Table 4.13, we can see that even for a large network of size 20000, with

25 built-in communities, our *Python* version of the Local Clique Index Maximization Algorithm 4.1 can finish the clustering procedure in less than 13 seconds with satisfactory *NMI*.

So far, we have introduced two novel network clustering algorithms, the Clique-based algorithm with user defined global parameter p , and the localized Auto Clique-based algorithm, which may automatically choose different values for parameter p in different stages of the hierarchical clustering procedure. It is worth noting that it may be unfair to simply compare the *NMI* between our global and localized algorithm in benchmark networks. These two approaches serve for two different purposes. The global algorithm in chapter 3 is useful when we have a very specific requirement in the link density (Clique Score) of very found clusters. The drawback is the uncontrolled risks of splitting an Erdős-Rényi network with Clique Score lower than parameter p . Furthermore, we are unsure if the found clusters are real clusters or they just formed at random. The localized algorithm in chapter 4 is useful when we need to make sure that every found cluster is statistically significant, and not just formed at random.

Chapter 5

Future Work

In this section, we will summarize the results of our three community detection algorithms described in chapter 2, 3 and 4, and mention the advantages and disadvantages respectively. We will then introduce a music web application we developed as the future platform for our community detection algorithms and other machine learning algorithm. Finally we talk about what future work we can do in this web platform, and discuss our thoughts about the role of statistics in the big data age.

5.1 Conclusion

So far, we have introduced three community detection algorithms in chapter 2, 3 and 4. In this section, we will briefly summarize our results and make comparisons among them.

In chapter 2, we introduced our novel distance-based mixed membership model. Our method is similar to the mixed membership stochastic block model in [Airoldi et al. \(2008\)](#), since it also allows every node to belong to multiple communities with different strength. In our model, every node i in the network is assumed to have a latent community membership vector Z_i which represent the relative strength of affiliation of node i with every candidate community. The essence of our distance-based model is to use the cosine similarity matrix of all the n community membership vectors $[Z_1, \dots, Z_n]$ to predict the relationship between every pair of nodes. The community detection process is a clustering algorithm applied to a network, or graph. Therefore, there should be some implied distance function defined on pair of nodes in network for this clustering procedure. Our distance-based mixed membership model provides great insights into the relationship between the community structure and the underlying cosine distance function. The major disadvantage is that the *MCMC* procedure is very slow, in comparison with optimization based algorithm. Future works about this model is to make it faster. One possible solution is to use the Variational Bayes method as in [Airoldi et al. \(2008\)](#), which transform a simulation problem to an optimization problem. If the network is large, we can also consider to use stochastic gradient method as in [Gopalan and Blei \(2013\)](#) to make the optimization of variational objective function faster.

In chapter 3, we introduced a Clique-based community detection algorithm with a global user-controlled parameter p . This algorithm may guarantee that every community it detects has internal link density higher than p . This property is especially useful

in applications when we need find a friend circle for every individual in a social network. In chapter 4, we introduced an Auto Clique-based community detection algorithm. It claims that every community it detects is significantly different from the Erdős-Rényi network. The major purpose for designing this algorithm is to overcome the resolution limitation of Modularity Maximization algorithm explained in [Lancichinetti and Fortunato \(2011\)](#). We adopt an localized strategy in this hierarchical clustering procedure, such that, in every step, the results of the partition only depends on the sub-network we work on. This strategy enables our algorithm to minimize the risk of over-splitting and under-splitting simultaneously, which is theoretically impossible in a global optimization algorithm.

The most important tool for our two Clique-based approaches are the p -clique matrix $C = A - p(J_n - I_n)$ for global algorithm, and the local clique matrix in expression (4.6) for localized algorithm. Although these matrix are not sparse, they are be written as the linear combination of the sparse adjacency matrix A , matrix of all ones, and the identity matrix. This property will make the matrix vector product very efficient. The implicitly restarted Lanczos method in [Calvetti et al. \(1994\)](#) of finding leading eigenvector of matrix can be done very efficiently since it only requires computing matrix vector product iteratively, which can be done very fast in our case.

We have *Python* implementation in the last three optimization-based algorithms described in this thesis: Modularity Maximization (Algorithm 3.1), Global p -clique Index Maximization (Algorithm 3.2), and Local Clique Index Maximization (Algorithm 4.1).

The code is available at <https://github.com/ouyang86/clustering>. Future work about our Clique-based algorithm is relaxing its assumption that any random network without community structure can be simulated from Erdős-Rényi Random Graph Model. This assumption may not be true in real world, as we have seen in expression (2.7) that the degree distribution of Erdős-Rényi network is Poisson, not power law. The Barabasi-Albert model introduced in Barabasi and Albert (1999) can be used to simulate random network with power law degree distribution. This model is different from Erdős-Rényi model in that it assumes a network may grow, and the link formation has preferential attachment effects. In other words, it is easier for popular people to form new connections in a social network, but harder for isolated people to make new friends. If we treat Barabasi-Albert model as our null model without community structure, we will need to change our algorithm. However, it should work better in real world networks.

5.2 Music Web Application

Every online social network is a big web application. Here, we built a music web application **MusicGalaxy** at www.musicgalaxy.space using YouTube music data fetched from YouTube V3 API. YouTube itself is an online social network, in which every YouTube Channel represents a node, and video watching and subscription history represents edges. Part of our future work is to incorporate our community detection algorithm into this web application, so web users can use them for their own purpose. In this section, we

will describe this web application.

MusicGalaxy is designed as a clean and neat music platform based on the YouTube music data universe. At this time, in addition to the basic service of searching and streaming YouTube music video, it provides two functions. One is the YouTube version Billboard chart of weekly top 100 music list. Users can easily get link to the most popular music videos of the past week in YouTube. The other one is a music recommendation engine. When a user clicked to watch a YouTube video, a classification algorithm will determine if it is a music video. If yes, a named entity recognition algorithm will extract the artist names, and song names from the video titles. Then links of other music videos with either same artist names or same song names will be recommended. For example, if you clicked on a YouTube video with title ‘Sia - Chandelier’, the artist name ‘Sia’ and song name ‘Chandelier’ will be extracted, and other music videos performed by ‘Sia’, or other versions of the song ‘Chandelier’ will be recommended.

There are two machine learning algorithms installed in this web application: a video classifier and a named entity recognizer. The video classifier is implemented by ridge logistic regression and responsible for filtering out non-music videos. The named entity recognizer is implemented by second order hidden markov model, and responsible for extracting song name and artist name from YouTube music video titles. Both of these two algorithms are crucial for constructing an *SQL* music database from unstructured YouTube Video title text data.

5.3 Future Work Platform

In section 5.1, we have discussed some future work to improve all of our three community detection algorithms. Our ultimate objective is to incorporate these effective and efficient algorithms into our music web application “MusicGalaxy” described in section 5.2. In the back end database of this web application, we have the meta data for about 100000 YouTube channels. Each of these channels have collected videos from other channels. We can view this data set as a big social network of YouTube channels. A community detection algorithm is a powerful tool for recommending other similar channels to YouTube users. Since the YouTube usage data is live, the communities we detect may also change. We should also consider developing community detection algorithm that may respond to change of online data in real time.

Appendix A

Summary Tables

A.1 Tables of Chapter 2

In this section, we provide the summary of the simulated posterior distribution of the community membership parameter $Z = (Z_1, \dots, Z_n)$ in the distance-based mixed membership model for the Karate Club data and dolphin network data. The description of each columns of the summary table is here:

1. Parameter: $z[i, k]$ represents the percentage of membership of node i in community k .
2. Mean: The average of posterior sample for the corresponding parameter.
3. S.D.: The standard deviation of posterior sample for the corresponding parameter.

4. $MCMC_{s.e.}$: The standard error of the Bayes estimator for the corresponding parameter.
5. $2.5th$: The $2.5th$ percentile of posterior sample for the corresponding parameter.
6. Median: The Median of posterior sample for the corresponding parameter.
7. $97.5th$: The $97.5th$ percentile of posterior sample for the corresponding parameter.
8. Start: The starting number of MCMC iteration in Algorithm 2.1 to collect values for posterior sample.
9. Size: The size of collected posterior sample.

Table A.1: Summary of Posterior Distribution of community membership parameter $Z = (Z_1, \dots, Z_n)$ in the distance-based mixed membership model for the Karate Club data set.

Parameter	Mean	S.D.	$MCMC_{s.e.}$	$2.5th$	Median	$97.5th$	Start	Size
$z[1,1]$	0.03616	0.03224	0.000629	0.000938	0.02693	0.1205	5001	10000
$z[1,2]$	0.9638	0.03224	0.000629	0.8796	0.9731	0.9991	5001	10000
$z[2,1]$	0.01314	0.01317	0.000247	0.000314	0.00924	0.04903	5001	10000
$z[2,2]$	0.9869	0.01317	0.000247	0.951	0.9908	0.9997	5001	10000
$z[3,1]$	0.0106	0.01121	0.000351	0.000279	0.007421	0.03918	5001	10000
$z[3,2]$	0.9894	0.01121	0.000351	0.9609	0.9926	0.9997	5001	10000
$z[4,1]$	0.009242	0.009716	0.000335	0.000232	0.006394	0.03406	5001	10000
$z[4,2]$	0.9908	0.009716	0.000335	0.966	0.9936	0.9998	5001	10000
$z[5,1]$	0.2928	0.07405	0.003272	0.1581	0.2898	0.442	5001	10000
$z[5,2]$	0.7072	0.07405	0.003272	0.5581	0.7103	0.8419	5001	10000

Continuation of Table A.1								
Parameter	Mean	S.D.	$MCMC_{s.e.}$	2.5th	Median	97.5th	Start	Size
z[6,1]	0.3161	0.06105	0.002712	0.199	0.3164	0.4313	5001	10000
z[6,2]	0.6839	0.06105	0.002712	0.5688	0.6836	0.8011	5001	10000
z[7,1]	0.3159	0.0615	0.002763	0.1962	0.3169	0.4313	5001	10000
z[7,2]	0.6841	0.0615	0.002763	0.5688	0.6831	0.804	5001	10000
z[8,1]	0.01185	0.01357	0.000431	0.000166	0.006259	0.04506	5001	10000
z[8,2]	0.9882	0.01357	0.000431	0.955	0.9937	0.9998	5001	10000
z[9,1]	0.6889	0.1377	0.01124	0.429	0.7293	0.9043	5001	10000
z[9,2]	0.3111	0.1377	0.01124	0.09572	0.2707	0.5712	5001	10000
z[10,1]	0.5039	0.1501	0.006042	0.1871	0.4832	0.8076	5001	10000
z[10,2]	0.4961	0.1501	0.006042	0.1924	0.5168	0.8131	5001	10000
z[11,1]	0.2942	0.07389	0.003203	0.1598	0.2923	0.4409	5001	10000
z[11,2]	0.7058	0.07389	0.003203	0.5591	0.7078	0.8404	5001	10000
z[12,1]	0.3383	0.1425	0.005854	0.0981	0.3793	0.5981	5001	10000
z[12,2]	0.6617	0.1425	0.005854	0.4032	0.6207	0.9019	5001	10000
z[13,1]	0.1455	0.09796	0.003443	0.0411	0.1205	0.4386	5001	10000
z[13,2]	0.8545	0.09796	0.003443	0.5614	0.8795	0.959	5001	10000
z[14,1]	0.01814	0.01664	0.000504	0.000285	0.01545	0.05735	5001	10000
z[14,2]	0.9819	0.01664	0.000504	0.9427	0.9846	0.9997	5001	10000
z[15,1]	0.891	0.09337	0.004066	0.6638	0.9126	0.9993	5001	10000
z[15,2]	0.109	0.09337	0.004066	0.000727	0.08744	0.3366	5001	10000
z[16,1]	0.8863	0.09698	0.004008	0.6391	0.9075	0.9991	5001	10000
z[16,2]	0.1137	0.09698	0.004008	0.000868	0.09252	0.3612	5001	10000
z[17,1]	0.334	0.06607	0.002904	0.2039	0.3362	0.4546	5001	10000

Continuation of Table A.1								
Parameter	Mean	S.D.	$MCMC_{s.e.}$	2.5th	Median	97.5th	Start	Size
z[17,2]	0.666	0.06607	0.002904	0.5454	0.6639	0.7962	5001	10000
z[18,1]	0.1355	0.09057	0.003065	0.03943	0.1135	0.4253	5001	10000
z[18,2]	0.8645	0.09057	0.003065	0.5749	0.8865	0.9606	5001	10000
z[19,1]	0.8799	0.0955	0.004074	0.6517	0.9016	0.9983	5001	10000
z[19,2]	0.1201	0.0955	0.004074	0.001684	0.09847	0.3487	5001	10000
z[20,1]	0.2573	0.1502	0.006442	0.05883	0.2051	0.5377	5001	10000
z[20,2]	0.7427	0.1502	0.006442	0.4624	0.795	0.9412	5001	10000
z[21,1]	0.8839	0.1023	0.004469	0.6075	0.9101	0.9989	5001	10000
z[21,2]	0.1161	0.1023	0.004469	0.00112	0.08991	0.3925	5001	10000
z[22,1]	0.1417	0.09387	0.003908	0.04295	0.1174	0.4311	5001	10000
z[22,2]	0.8583	0.09387	0.003908	0.569	0.8826	0.9571	5001	10000
z[23,1]	0.8832	0.09578	0.004005	0.6367	0.9042	0.999	5001	10000
z[23,2]	0.1168	0.09578	0.004005	0.000976	0.09585	0.364	5001	10000
z[24,1]	0.9141	0.1098	0.006949	0.6464	0.9748	0.9997	5001	10000
z[24,2]	0.08586	0.1098	0.006949	0.00028	0.02525	0.3536	5001	10000
z[25,1]	0.619	0.06539	0.004337	0.5027	0.6138	0.7521	5001	10000
z[25,2]	0.381	0.06539	0.004337	0.2481	0.3862	0.4974	5001	10000
z[26,1]	0.6235	0.0677	0.004417	0.5027	0.6159	0.7608	5001	10000
z[26,2]	0.3765	0.0677	0.004417	0.2393	0.3841	0.4974	5001	10000
z[27,1]	0.9402	0.08809	0.006435	0.7102	0.9896	0.9999	5001	10000
z[27,2]	0.05981	0.08809	0.006435	0.00014	0.0104	0.29	5001	10000
z[28,1]	0.6701	0.1207	0.006145	0.4744	0.661	0.9705	5001	10000
z[28,2]	0.3299	0.1207	0.006145	0.02966	0.339	0.5258	5001	10000

Continuation of Table A.1								
Parameter	Mean	S.D.	$MCMC_{s.e.}$	2.5th	Median	97.5th	Start	Size
z[29,1]	0.5792	0.08532	0.003828	0.4299	0.575	0.7559	5001	10000
z[29,2]	0.4208	0.08532	0.003828	0.2442	0.425	0.5701	5001	10000
z[30,1]	0.9635	0.06229	0.005054	0.7759	0.9934	0.9998	5001	10000
z[30,2]	0.03652	0.06229	0.005054	0.000188	0.00664	0.2242	5001	10000
z[31,1]	0.7	0.1359	0.01112	0.4419	0.7368	0.9226	5001	10000
z[31,2]	0.3	0.1359	0.01112	0.07751	0.2632	0.5582	5001	10000
z[32,1]	0.6171	0.06621	0.004238	0.5011	0.6118	0.7561	5001	10000
z[32,2]	0.3829	0.06621	0.004238	0.2439	0.3882	0.4989	5001	10000
z[33,1]	0.9348	0.04282	0.002239	0.8382	0.9391	0.9965	5001	10000
z[33,2]	0.06517	0.04282	0.002239	0.003468	0.06088	0.1618	5001	10000
z[34,1]	0.9372	0.04205	0.000688	0.8391	0.9439	0.9957	5001	10000
z[34,2]	0.06284	0.04205	0.000688	0.004264	0.0561	0.1609	5001	10000

Table A.2: Summary of Posterior Distribution of community membership parameter $Z = (Z_1, \dots, Z_n)$ in the distance-based mixed membership model for the Dolphin Network data set.

Parameter	Mean	S.D.	$MCMC_{s.e.}$	2.5th	Median	97.5th	Start	Size
z[1,1]	0.003856	0.01087	0.000348	0.0000683	0.002117	0.01484	1001	5000
z[1,2]	0.9961	0.01087	0.000348	0.9852	0.9979	0.9999	1001	5000
z[2,1]	0.1065	0.06463	0.006143	0.03656	0.08161	0.2624	1001	5000
z[2,2]	0.8935	0.06463	0.006143	0.7377	0.9184	0.9635	1001	5000
z[3,1]	0.275	0.2529	0.02846	0.000125	0.3939	0.618	1001	5000
z[3,2]	0.725	0.2529	0.02846	0.3821	0.6061	0.9999	1001	5000

Continuation of Table A.2								
Parameter	Mean	S.D.	$MCMC_{s.e.}$	2.5th	Median	97.5th	Start	Size
z[4,1]	0.7162	0.04966	0.004548	0.612	0.7219	0.7981	1001	5000
z[4,2]	0.2838	0.04966	0.004548	0.202	0.2781	0.3881	1001	5000
z[5,1]	0.5339	0.1134	0.006199	0.3374	0.5283	0.8026	1001	5000
z[5,2]	0.4661	0.1134	0.006199	0.1977	0.4717	0.6628	1001	5000
z[6,1]	0.175	0.07466	0.007636	0.03351	0.1776	0.3189	1001	5000
z[6,2]	0.825	0.07466	0.007636	0.6813	0.8224	0.9665	1001	5000
z[7,1]	0.1566	0.04893	0.00485	0.055	0.158	0.2532	1001	5000
z[7,2]	0.8434	0.04893	0.00485	0.747	0.842	0.9451	1001	5000
z[8,1]	0.08182	0.07648	0.008624	0.02455	0.0522	0.3139	1001	5000
z[8,2]	0.9182	0.07648	0.008624	0.6862	0.9478	0.9755	1001	5000
z[9,1]	0.7192	0.04674	0.004289	0.6238	0.7242	0.7978	1001	5000
z[9,2]	0.2808	0.04674	0.004289	0.2022	0.2759	0.3762	1001	5000
z[10,1]	0.1562	0.04808	0.00512	0.06141	0.1545	0.2545	1001	5000
z[10,2]	0.8438	0.04808	0.00512	0.7457	0.8455	0.9389	1001	5000
z[11,1]	0.001996	0.002333	0.000114	0.0000536	0.001287	0.008082	1001	5000
z[11,2]	0.998	0.002333	0.000114	0.9919	0.9987	0.9999	1001	5000
z[12,1]	0.5522	0.1174	0.007133	0.3555	0.5477	0.8017	1001	5000
z[12,2]	0.4478	0.1174	0.007133	0.1988	0.4523	0.6449	1001	5000
z[13,1]	0.5211	0.1064	0.006296	0.3461	0.5122	0.7707	1001	5000
z[13,2]	0.4789	0.1064	0.006296	0.2301	0.4878	0.6544	1001	5000
z[14,1]	0.1487	0.04501	0.004674	0.06478	0.1454	0.2423	1001	5000
z[14,2]	0.8513	0.04501	0.004674	0.7578	0.8546	0.9354	1001	5000
z[15,1]	0.9927	0.005924	0.000153	0.9779	0.994	0.9997	1001	5000

Continuation of Table A.2								
Parameter	Mean	S.D.	$MCMC_{s.e.}$	2.5th	Median	97.5th	Start	Size
z[15,2]	0.007346	0.005924	0.000153	0.000287	0.006003	0.02216	1001	5000
z[16,1]	0.8134	0.04788	0.004368	0.6973	0.8218	0.9043	1001	5000
z[16,2]	0.1866	0.04788	0.004368	0.09566	0.1782	0.3027	1001	5000
z[17,1]	0.9922	0.00975	0.000495	0.9634	0.9953	0.9999	1001	5000
z[17,2]	0.007753	0.00975	0.000495	0.0000709	0.004685	0.0367	1001	5000
z[18,1]	0.1932	0.0542	0.003711	0.09405	0.1875	0.3059	1001	5000
z[18,2]	0.8068	0.0542	0.003711	0.6942	0.8125	0.906	1001	5000
z[19,1]	0.8684	0.03174	0.00292	0.786	0.873	0.9165	1001	5000
z[19,2]	0.1316	0.03174	0.00292	0.08354	0.127	0.2141	1001	5000
z[20,1]	0.07908	0.07084	0.008133	0.02664	0.05308	0.3023	1001	5000
z[20,2]	0.9209	0.07084	0.008133	0.6978	0.9469	0.9734	1001	5000
z[21,1]	0.7612	0.0991	0.01039	0.6239	0.7316	0.9541	1001	5000
z[21,2]	0.2388	0.0991	0.01039	0.04588	0.2684	0.3762	1001	5000
z[22,1]	0.893	0.03522	0.003336	0.8042	0.9006	0.9437	1001	5000
z[22,2]	0.107	0.03522	0.003336	0.05631	0.09937	0.1961	1001	5000
z[23,1]	0.3353	0.1037	0.006714	0.1345	0.341	0.5553	1001	5000
z[23,2]	0.6647	0.1037	0.006714	0.4449	0.6591	0.8655	1001	5000
z[24,1]	0.7341	0.0836	0.006818	0.5708	0.7308	0.8874	1001	5000
z[24,2]	0.2659	0.0836	0.006818	0.1127	0.2692	0.4294	1001	5000
z[25,1]	0.8636	0.03548	0.003299	0.7712	0.8684	0.9209	1001	5000
z[25,2]	0.1364	0.03548	0.003299	0.07932	0.1316	0.2291	1001	5000
z[26,1]	0.2477	0.0783	0.008603	0.07727	0.2649	0.373	1001	5000
z[26,2]	0.7523	0.0783	0.008603	0.6273	0.7351	0.9228	1001	5000

Continuation of Table A.2								
Parameter	Mean	S.D.	$MCMC_{s.e.}$	2.5th	Median	97.5th	Start	Size
z[27,1]	0.246	0.08321	0.009308	0.07255	0.2666	0.3735	1001	5000
z[27,2]	0.754	0.08321	0.009308	0.6271	0.7334	0.9275	1001	5000
z[28,1]	0.2371	0.08197	0.008907	0.06877	0.2573	0.3665	1001	5000
z[28,2]	0.7629	0.08197	0.008907	0.6336	0.7427	0.9313	1001	5000
z[29,1]	0.05031	0.0885	0.009597	0.008371	0.02287	0.3437	1001	5000
z[29,2]	0.9497	0.0885	0.009597	0.6567	0.9771	0.9917	1001	5000
z[30,1]	0.8842	0.03488	0.003451	0.792	0.8909	0.936	1001	5000
z[30,2]	0.1158	0.03488	0.003451	0.06411	0.1092	0.2081	1001	5000
z[31,1]	0.03989	0.06981	0.008078	0.004491	0.02127	0.3137	1001	5000
z[31,2]	0.9601	0.06981	0.008078	0.6868	0.9787	0.9956	1001	5000
z[32,1]	0.3398	0.0993	0.005496	0.1445	0.3379	0.5463	1001	5000
z[32,2]	0.6602	0.0993	0.005496	0.454	0.6622	0.8556	1001	5000
z[33,1]	0.3291	0.09228	0.008137	0.05441	0.3432	0.4724	1001	5000
z[33,2]	0.6709	0.09228	0.008137	0.5278	0.6568	0.9456	1001	5000
z[34,1]	0.9964	0.003692	0.000115	0.9866	0.9975	0.9999	1001	5000
z[34,2]	0.00361	0.003692	0.000115	0.0000914	0.002467	0.01339	1001	5000
z[35,1]	0.8373	0.1823	0.01991	0.5217	0.9651	0.9998	1001	5000
z[35,2]	0.1627	0.1823	0.01991	0.000185	0.03495	0.4785	1001	5000
z[36,1]	0.5393	0.1159	0.00636	0.351	0.5306	0.8078	1001	5000
z[36,2]	0.4607	0.1159	0.00636	0.1924	0.4695	0.6492	1001	5000
z[37,1]	0.7124	0.07273	0.00705	0.5837	0.7045	0.9023	1001	5000
z[37,2]	0.2876	0.07273	0.00705	0.0978	0.2956	0.4167	1001	5000
z[38,1]	0.9929	0.007449	0.000346	0.9734	0.9952	0.9999	1001	5000

Continuation of Table A.2								
Parameter	Mean	S.D.	$MCMC_{s.e.}$	2.5th	Median	97.5th	Start	Size
z[38,2]	0.007128	0.007449	0.000346	0.0000943	0.004858	0.02694	1001	5000
z[39,1]	0.9916	0.01033	0.000628	0.9635	0.996	0.9999	1001	5000
z[39,2]	0.008397	0.01033	0.000628	0.0000731	0.004021	0.03651	1001	5000
z[40,1]	0.5064	0.1292	0.008143	0.2506	0.5085	0.7126	1001	5000
z[40,2]	0.4936	0.1292	0.008143	0.2875	0.4916	0.7495	1001	5000
z[41,1]	0.9394	0.07351	0.007761	0.7001	0.9635	0.9941	1001	5000
z[41,2]	0.06058	0.07351	0.007761	0.005961	0.03656	0.3	1001	5000
z[42,1]	0.1214	0.0417	0.003801	0.06942	0.1095	0.2287	1001	5000
z[42,2]	0.8786	0.0417	0.003801	0.7714	0.8905	0.9306	1001	5000
z[43,1]	0.002375	0.002379	0.0000813	0.000047	0.001661	0.008862	1001	5000
z[43,2]	0.9976	0.002379	0.0000813	0.9911	0.9983	1	1001	5000
z[44,1]	0.9938	0.007717	0.000416	0.9733	0.9962	0.9999	1001	5000
z[44,2]	0.006169	0.007717	0.000416	0.000058	0.0038	0.02676	1001	5000
z[45,1]	0.6969	0.1418	0.01506	0.4764	0.6569	0.9657	1001	5000
z[45,2]	0.3031	0.1418	0.01506	0.03429	0.3431	0.524	1001	5000
z[46,1]	0.8709	0.03339	0.003063	0.788	0.8759	0.9213	1001	5000
z[46,2]	0.1291	0.03339	0.003063	0.07872	0.1242	0.2121	1001	5000
z[47,1]	0.5396	0.08	0.007371	0.4027	0.5319	0.7365	1001	5000
z[47,2]	0.4604	0.08	0.007371	0.2636	0.4681	0.5974	1001	5000
z[48,1]	0.004176	0.004561	0.000295	0.0000675	0.002453	0.01645	1001	5000
z[48,2]	0.9958	0.004561	0.000295	0.9836	0.9975	0.9999	1001	5000
z[49,1]	0.3672	0.09843	0.005645	0.1329	0.3739	0.56	1001	5000
z[49,2]	0.6328	0.09843	0.005645	0.4402	0.6261	0.8675	1001	5000

Continuation of Table A.2								
Parameter	Mean	S.D.	$MCMC_{s.e.}$	2.5th	Median	97.5th	Start	Size
z[50,1]	0.5417	0.08024	0.007531	0.4025	0.5349	0.739	1001	5000
z[50,2]	0.4583	0.08024	0.007531	0.2612	0.4651	0.5978	1001	5000
z[51,1]	0.9355	0.04384	0.002608	0.8153	0.9433	0.9943	1001	5000
z[51,2]	0.0645	0.04384	0.002608	0.005711	0.05667	0.1847	1001	5000
z[52,1]	0.8706	0.03218	0.002893	0.7901	0.8761	0.9191	1001	5000
z[52,2]	0.1294	0.03218	0.002893	0.08096	0.1239	0.2101	1001	5000
z[53,1]	0.947	0.05431	0.00596	0.7477	0.9612	0.986	1001	5000
z[53,2]	0.05299	0.05431	0.00596	0.0143	0.03882	0.2533	1001	5000
z[54,1]	0.5115	0.06868	0.006131	0.3844	0.5098	0.6369	1001	5000
z[54,2]	0.4885	0.06868	0.006131	0.3631	0.4903	0.6156	1001	5000
z[55,1]	0.09631	0.03993	0.003491	0.04651	0.08546	0.1984	1001	5000
z[55,2]	0.9037	0.03993	0.003491	0.8018	0.9146	0.9535	1001	5000
z[56,1]	0.7715	0.087	0.006162	0.5204	0.8002	0.8886	1001	5000
z[56,2]	0.2285	0.087	0.006162	0.1115	0.1998	0.4807	1001	5000
z[57,1]	0.2103	0.09711	0.009663	0.02744	0.2165	0.384	1001	5000
z[57,2]	0.7897	0.09711	0.009663	0.6173	0.7835	0.9726	1001	5000
z[58,1]	0.151	0.04398	0.004542	0.06736	0.1478	0.2402	1001	5000
z[58,2]	0.849	0.04398	0.004542	0.76	0.8522	0.9328	1001	5000
z[59,1]	0.5202	0.1042	0.005715	0.3496	0.5165	0.75	1001	5000
z[59,2]	0.4798	0.1042	0.005715	0.2501	0.4835	0.6504	1001	5000
z[60,1]	0.7246	0.04667	0.004314	0.6238	0.7304	0.8046	1001	5000
z[60,2]	0.2754	0.04667	0.004314	0.1955	0.2696	0.3765	1001	5000
z[61,1]	0.3587	0.08509	0.007553	0.05253	0.3629	0.5094	1001	5000

Continuation of Table A.2								
Parameter	Mean	S.D.	$MCMC_{s.e.}$	2.5th	Median	97.5th	Start	Size
$z[61,2]$	0.6413	0.08509	0.007553	0.492	0.6371	0.9482	1001	5000
$z[62,1]$	0.5092	0.06918	0.006368	0.3779	0.5076	0.6312	1001	5000
$z[62,2]$	0.4908	0.06918	0.006368	0.3688	0.4925	0.6222	1001	5000

A.2 Tables of Chapter 4

In this section, we select 102 Erdős-Rényi Random Graph models of different sizes n and link densities p_0 , and use each of them to simulated 100 random networks. Our purpose is to test the Type 1 error rate of applying our p -clique Index Maximization Clustering Algorithm [3.2](#) to these random networks. We summarize the results in Table [A.3](#). The description of each column of the summary table is here:

1. Size n : The total number of size of the Erdős-Rényi Random Network.
2. $E(deg(v))$: The expected degree of a particular node
3. p_0 : The link density parameter of Erdős-Rényi Random Graph Model used.
4. p : The threshold parameter used in p -clique Index Maximization Clustering Algorithm [3.2](#).
5. Type 1 Error: The type 1 error of splitting an Erdős-Rényi Random Network.

Table A.3: Summary of Type 1 error rate of splitting Erdős-Rényi Random Networks using p -clique Index Maximization algorithm 3.2.

Size n	$E(deg(v))$	p_0	p	Type 1 Error
2	0.85	0.85	0.12	9.00%
2	0.9	0.9	0.286	5.50%
2	0.95	0.95	0.504	2.00%
3	1.5	0.75	0.124	6.00%
3	1.6	0.8	0.221	3.67%
3	1.8	0.9	0.466	1.67%
3	1.9	0.95	0.635	4.33%
4	1.8	0.6	0.021	8.00%
4	2.1	0.7	0.159	4.50%
4	2.4	0.8	0.328	3.50%
4	2.7	0.9	0.546	3.75%
4	2.85	0.95	0.693	6.00%
5	2.16	0.54	0.03	5.00%
5	2.4	0.6	0.099	3.40%
5	2.8	0.7	0.231	4.00%
5	3.2	0.8	0.391	4.20%
5	3.6	0.9	0.593	4.80%
5	3.8	0.95	0.727	3.40%
10	2.61	0.29	0	3.20%
10	2.7	0.3	0	2.40%
10	3.6	0.4	0.066	2.60%
10	4.5	0.5	0.159	4.00%

Continuation of Table A.3				
Size n	$E(deg(v))$	p_0	p	Type 1 Error
10	5.4	0.6	0.266	4.00%
10	6.3	0.7	0.388	4.20%
10	7.2	0.8	0.527	4.50%
10	8.1	0.9	0.695	7.70%
10	8.55	0.95	0.801	7.00%
12	2.64	0.24	0	3.50%
12	3.3	0.3	0.017	3.17%
12	4.4	0.4	0.098	2.25%
12	5.5	0.5	0.192	2.83%
12	6.6	0.6	0.298	3.25%
12	7.7	0.7	0.417	4.42%
12	8.8	0.8	0.553	4.25%
12	9.9	0.9	0.715	5.83%
12	10.45	0.95	0.816	6.25%
20	2.85	0.15	0	3.20%
20	3.8	0.2	0.012	2.50%
20	5.7	0.3	0.085	3.40%
20	7.6	0.4	0.17	2.60%
20	9.5	0.5	0.265	3.55%
20	11.4	0.6	0.37	3.60%
20	13.3	0.7	0.485	3.55%
20	15.2	0.8	0.612	4.50%
20	17.1	0.9	0.759	5.70%

Continuation of Table A.3				
Size n	$E(deg(v))$	p_0	p	Type 1 Error
25	2.88	0.12	0	2.68%
25	4.8	0.2	0.033	1.72%
25	7.2	0.3	0.109	3.20%
25	9.6	0.4	0.195	2.92%
25	12	0.5	0.291	3.24%
25	14.4	0.6	0.395	4.40%
25	16.8	0.7	0.509	4.40%
25	19.2	0.8	0.633	3.24%
25	21.6	0.9	0.775	6.16%
50	2.94	0.06	0	2.80%
50	4.9	0.1	0.012	1.88%
50	9.8	0.2	0.083	2.76%
50	14.7	0.3	0.166	3.38%
50	19.6	0.4	0.257	3.64%
50	24.5	0.5	0.354	3.64%
50	29.4	0.6	0.457	3.82%
50	34.3	0.7	0.566	4.14%
50	39.2	0.8	0.683	3.84%
50	44.1	0.9	0.812	5.28%
100	2.97	0.03	0	2.89%
100	7.92	0.08	0.024	2.52%
100	9.9	0.1	0.038	2.88%
100	19.8	0.2	0.118	3.05%

Continuation of Table A.3				
Size n	$E(deg(v))$	p_0	p	Type 1 Error
100	29.7	0.3	0.206	3.44%
100	39.6	0.4	0.299	3.60%
100	49.5	0.5	0.397	3.65%
100	59.4	0.6	0.499	4.20%
100	69.3	0.7	0.606	3.69%
100	79.2	0.8	0.718	3.81%
100	89.1	0.9	0.838	4.47%
200	2.985	0.015	0	2.94%
200	5.97	0.03	0.005	1.84%
200	9.95	0.05	0.018	2.72%
200	19.9	0.1	0.056	2.83%
200	29.85	0.15	0.098	3.49%
200	39.8	0.2	0.142	3.42%
200	59.7	0.3	0.234	3.82%
200	79.6	0.4	0.329	3.53%
400	3.192	0.008	0	2.30%
400	5.985	0.015	0.003	1.94%
400	11.97	0.03	0.013	2.74%
400	19.95	0.05	0.028	3.03%
400	39.9	0.1	0.069	3.32%
400	58.85	0.15	0.113	3.47%
400	79.8	0.2	0.159	3.45%
400	119.7	0.3	0.253	3.47%

Continuation of Table A.3				
Size n	$E(deg(v))$	p_0	p	Type 1 Error
400	159.6	0.4	0.35	3.64%
800	3.196	0.004	0	2.37%
800	7.99	0.01	0.003	3.64%
800	15.98	0.02	0.099	2.89%
800	23.97	0.03	0.018	3.08%
800	39.95	0.05	0.034	3.24%
800	55.93	0.07	0.052	3.33%
800	79.9	0.1	0.078	3.28%
800	119.85	0.15	0.124	3.49%
800	159.8	0.2	0.171	3.50%
800	240	0.3	0.267	3.74%

Bibliography

Airoldi, E. M., Blei, D. M., Fienberg, S. E., and Xing, E. P. (2008). Mixed membership stochastic blockmodels. *The Journal of Machine Learning Research*, 9:1981–2014.

Barabasi, A. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286:509–521.

Calvetti, D., Reichel, L., and Sorensen, D. C. (1994). An implicitly restarted lanczos methods for large symmetric eigenvalue problems. *Electronic Transactions on Numerical Analysis*, 2:1–21.

Casella, G. and George, E. I. (1992). Explaining the gibbs sampler. *The American Statistician*, 46:167–174.

Chung, F. R. K. (1997). *Spectral Graph Theory*. CBMS Regional Conference Series in Mathematics. AMS and CBMS.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38.

Erdős, P. and Rényi, A. (1959). On random graphs. 1. *Publications in Mathematics*, 6:290–297.

Erdős, P. and Rényi, A. (1960). On the evolution of random graphs. *Publications of Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61.

Fortunato, S. and Barthelemy, M. (2007). Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104:36–41.

Frey, A. L. N. and Jain, A. K. (2003). Robust data clustering. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:128 – 133.

Freeman, L. C. (1977). A set of measures of centrality based upon betweenness. *Sociometry*, 40:35–41.

Gelfand, A. E. and Smith, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85:398–409.

Geman, S. and Geman, D. (1983). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

- Girvan, M. and Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99:7821–7826.
- Gopalan, P. K. and Blei, D. M. (2013). Efficient discovery of overlapping communities in massive networks. *Proceedings of the National Academy of Sciences*, 110:14534–14539.
- Hagen, L. and Kahng, A. B. (1992). New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-aided Design*, 11:1074–1085.
- Handcock, M. S., Raftery, A. E., and Tantrum, J. M. (2007). Model-based clustering for social networks. *Journal of Statistical Society, Series A*, 170:301–354.
- Hoff, P. D., Raftery, A. E., and Handcock, M. S. (2002). Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97:1090–1098.
- Hogben, L. (2013). *Handbook of Linear Algebra*. Chapman and Hall/CRC.
- Holland, P. and Leinhardt, S. (1981). An exponential family of probability distributions for directed graphs. *Journal of the American Statistical Association*, 76:33–50.
- Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, New Series*, 220:671–680.
- Lancichinetti, A. and Fortunato, S. (2011). Limits of modularity maximization in community detection. *Physical Review E*, 84:66122–66129.
- Lusseau, D., Schneider, K., Boisseau, O., Haase, P., Slooten, E., and Dawson, S. (2003). The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54:396–405.
- MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif. University of California Press.
- Newman, M. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103:8577–8582.
- Ng, A., Jordan, M., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 14:849–856.
- Nowicki, K. and Snijders, T. A. B. (2001). Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association*, 96:1077–1087.

- Reichardt, J. and Bornholdt, S. (2006). Statistical mechanics of community detection. *Physical Review E*, 74:016110.
- Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905.
- Snijders, T. A. B. and Nowicki, K. (1997). Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification*, 14:75–100.
- Zachary, W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473.
- Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–51.