COMMUNITY STRUCTURE OF LARGE NETWORKS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Jaewon Yang
March 2014
I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Jurij Leskovec, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Hector Garcia-Molina, Co-Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Leonidas Guibas

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.
This thesis is dedicated to my wife, Bobae.
Acknowledgement

First and foremost, I would like to express my sincere gratitude to my adviser Jure Leskovec, who is a genius when it comes to discovering insights from large data. I was exceptionally fortunate to become his first student, which allowed me to work with him very closely. I cannot thank more for the enormous amount of time that he spent on improving my research and mentoring me. Not only is he a good researcher, but also he was a great supporter for me. Even when my research faced rough moments, he kept encouraging me with great passion. Without his support and guidance, my journey to community detection would not have been as successful as now.

I am deeply grateful to my first reader Hector Garcia-Molina for his mentorship. I admire his clear thinking and superb communication ability. Even when my research topic seemed foreign to him, he always stunned me by providing crisp and insightful feedback in the end.

It has been my honor to have Leonidas Guibas as the second reader for this dissertation, and I would like to thank him for his advice. I would like to thank Ashish Goel for kindly accepting the invitation to chair my oral examination.

I thank Rok Sosic for serving on my orals committee and for giving well-balanced advice. Based on his long experience both in academia and industry, Rok gave me wonderful career advice that is hard to find elsewhere.

I would also like to thank my co-authors Deepak Agarwal, Bee-Chung Chen, Jeff Jacobs, Paea LePendu, Julian McAuley, Nigam Shah, Dafna Shahaf, and Heidi Wang. I especially thank Julian McAuley for the collaboration during my last year at Stanford. He was a great colleague and also supportive friend. Discussion with him always ended with a smile and a promising research idea. I also would like to thank Bee-Chung Chen and
Deepak Agarwal for their mentorship during my internship at LinkedIn. My internship was such a pleasure that I decided to work with them after Stanford.

I was very grateful to be a member of the Stanford infolab, which I found to be both incredibly social and intellectual. I want to thank all Infolab members and alumni that I had a chance to interact with. I would especially like to thank Hyunjung Park, Petros Venetis, and Steven Whang for their great help during my job search. I thank Andy Kacsmar, Andrej Krevl, and Mariane Siroker for providing wonderful administrative support.

I thank our SNAP group members. It was a pleasure to attend a group meeting on every Monday, and to share the office with Seth Meyers and Myunghwan Kim. I would like to single out Myunghwan Kim for being one of the closest friend and colleague.

I would like to thank my Korean buddies at Stanford: Youngsik Kim, Kietae Kwon, Sunhae Hong, Sunghee Park, HyungChai Park, Sangheok Kim. My life at Stanford would have been much more boring if I had not met these buddies. I also thank my buddies in Korea: Soohyun Baek, Kwangmoon Kim, Seungil Ahn and others who always greeted me with a smile whenever I visited Korea.

I would like to thank my parents, Choonseung and Kyongsook, and my sister Jaehyung for always believing in me and supporting me. Last, but certainly not least, I thank my wife, Bobae, whose boundless love and unconditional support made me go through this incredible journey called Ph.D. I look forward to our next voyage after Stanford.
Abstract

One of the main organizing principles in real-world networks is that of network communities, which are sets of nodes that share common properties, functions, or roles. Communities in networks often overlap as nodes can belong to multiple communities at once. Identifying such overlapping network communities is crucial for an understanding of social, technological, and biological networks.

In this thesis, we develop a family of accurate and scalable community detection methods and apply them to large networks. We begin by challenging the conventional view that defines network communities as densely connected clusters of nodes. We show that the conventional view leads to an unrealistic structure of community overlaps. We present a new conceptual model of network communities, which reliably captures the overall structure of a network as well as accurately models community overlaps. Based on our model, we develop accurate algorithms for detecting overlapping communities that scale to networks an order of magnitude larger than what was possible before.

Our approach leads to novel insights that unify two fundamental organizing principles of networks: modular communities and the commonly observed core-periphery structure. In particular, our results show that dense network cores stem from the overlaps between many communities. As the final part of the thesis, we present several extensions of our models such that we can detect communities with a bipartite connectivity structure and we combine the node attributes and the network structure for community detection.


## Contents

**Acknowledgement**

**Abstract**

### 1 Introduction

1.1 Thesis Overview and Contribution ........................................... 4

1.1.1 Step 1: Observations (Chapters 3, 4) .................................. 5

1.1.2 Step 2: Models (Chapter 5) ............................................. 6

1.1.3 Step 3: Algorithms (Chapters 6, 7) ..................................... 7

1.1.4 Step 4: Implications (Chapter 8) ....................................... 7

1.1.5 Step 5: Extensions (Chapters 9, 10) .................................... 8

### 2 Background and Related Work

2.1 Basic Concepts and Definitions ............................................. 10

2.1.1 General Graph-theoretic Concepts .................................... 10

2.1.2 The Community Detection Problem .................................... 11

2.1.3 Community Scoring Functions ........................................... 12

2.2 Previous Work on Network Community Detection ........................ 13

2.2.1 Non-overlapping Community Detection Methods ..................... 13

2.2.2 Overlapping Community Detection Methods .......................... 16

2.3 Table of Symbols .............................................................. 20

### 3 Ground-truth Communities

3.1 Introduction ........................................................................... 21
# Table of Contents

3.2 Intuition behind ground-truth communities .................................................. 22
3.3 Networks with Ground-truth Communities ................................................. 23
3.4 Distributions of Ground-truth Communities ............................................. 26
3.5 Use Case: Evaluating Structural Definitions of Communities .................... 28
3.6 Related Work .............................................................................................. 29
3.7 Conclusion ................................................................................................. 31

4 Observations of Community Overlaps ......................................................... 32
4.1 Introduction ............................................................................................... 32
4.1.1 Outline of Chapter ............................................................................. 33
4.2 Empirical Observations on Community Overlaps .................................... 33
4.3 Consequences for Present Overlapping Community Detection Methods .... 36
4.3.1 Clique Percolation ........................................................................... 38
4.3.2 Link Clustering ................................................................................ 39
4.3.3 Stochastic Block Models .................................................................. 42
4.3.4 Other Models of Network Communities ............................................ 43
4.4 Related work ............................................................................................. 44
4.5 Conclusion ............................................................................................... 45

5 Community-Affiliation Graph Model (AGM) ............................................... 46
5.1 Introduction ............................................................................................... 46
5.1.1 Outline of Chapter ........................................................................... 48
5.2 Community-Affiliation Graph Model ....................................................... 48
5.2.1 Flexibility of AGM ........................................................................... 51
5.3 Experimental Evaluations ......................................................................... 52
5.3.1 Experimental setup .......................................................................... 52
5.3.2 Estimating $p_c$ via convex optimization .......................................... 53
5.3.3 Evaluation: Edge Probability as a Function of Shared Community Memberships ........................................... 54
5.3.4 Evaluation: Properties of the network .............................................. 55
5.4 Related work on Models for Overlapping Communities ......................... 57
5.5 Conclusion ............................................................................................... 58
6 Community Detection with AGM

6.1 Introduction

6.1.1 Outline of Chapter

6.2 Fitting AGM

6.2.1 Automatically Finding the Number of Communities

6.2.2 AGM Does Not Suffer From the “Resolution” Limit

6.3 Experiments: Networks with Ground-truth Communities

6.3.1 Experimental Setup

6.3.2 Methods for Comparison

6.3.3 Evaluation Metrics

6.3.4 Results

6.4 Experiments: Biological Networks

6.4.1 Dataset description

6.4.2 Evaluation metrics

6.4.3 Results

6.5 Experiments: Networks in Ahn et al.

6.5.1 Dataset Description

6.5.2 Evaluation Metrics

6.5.3 Results

6.6 Conclusion

7 Fast Algorithm for Fitting AGM

7.1 Introduction

7.1.1 Outline of Chapter

7.2 Cluster-Affiliation Model for Big networks

7.3 Fitting Algorithm for BigCLAM

7.4 Experiments

7.4.1 Experiments on Synthetic Networks

7.4.2 Experiments on Networks with Ground-truth Communities

7.4.3 Experiments on Networks in Ahn et al.

7.4.4 Experiments on Large Networks
# Table of Contents

7.5 Related Work .................................................. 96
7.6 Conclusion ..................................................... 98

8 Implications to Core-Periphery .................................................. 99
8.1 Introduction ..................................................... 99
8.1.1 Outline of Chapter ........................................ 100
8.2 Dataset Description ............................................. 101
8.3 Dense Community Overlaps Lead to Global Core-periphery .............. 102
8.4 Existence of Local Cores ........................................ 105
8.5 Comparison to other notions of core-periphery .............................. 107
8.6 Related Work ................................................... 109
8.7 Conclusion ...................................................... 110

9 2-mode Communities ..................................................... 111
9.1 Introduction ..................................................... 111
9.1.1 Outline of Chapter ........................................ 116
9.2 Models for Directed Community Affiliations .................................. 116
9.3 Community Detection by CoDA ..................................... 120
9.4 Experiments ....................................................... 124
9.4.1 Dataset Description ........................................... 124
9.4.2 Experimental Setup .......................................... 125
9.4.3 Discovering Social Circles .................................... 126
9.4.4 Discovering Recipient Lists in Email Networks ............................ 127
9.4.5 Experiments on Large Networks ................................ 128
9.4.6 Scalability .................................................... 130
9.5 Qualitative Analysis on 2-mode communities and Cohesive Communities . 130
9.5.1 Biological and Foodweb Communities ............................. 132
9.5.2 2-mode vs. Cohesive Communities .................................. 134
9.6 Conclusion ....................................................... 136

10 Community Detection with Node Attributes .................................... 137
10.1 Introduction ..................................................... 137
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.1 Outline of Chapter</td>
<td>141</td>
</tr>
<tr>
<td>10.2 Models for Communities in Networks with Node Attributes</td>
<td>142</td>
</tr>
<tr>
<td>10.3 Community Detection with CESNA</td>
<td>145</td>
</tr>
<tr>
<td>10.4 Experimental Evaluations</td>
<td>149</td>
</tr>
<tr>
<td>10.5 Qualitative Analysis on Communities and Node Attributes</td>
<td>157</td>
</tr>
<tr>
<td>10.6 Related Work</td>
<td>160</td>
</tr>
<tr>
<td>10.7 Conclusion</td>
<td>161</td>
</tr>
<tr>
<td>11 Conclusions</td>
<td>162</td>
</tr>
<tr>
<td>11.1 Summary of Contributions</td>
<td>162</td>
</tr>
<tr>
<td>11.1.1 Summary of Our Methods</td>
<td>162</td>
</tr>
<tr>
<td>11.2 Future Work</td>
<td>163</td>
</tr>
<tr>
<td>11.2.1 Medium-Term Research Agenda</td>
<td>163</td>
</tr>
<tr>
<td>11.2.2 Long-Term Research Agenda</td>
<td>165</td>
</tr>
<tr>
<td>11.2.3 Outlook</td>
<td>166</td>
</tr>
<tr>
<td>Bibliography</td>
<td>167</td>
</tr>
<tr>
<td>A Appendix</td>
<td>183</td>
</tr>
<tr>
<td>A.1 Table of Symbols</td>
<td>183</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table Number</th>
<th>Table Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>The structure of the thesis and the associated publications.</td>
<td>5</td>
</tr>
<tr>
<td>3.1</td>
<td>Networks with ground-truth communities</td>
<td>23</td>
</tr>
<tr>
<td>5.1</td>
<td>Relative difference in the KS-statistic of AGM and LFR for network properties.</td>
<td>55</td>
</tr>
<tr>
<td>6.1</td>
<td>Network statistics</td>
<td>78</td>
</tr>
<tr>
<td>7.1</td>
<td>Relative improvement of BigCLAM over Metis and Graclus in detecting communities in large scale networks</td>
<td>96</td>
</tr>
<tr>
<td>8.1</td>
<td>Network statistics</td>
<td>102</td>
</tr>
<tr>
<td>8.2</td>
<td>Comparison with the cores detected by Rombach et al. [127]</td>
<td>109</td>
</tr>
<tr>
<td>9.1</td>
<td>Dataset statistics</td>
<td>124</td>
</tr>
<tr>
<td>9.2</td>
<td>Performance on Facebook, Google+, and Twitter</td>
<td>125</td>
</tr>
<tr>
<td>9.3</td>
<td>Performance of recipient discovery on the Enron network</td>
<td>128</td>
</tr>
<tr>
<td>9.4</td>
<td>Relative accuracy (compared to the worst performing method) of detected communities on large scale social networks</td>
<td>129</td>
</tr>
<tr>
<td>9.5</td>
<td>Dataset statistics</td>
<td>131</td>
</tr>
<tr>
<td>10.1</td>
<td>Methods for community detection in networks with node attributes</td>
<td>139</td>
</tr>
<tr>
<td>10.2</td>
<td>Dataset statistics</td>
<td>150</td>
</tr>
<tr>
<td>10.3</td>
<td>Performance of methods on five datasets</td>
<td>152</td>
</tr>
<tr>
<td>11.1</td>
<td>Summary of the proposed community detection methods</td>
<td>163</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Ground-truth community size distribution</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>Node membership distribution</td>
<td>26</td>
</tr>
<tr>
<td>3.3</td>
<td>Community overlap distribution</td>
<td>27</td>
</tr>
<tr>
<td>3.4</td>
<td>Relative community overlap</td>
<td>28</td>
</tr>
<tr>
<td>4.1</td>
<td>Community overlaps are more densely connected than the non-overlapping parts of communities</td>
<td>35</td>
</tr>
<tr>
<td>4.2</td>
<td>Three different definitions of network communities</td>
<td>36</td>
</tr>
<tr>
<td>4.3</td>
<td>Clique Percolation Method cannot detect communities with dense overlaps</td>
<td>38</td>
</tr>
<tr>
<td>4.4</td>
<td>Link Clustering cannot detect communities with dense overlaps</td>
<td>40</td>
</tr>
<tr>
<td>4.5</td>
<td>Example of Link Clustering on densely overlapping communities</td>
<td>42</td>
</tr>
<tr>
<td>4.6</td>
<td>The result of a Stochastic block model and the Mixed Membership Stochastic Block Model on a network of two communities with dense overlap</td>
<td>42</td>
</tr>
<tr>
<td>5.1</td>
<td>Flexibility of AGM</td>
<td>51</td>
</tr>
<tr>
<td>5.2</td>
<td>Model comparison: Conditional edge probability between two nodes given the number of common groups that they belong to</td>
<td>54</td>
</tr>
<tr>
<td>5.3</td>
<td>Model comparison: Network properties. (Degree distribution, Clustering Coefficient distribution)</td>
<td>59</td>
</tr>
<tr>
<td>5.4</td>
<td>Model comparison: Network properties. (Hop distribution, Triad participation)</td>
<td>60</td>
</tr>
<tr>
<td>5.5</td>
<td>Model comparison: Network properties. (Eigenvectors, Eigenvalues)</td>
<td>61</td>
</tr>
<tr>
<td>6.1</td>
<td>Facebook communities detected by AGM</td>
<td>63</td>
</tr>
</tbody>
</table>
9.6 Examples of overlapping 2-Mode communities detected by our method in the Chesapeake bay food web network ......................................................... 132
9.7 Overlapping 2-Mode communities detected by our method in a Protein-Protein interaction network ................................................................. 133
9.8 Fraction of 2-Mode communities and cohesive communities ................................................................. 134
10.1 Two ways of modeling the statistical relationship between a graph $G$, attributes $X$, and communities $F$ ................................................................. 141
10.2 Plate representation of CESNA ................................................................................................. 144
10.3 Robustness to edge deletion ................................................................................................. 155
10.4 Algorithm runtime comparison ................................................................................................. 156
10.5 Communities of philosophers found by CESNA (left) and equivalent communities detected by BigCLAM (right) ......................................................................................... 158
10.6 The node attributes which CESNA learns to be associated with the communities ................................................................. 160
Chapter 1

Introduction

Networks provide a fundamental way to represent and describe many real-world complex systems such as social [162], technological [22], and biological systems [161]. In networks, the nodes correspond to individual entities (e.g., people in social networks, web-pages in web graphs, and proteins or genes in biological systems), and the edges indicate pair-wise relationship between the entities (e.g., friendship between people, hyperlink between webpages, and interaction between proteins).

Traditionally, networks have been analyzed at two very different scales: “microscale” and “macroscale”. The aim of microscale is to understand how individual nodes interact with each other [148]. As a running example, consider a collaboration network among computer science researchers (e.g., the DBLP co-authorship network [10]) where the nodes are researchers and edges mean co-authoring a paper. In this network, at microscale level of analysis one would investigate the following questions: Which node (researcher) is the most central in the network? [22] Which pair of researchers would be the most likely to collaborate together in the near future? [98] However, such analyses tell us little about the large-scale structure of the entire network as they focus on a few individual researchers.

On the other extreme is macroscale that focuses on global statistical properties of the entire network. In the example of the author collaboration network, at macroscale one may pose the following questions: What is the degree (i.e., the number of collaborators of a researcher) distribution? [43] How closely correlated are the degrees of two researchers
CHAPTER 1. INTRODUCTION

when they collaborate? These questions lead to finding the average characteristics of a given network; however, these analyses provide limited information for the structure of the network because they do not reveal how individual nodes are organized in the network.

Thus, arguably the most useful scale for analyzing the structure of networks is something in between macroscale and microscale, which we refer to “mesoscale”. In mesoscale analyses, we study the groups of nodes in networks. In the DBLP collaboration network, for example, we would aim to identify groups of researchers who share the same research interests (e.g., database researchers, machine learning researchers, or architecture researchers). These groups, if identified, would clearly show the structure and the organization of the underlying system that we are modeling as a network. In particular, these groups will tell us how the research in computer science is organized around different research interests and how individual researchers belong to different lines of research.

The best-studied form of the mesoscale structure in networks is that of network communities. Communities are groups of nodes that have common functions, roles, or properties. In the DBLP example, one can think of database researchers or machine learning researchers as communities. The key task in the mesoscale analyses is network community detection, which refers to identifying the communities in a given network from the network structure (i.e., nodes and edges). The premise behind community detection is that nodes sharing certain properties (nodes belonging to a same community) tend to form a more densely connected cluster, i.e., the network structure reflects communities. In the DBLP network, for example, the premise means that researchers sharing the same interest would collaborate more often. There are other examples where the premise naturally holds. Society (social network) organizes into various social communities such as families, friendship circles, villages and associations. In biological networks, such as protein-protein interaction networks, the proteins involved in a common function are more likely to interact with each other. In web graphs, pages about a same topic link to each other more often.

Communities can be defined in two ways, either structurally or functionally. A structural definition of communities means that a community is a set of nodes with a particular connectivity structure. For example, claimed that a community is a set of nodes with high Modularity score. On the other hand, a functional definition of communities specifies
a community to be a set of nodes with a common function, which can be common role, affiliation, or attribute [46, 58].

Unfortunately, historically the functional and the structural definitions of communities have often been confused with each other. However, in this thesis we make a clear distinction between the two definitions. Our ultimate goal in network community detection is to discover functional communities (e.g., in DBLP, groups of researchers with the same research interests). To identify such node groups, we employ the structural definition of communities (e.g., finding such groups of researchers based on their connectivity structure in the DBLP network).

Network community detection is a fundamental task for understanding the structure of networks [50]. In addition, the network communities that one detects prove to be very useful in many applications. For example, communities allow us to discover functionally related objects in information networks [52, 53, 156] and biological networks [81]. With communities, we can study interactions between different modules [5]. Finally, when we need to infer missing attribute of nodes [11, 34] or predict unobserved connections [29, 78], knowing community structure improves the accuracy of prediction.

Unfortunately, developing a reliable community detection method is a very challenging task for several reasons. First is modeling: Communities often overlap each other as some nodes belong to multiple communities. In the running example of the CS collaboration network, a data mining research community overlaps with a machine learning community because the researchers who work on “large scale machine learning” belong to the both communities. On the other hand, some communities may not overlap with each other (e.g., machine learning community and architecture community). In fact, communities may not overlap [53], or may overlap [118], or may be even hierarchically nested [4] — and it is not clear how to model such complex relationships among communities. The second challenge is computation: Many algorithmic formalizations of community detection lead to intractable problems [118] and thus the existing overlapping community methods cannot scale to networks with millions of nodes [132]. The third challenge is evaluation: Ground-truth communities are challenging to define. The lack of a reliable ground-truth makes the evaluation of extracted communities and comparison of algorithms extremely difficult. Developing community detection methods that can address these difficulties would be a
crucial contribution for broadening our understanding of networks.

1.1 Thesis Overview and Contribution

The goal of this thesis is to:

Develop a family of accurate and scalable methods to detect network communities, and study the community structure of large networks.

To achieve this goal, we take the following five steps throughout the thesis. At each step, we begin by asking a motivating question, and then we find an answer with a principled approach. By taking these steps, we will also address the aforementioned challenges in community detection (modeling, computation, and evaluation).

• Step 1 – Observations: How do real communities overlap? To answer this question, we begin with collecting “ground-truth” communities from the networks where the nodes explicitly state their community memberships. We then examine how ground-truth communities overlap with each other. We find that the existing methods commonly made the implicit assumption that community overlaps are sparse and show that the assumption is not supported by the data. Rather, community overlaps are denser than individual communities.

• Step 2 – Models: How can we model communities and networks? Based on the novel observation that community overlaps are denser, we design a simple conceptual model for how networks organize into overlapping communities. This model allows us to gain intuitions about the networks and communities and to simulate various scenarios for networks. This model solves the modeling challenge in community detection because it can naturally represent various overlapping relationships among communities.

• Step 3 – Algorithms: How can we detect communities accurately in large networks? We develop algorithms for community detection. Our algorithms detect communities in a given network by fitting our conceptual model to the network. We start by developing a local-search method that finds the best-fit community affiliations for a
given network. Next, we describe how we can speed up the fitting procedure by using an approximate algorithm. This step addresses the computation challenge as our proposed methods can scale to million node networks, and the evaluation challenge as we quantitatively evaluate the performance of our method using the gold-standard ground-truth that we defined in Step 1.

- Step 4 – Implications: *What are the implications of our findings?* We conduct an in-depth study on the community structure of large networks by using our community detection methods. We find that densely overlapping communities explain the emergence of the core-periphery structure of networks.

- Step 5 – Extensions: *How can we extend our methods?* We propose two extensions of our models. The first extension focuses on detecting a different kind of community, which we call a 2-mode community, and on handling directed networks as well as undirected networks. The second extension focuses on incorporating node attributes as well as network structure for community detection.

The five steps of the thesis, as well as the associated chapters and published papers, are shown in Table 1.1. In addition, Chapter 2 introduces backgrounds about community detection and discusses existing methods.

### 1.1.1 Step 1: Observations (Chapters 3, 4)

In this step, we begin by collecting a set of networks where the nodes explicitly state their memberships to communities, which we call “ground-truth” communities. Then, we study the connectivity structure of ground-truth communities and their overlap in order to gain insights for developing better community detection methods.
In the networks with ground-truth communities, we observe that the nodes residing in several communities are *more densely* connected than the nodes that reside in a single community. This observation is striking because it directly contradicts the common wisdom on community overlap that the nodes in the overlaps would be *more sparsely* connected to each other. In particular, current community detection methods assume that if nodes $u, v$ belongs to multiple communities together, these two nodes are less likely to be connected than if they would belong to only one community together, and our observation is opposed to this assumption. Moreover, our analysis proves that the existing methods that assume sparse community overlaps cannot detect densely overlapping communities accurately. Our findings suggest a need to develop novel models and algorithms that account for dense community overlaps.

1.1.2 Step 2: Models (Chapter 5)

In this step, we build on our observations in the previous step and ask the following question: What underlying process causes community overlaps to be denser than the communities themselves? This question motivates our contribution in this step: We present a family of probabilistic generative models for networks that capture the observed phenomena and that produce networks with realistic community structure.

We build on models of affiliation graphs [21, 88] and develop the *Community-Affiliation Graph Model* (AGM) which reliably reproduces the organization of networks into communities and the overlapping community structure. Our model specifies the probability of an edge between a pair of nodes as a function of the communities that the two nodes share, meaning that the edges of the network arise due to shared node community memberships. Our model allows us to model a complex overlapping relationships between communities in a very natural way, which addresses the aforementioned challenge in modeling the community structure.

We show theoretically that the our model generates networks with dense community overlaps. In addition to the theoretical analyses, our experiments on a range of networks demonstrate that the our model reliably captures nodes’ community memberships, internal structure of the communities, and realistic community overlaps. We also discuss the
CHAPTER 1. INTRODUCTION

connection between our model and the literatures in sociology.

1.1.3 Step 3: Algorithms (Chapters[6],[7])

In this step, our goal is to develop a community detection method that successfully detects overlapping, non-overlapping, as well as nested communities in networks. We achieve this goal by fitting our AGM (i.e., finding the best-fit node-community affiliation graph) to a given network. To fit the AGM, we perform the maximum likelihood estimation, meaning that we search the community affiliation graph that maximizes the probability of the AGM generating the network.

We present two algorithms for the maximum likelihood estimation. The first algorithm performs a local search on the space of all possible community affiliation graphs. We combine the Markov Chain Monte Carlo method and convex optimization in order to search for the optimal community affiliations. This method achieves higher accuracy in community detection for the networks with a few thousand nodes.

The second is a fast approximation algorithm for the first one, where we approximate community affiliation graphs by real-valued membership factors and then estimate the best values for the membership factors. We call the second method Cluster-Affiliation Model for Big networks (BigCLAM). Our experiments show that, while achieving a similar accuracy to the first method, this fast method is 100 times faster in average compared to the existing community detection methods. These results suggest that our method is both the most accurate and most scalable method compared to the previous state-of-the-art.

1.1.4 Step 4: Implications (Chapter[8])

Having developed novel community detection methods, in this step, we study the implications of the overlapping communities that we discover with our methods. In particular, we discover that overlapping communities lead to a global core-periphery network structure.

The core-periphery structure captures the notion that networks are composed of a densely connected core and a sparsely connected periphery [18, 68, 128]. Traditional community detection methods identify the network core as a single giant ‘miscellaneous’ community [32, 97], and thus they lead to the conclusion that the core is absent of network
communities. However, the AGM can detect more communities in the core than in the periphery, because we observe that the nodes in the center of the network belong to more communities than the periphery nodes. Our results suggest a novel connection between the community structure and the core-periphery structure: a network core forms as a result of a convolution of many overlapping communities. In the analyses on thirteen different networks ranging from social and information networks to biological networks, our results consistently support this connection. Additionally, our methodology demonstrates that some networks have secondary cores while other networks have only one (global) core.

1.1.5 Step 5: Extensions (Chapters 9, 10)

In the final step, we extend our AGM in two different directions. Our first extension discovers a different kind of network communities and handles directed networks as well as undirected networks. The second extension is to detect communities using the attributes of the nodes as well as the network structure.

We continue to apply our model-based approaches (as we did with AGM and BigCLAM) for building these extensions, that is, we develop generative models for networks, and then we detect communities by fitting the model to a given network. For developing generative models, we build on the generative process of AGM. For devising fitting algorithms, we build on the procedures of BigCLAM.

Extension 1: Detecting 2-mode communities and cohesively connected communities in directed and undirected networks. Let’s consider a predator-prey (who feeds on whom) network among marine life (e.g., the Florida bay foodweb [142]). In such network, one can imagine a “community” of predators at the bottom of the sea (e.g., crabs, catfish, or croakers). However, the members of this community are not linked each other; rather, they prey upon the same set of smaller creatures at the bottom of the sea (e.g., worms and clams). We note that the traditional community detection methods (including our AGM) may not be suitable for detecting this community of predators for two reasons. First, the input network is directed while the existing methods assume an undirected input network. Second and more importantly, this community is not densely interlinked as opposed to what existing methods assume. We refer to this kind of communities as “2-mode” communities
and distinguish them from “cohesive” communities where the member nodes are densely connected to each other within the communities.

Now we observe that, while two kinds of communities (cohesive or 2-mode) exist in two kinds of networks (directed or undirected), the previous methods focus only on one configuration: Identifying cohesive communities in undirected networks. In this extension, we build on our AGM and develop a method to cover all four different configurations. We achieve this by introducing directed memberships between nodes and communities by distinguishing whether the node receives or creates links (or both) to other members of the community. Directed affiliations allow us to simultaneously represent cohesive as well as 2-mode communities. In cohesive communities node affiliations are bidirectional (a node links to other members and also receives links from them); 2-mode communities are modeled with unidirectional memberships where some members mostly create links while others mostly receive them. Moreover, directed affiliations can be applied to both directed and undirected networks as they assume directed edges.

**Extension 2: Community detection in networks with node attributes.** In our second extension, we detect communities by incorporating node attributes as well as the network structure. Our intuition here is that the node attributes are as important as the network structure because the nodes in the same communities are likely not only to be connected to each other [158], but also to have similar attributes each other [35, 102].

In this extension, we focus on developing a high performance (accurate and scalable) community detection method for networks where node attributes are also available. To do this, we extend our AGM to incorporate both network and the node attributes. To model the generative process for the network edges, we employ the generative process from AGM. For the node attributes, we consider a set of logistic models so that members of each community tend to share common traits. Community detection with our extension can be achieved by fitting our generative model as we did with AGM.
Chapter 2

Background and Related Work

In this chapter, we review the basic concepts and terminology used in this thesis and introduce the notation. Next, we survey the previous works on network community detection.

2.1 Basic Concepts and Definitions

In this section, we briefly define concepts and terminology that we will use throughout the thesis.

2.1.1 General Graph-theoretic Concepts

Network data is usually represented as a graph. A graph \( G(V, E) \) is defined with a node set \( V \) and an edge set \( E \). We interchangeably use terms “vertex” or “node” to refer to elements of the node set \( V \), and similarly “edge” or “link” to refer to elements of the edge set \( E \).

A convenient way to represent a graph \( G(V, E) \) is by using an adjacency matrix \( A \in \{0, 1\}^{|V| \times |V|} \), where \( A_{uv} = 1 \) if \((u, v) \in E\) and 0 otherwise.

Next, we define several basic concepts for graphs:

- **Bipartite graph**: Graph \( G(V, E) \) is bipartite if node set \( V \) can be partitioned into two disjoint sets \( V_1, V_2 \) such that there exists no edges between the nodes of the same partition (i.e., between the nodes of \( V_1 \) or between the nodes of \( V_2 \)).
CHAPTER 2. BACKGROUND AND RELATED WORK

- **Directed and undirected graph**: Graph $G(V, E)$ is undirected if $(u, v) \in E \iff (v, u) \in E$, i.e., edges are unordered pairs of nodes. If edges are ordered (i.e., $(u, v) \in E$ does not necessarily mean $(v, u) \in E$), graph $G(V, E)$ is directed.

- **Connected component**: A connected component is a maximal set of nodes where a path exists for every pair of the nodes in the set. In directed graphs, a connected component is weakly connected if there exists an undirected path for every pair of the nodes in the set, and strongly connected if there exists a directed path.

- **Induced subgraph**: An induced subgraph $G_s(V_s, E_s)$ of a graph $G(V, E)$ is a subset of nodes and all their edges: $V_s \subset V$ and $E_s = \{(u, v) \in E : u, v \in V_s\}$.

- **Node degree**: In undirected graphs, node $u$ has degree $d(u)$ if it has $d(u)$ incident edges. For directed graphs, we distinguish in-degree $d_{in}(u)$, which is the number of incident edges to node $u$, and out-degree $d_{out}(u)$, which is the number of edges pointing from the node.

- **Triad**: A triad (or a triangle) is a triple of connected nodes $(u, v, w)$, i.e., $(u, v), (v, w), (w, u) \in E$.

- **Clique**: A clique is a set of nodes where every pair of nodes has an edge between them. For example, a triad is a clique with three nodes.

- **Ego-network**: An ego-network of node $u$ of graph $G(V, E)$ is an induced subgraph of $G(V, E)$ consisting of the neighbors of $u$. The ego node $u$ is not included unless specified.

2.1.2 The Community Detection Problem

We formally define the problem of network community detection.

**Problem 1. (Network Community Detection)** Given a network $G(V, E)$, we discover communities $C_1, C_2, \ldots, C_K$ where each community $C_i \subseteq V$ is a node set consisting of node $u \in V$ that belong to $C_i$. 
In other words, network community detection is to cluster the nodes in a given network using the network connectivity structure. Note that the total number of communities \( K \) is to be estimated as well.

If there is an additional constraint that \( C_i \)'s are mutually exclusive (\( C_i \cap C_j = \emptyset \)), we refer to such version as *non-overlapping community detection*. In this thesis, however, we focus on the problem without the mutual exclusiveness constraint. To distinguish from non-overlapping community detection, we call Problem 1 *overlapping community detection*.

Note that the only input to the problem is a network \( G(V, E) \), i.e., in community detection, we usually do not consider any node attribute or edge attribute. In this thesis, we will mostly focus on community detection without using attribute data. Only in Chapter 10 we will cover community detection in networks with node attributes.

### 2.1.3 Community Scoring Functions

We briefly discuss a few famous mathematical functions that are directly related to network community detection. We refer to such functions as *community scoring function* in the sense that, for a set of nodes, the functions output a quality score which characterizes how much the connectivity structure of a given set of nodes resembles that of a community. A number of methods \([32, 115]\) achieve community detection by finding sets of nodes with high scores.

In practice nearly all scoring functions build on the intuition that communities are sets of nodes with many connections between the members and few connections from the members to the rest of the network. Here, we describe three well-known community scoring functions. We briefly note that, in \([153]\) and \([155]\), we have evaluated a wide range of community scoring functions on their ability to discover communities.

Given a set of nodes \( S \), we consider a function \( f(S) \) that characterizes the community-quality of a given set of nodes \( S \). Let \( G(V, E) \) be an undirected graph with \( n = |V| \) nodes and \( m = |E| \) edges. Let \( S \) be the set of nodes, where \( n_S \) is the number of nodes in \( S \), \( n_S = |S| \); \( m_S \) the number of edges in \( S \), \( m_S = |\{(u, v) \in E : u \in S, v \in S\}| \); and \( c_S \), the number of edges on the boundary of \( S \), \( c_S = |\{(u, v) \in E : u \in S, v \notin S\}| \); and \( d(u) \) is the degree of node \( u \).
• **Conductance:** \( f(S) = \frac{c_S}{2m_S + c_S} \) measures the fraction of total edge volume that points outside the cluster [135].

• **Normalized Cut:** \( f(S) = \frac{c_S}{2m_S + c_S} + \frac{c_S}{2(m - m_S) + c_S} \) [135].

• **Modularity:** \( f(S) = \frac{1}{4}(m_S - E(m_S)) \) is the difference between \( m_S \), the number of edges between nodes in \( S \) and \( E(m_S) \), the expected number of such edges in a random graph with identical degree sequence [114].

### 2.2 Previous Work on Network Community Detection

In this section, we briefly discuss previous work on network community detection. We describe how different methods detect communities and what are the intuitions behind the methods. For more complete surveys, refer to [50, 121, 132, 143, 150].

#### 2.2.1 Non-overlapping Community Detection Methods

Early works on network community detection were heavily influenced by the theory on the strength of weak ties [58]. This theory led researchers to think of networks as consisting of dense clusters that are linked by a small number of long-range ties. This view of networks implies that, by identifying long-range ties, the networks can be partitioned into several dense clusters (i.e., communities). Following this view, many methods were developed to separate the network into dense clusters (i.e., communities). In these methods, the detected communities do not overlap because each node is assigned to only one community.

**Spectral clustering.** Spectral clustering was originally proposed for the algorithms for parallel computation. The task in the algorithms is to partition a given network so that the communication (i.e., the number of edges) between the partitions is minimized, and the spectral clustering algorithms provide an approximation solution for the task.

In spectral clustering, one constructs Laplacian \( L \) which has components \( L_{uv} = d(u) \delta(u, v) - A_{uv} \), where \( d(u) \) is a degree of node \( u \) and \( \delta(u, v) \) is the Kronecker delta (\( \delta(u, v) = 1 \) if \( u = v \) and 0 otherwise). One computes a matrix where each column is the eigenvector of the Laplacian and each row of the matrix corresponds to the each node. Finally, one partitions the rows of the eigenvector matrix.
Early attempts [149] focused on minimizing cut $c_S$ (which is the number of the edges between the partitions) and noted that minimizing cut leads to finding trivially small clusters. Thus, later methods [135] aimed to minimize the Normalized cut (Section 2.1.3) by modifying Laplacian. Another variation [114] of spectral clustering was developed for optimizing Modularity (described below).

Due to its simplicity and theoretical guarantee, spectral clustering has been extensively used for graph partitioning. To further speed up spectral clustering, Karypis et al. [77] and Dhillon et al. [38] developed multi-level approaches for graph partitioning.

**Modularity.** Newman [114] proposed Modularity (Section 2.1.3), which measures the concentration of the edges compared to that under a null model within each community. Given a network, a null model is the case where we randomly shuffle the edges of the network while preserving the degree of individual nodes [109]. Modularity is the sum of the differences between the actual number of the edges in each cluster and the expected number of the edges under the null model in each community.

Because optimizing Modularity is NP-complete [121], many methods have been developed to provide approximate solutions. Clauset et al. [32] proposed a fast greedy heuristics, and Blondel et al. [16] developed a multi-level greedy method that easily scales to networks with millions of nodes and billions of edges. Other than greedy approaches, Newman [114] proposed a spectral clustering method and Duch and Arenas [39] developed a method based on simulated annealing. These non-greedy methods achieve better Modularity than greedy methods while sacrificing scalability.

Even though Modularity is considered one of the most popular community scoring functions, it has a theoretical limitation known as “the resolution limit” [51]. Fortunato et al. [51] proved that Modularity is unable to distinguish communities when the network is too “large” compared to community sizes, and thus it is not suitable for community detection in networks with millions of edges.

**Betweenness centrality.** Girvan and Newman [53] devised a community detection method based on the notion of *betweenness centrality*. Betweenness centrality of an edge is the number of the shortest node-to-node paths going through the edge. Edges with high betweenness centrality correspond to the long-range ties that are described in the theory of
Thus, the method separates a given network into communities by identifying such long-range ties. The method ranks all the edges by betweenness centrality, removes the edge with the highest value, and the recalculates the betweenness centrality of the remaining edges. This method was one of the first successful attempts that revealed the community structure in social, information, and biological networks [53]; however, the method is slow for large networks because the recalculation step is computationally expensive.

**Label Propagation.** Label Propagation is an extremely fast heuristic that can partition the nodes in a given network into different “labels” (i.e., communities). At initialization, each node is assigned to its own label. Then, the nodes iteratively adopt the label of the majority of their neighbors until convergence. Raghavan et al. [124] first used Label Propagation to detect communities, and showed empirically that Label Propagation tends to take a linear time in the number of the nodes for convergence.

One drawback of Label Propagation is that it often leads to finding trivially small communities if some labels dominate the others. To fix this drawback, Ugander and Backstrom [141] proposed a “balanced” Label Propagation where one can specify the desired range of the sizes of detected communities.

**Random walks.** Random walk on the graph is a process where we begin from a starting node and then move to a neighbor of the node that is chosen at uniformly random. Several methods have used random walks as an intermediate step for community detection.

The Infomap algorithm [129] performs random walks from multiple different starting points and then seek node partitions (i.e., communities) with which one can minimize the entropy of the traces of random walks.

Local Spectral Clustering [7, 8] uses random walks for community detection around a specific seed node. The goal here is to detect communities that the seed node belongs to. The local spectral clustering method performs random walks with restarts from the seed node [8]. Then the method sorts the nodes by their random walk scores and takes a community of top $k$ nodes where the Conductance of the node set is maximized. The intuition here is that the nodes with high random walk scores from the seed node correspond to the well-connected nodes around the seed node. The method has a theoretical guarantee on the Conductance of the detected community [8] and its time complexity is linear to the
size of the detected community (regardless of the size of the network). For this reason, this method was able to determine the existence of good-conductance communities in large networks [97]. The drawback of this method is that the quality of the detected community highly depends on the seed node [54].

**Stochastic Block Models.** Stochastic Block Model [67] assumes that an input network is generated in the following way. Each pair of blocks has an edge probability between the nodes in the blocks. Each node is assigned to one block and a pair of nodes is connected with the edge probability between the blocks that they belong to. Given an input network, community detection can be achieved by fitting the model, i.e., inferring the best-fit block assignments (community assignments) and block-to-block edge probabilities. As a fitting algorithm, one usually employs a Markov Chain Monte Carlo method or the spectral clustering method [76].

One well-known caveat of Stochastic Block Model is that it ignores variation in node degree because of its assumption that the nodes in the same block would have the same node degree. To fix this caveat, the Degree-corrected Stochastic Block Model [76] is proposed. This model additionally assumes that each node has a latent parameter for its degree. The network edges are generated by the combination of the degree parameters and the block memberships of the nodes. A MCMC method similar to the standard Stochastic Block Model is applied for fitting the model.

### 2.2.2 Overlapping Community Detection Methods

After the surge of non-overlapping community detection methods described in [2.2.1], it was realized that the nodes in many real-world networks belong to multiple communities at once. For example, a person in a social network belongs different social communities depending on her schools, interests, and affiliations. This fact leads to overlapping community structure in networks, which non-overlapping communities cannot capture. To account for this aspect, a great deal of efforts has been devoted to develop overlapping community detection methods. Each overlapping community detection method has its own intuition and definition for communities. We discuss what is the intuition of the methods and how the methods detect communities.
Clique-based methods. A single node may belong to multiple different cliques at the same time. Based on this fact, several methods use cliques as a basis for detecting overlapping communities.

The first such method (and the most famous one) is the Clique Percolation Method developed by Palla et al. [118]. The method has an input parameter $k$ and identifies all cliques of $k$ nodes. Then, the method merges two $k$-cliques if they share $k - 1$ nodes until convergence, and each of the merged cliques is a detected community. Nodes may belong to multiple communities if they belong to more than one cliques that cannot be merged together. The method was the first successful attempts to detect overlapping communities in social and biological networks. However, the method is scalable up to networks with only a few thousands node due to its exponential computational complexity [118].

EAGLE [134] and GCE [89] also begin by identifying all maximal cliques. EAGLE merges cliques using a hierarchical agglomerative framework where the method builds a dendrogram by iteratively merging the pair of cliques with the maximum similarity. The optimal cut on the dendrogram is chosen by Modularity. Rather than merging cliques, GCE expands the maximal cliques by adding adjacent nodes. GCE uses a local fitness function (which is similar to Conductance) to determine which node to add.

Link partitioning methods. Link partitioning methods use the intuition that if one partitions links (edges), the nodes may belong to multiple partitions (communities). Link Clustering [4] is the most prominent approach among such methods. Link Clustering partitions the edges using agglomerative clustering, where the method merges two edges sharing one node with the maximum Jaccard similarity between the neighbors of the two unshared nodes. The dendrogram is cut at the point where a quality function called the partition density [4] is maximized. The advantage of Link Clustering is that the inferred dendrogram describes the hierarchy among overlapping communities.

Similarly, Evans and Lambiotte [42] computes a similarity weight between the links of a given network. Then, the method constructs a line graph, where the nodes are the links of the given network and the edges are the similarity weights between the links. Finally, the method partitions the line graph (i.e., partitioning the links of the given network) by applying a non-overlapping community detection method [17].

Link partitioning methods provide users with flexibility to choose any non-overlapping
clustering method to partition links. It will be an interesting future work to explore other non-overlapping clustering methods for link partitioning.

**Mixed Membership Stochastic Block Model.** Mixed Membership Stochastic Block Model [5] is an extension of Stochastic Block Model for detecting overlapping communities. The model assumes the following generative process for a network. Like Stochastic Block Model, each pair of blocks has an edge probability between them. However, instead of belonging to one block, each node has a *membership distribution* to belong to a particular block. A pair of nodes is connected with the edge probability corresponding to their block assignments, which are sampled from their membership distributions. By fitting the model to a given network, one can infer the block membership distributions of the nodes.

Mixed Membership Stochastic Block Model focuses on modeling the edges between different communities as well as the edges within the same communities, whereas other community detection methods focus only on the latter. This difference gives the model flexibility to better explain the edges in the network. In addition, the model is a Bayesian probabilistic model, which is mathematically well-founded and allows us to apply a variety of inference techniques. However, the model has a drawback in identifying overlapping community memberships due to its assumption that a node’s community memberships are a probability distribution (i.e., the sum of the memberships is one). Under such assumption, a node cannot have high (namely, higher than 0.5) membership probabilities to more than one communities. In other words, a node cannot belong to multiple communities simultaneously. In Section 4.3, we will discuss the limitation of this assumption on modeling overlapping communities.

**Local methods.** Local methods are based on identifying a “local” community around a seed node. These methods detect a community for each of multiple seed nodes. Because each local community is identified independently, the detected communities may overlap with each other.

LFM [87] starts from a random seed node and expands a community until the community reaches the maximum value of the fitness function. The fitness function depends on the number of edges in the community and the number of edges on the boundary and has an input parameter $\alpha$ that controls the size of the detected community. After detecting one community, the method randomly chooses another seed node that does not belong to any
community, and then repeat the process.

MONC \cite{64} adopts a similar procedure with a modified fitness function. Unlike LFM, however, MONC’s fitness function is parameter-free because MONC can estimate the optimal value of $\alpha$ automatically. In addition, MONC is faster than LFM because MONC skips detecting redundant communities that are included by other communities.

DEMON \cite{34} combines the local approach with Label Propagation. DEMON detects communities by Label Propagation in the ego network of each node, and then merge the detected communities if their similarity is higher than a given threshold. Due to high scalability of local approaches and Label Propagation, DEMON easily scales to networks with hundreds of thousands of nodes. However, the quality of the method is highly dependent on the value of threshold for merging communities and it is not clear to determine the optimal value for the threshold.

Other methods.

Some community detection methods are based on Non-negative Matrix Factorization (NMF) \cite{90}. Given an adjacency matrix $A$ of a given network, NMF learns non-negative factor matrices $F, H$ whose product $FH^T$ can approximate $A$. From learned $F, H$, one then can infer community memberships of the nodes. Wang et al. \cite{146} is one of the first successful attempts to detect communities with NMF. Psorakis et al. \cite{122} proposed a multi-level Bayesian framework for NMF where $F, H$ are generated from Gamma prior distributions. Despite its mathematical elegance, NMF-based methods are not very scalable due to their complexity is quadratic in the number of nodes (the methods have to go through all pairs of the nodes.)

Label Propagation, which is originally a non-overlapping community detection method (Section 2.2.1), can be extended for overlapping community detection. Since each run of the method leads to different partitions, one can infer overlapping communities by aggregating the results from multiple runs. Gregory \cite{59} developed such method, and SLPA \cite{150} used a similar strategy with a modified label propagation process. Due to the simplicity of Label Propagation, these methods scale to networks with hundreds of thousands of nodes.
2.3 Table of Symbols

Tables A.1, A.2, A.3 and A.4 describe the symbols used in the thesis.
Chapter 3

Ground-truth Communities

3.1 Introduction

In this chapter, we define a robust notion of ground-truth communities. To define ground-truth communities, we begin by distinguishing between structural and functional definitions of communities. We then argue that the goal of community detection is to extract functional communities based on the connectivity structure of the nodes in the network. For this reason, we claim that functional communities can be regarded as reliable ground-truth communities. We then describe how we identify networks where ground-truth communities (functional communities) are explicitly specified.

Developing a notion of ground-truth provides profound impacts throughout this thesis. The first impact is to allow for development of new community detection methods. This impact is due to the fact that studying ground-truth communities can improve our understanding of how communities manifest themselves in networks. In the following chapters, we make novel observations that the common wisdom on community overlaps does not hold on the ground-truth communities (Chapter 4). We then develop reliable community detection methods based on our novel observations (Chapter 5, 6, and 7).

The second impact is to allow us to measure the performance of community detection methods quantitatively (Chapter 6 and Chapter 7). Previously, the performance of community detection methods is usually evaluated by manual inspection. For each detected community, an effort is made to interpret it as a “real” community by identifying a common...
property or external attribute shared by all the members of the community. For example, when examining communities in a scientific collaboration network, we might by manual inspection discover that many of detected communities correspond to groups of scientists working in common areas of science [117]. Such anecdotal evaluation procedures require extensive manual effort, are non-comprehensive, and are limited to small networks. However, our ground-truth communities allow for quantitative and large-scale evaluation of network community detection methods.

3.2 Intuition behind ground-truth communities

We begin by explaining the intuition behind our definition of the ground-truth communities. We distinguish between structural and functional definitions of communities. A structural definition of communities is a set of nodes with a particular connectivity structure (e.g., set of nodes with high edge density or set of nodes with high Modularity [117] score). A functional definition of communities is a set of nodes with a common function, which can be common role, affiliation, or attribute [46, 58].

With these two definitions of communities, community detection process generally follows a two-step procedure: First, one discovers communities based on a structural definition. Second, one argues that the discovered communities correspond to functional communities. For example, Palla et al. [118] identified structural communities by identifying sets of overlapping k-cliques on protein-protein interaction networks. Then, they found that these structurally defined communities of proteins correspond to functional modules of proteins. In this example, communities are extracted based on the structural definition and then evaluated based on the functional definition by arguing that ‘belonging to the same functional module’ is the common function of nodes.

Our approach takes the opposite direction: Rather than developing a structural definition of communities, we begin with collecting large-scale networks where functional communities are explicitly labeled, and we consider such functional communities as “ground-truth” communities. Our main insight here is that, while explicitly labeled structural communities are nearly impossible to obtain, in some networks, functional communities are
CHAPTER 3. GROUND-TRUTH COMMUNITIES

3.3 Networks with Ground-truth Communities

Overall we consider 6 large social, collaboration and information networks, where for each network we have a graph and a set of functionally defined ground-truth communities. We describe the networks that we consider and the meaning of the ground-truth communities in each of the networks.

Social networks. First we consider four very different online social networks (the LiveJournal blogging community [10], the Friendster online network [107], the Orkut social network [107], and the YouTube social network [107]) where users create explicit groups which other users then join. Such groups serve as organizing principles of nodes in social networks and are focused on specific topics, interests, hobbies, affiliations, and geographical regions. Groups range from small to very large and are created based on specific topics, interests, hobbies and geographical regions. For example, LiveJournal categorizes groups into the following types: culture, entertainment, expression, fandom, life/style, life/support, gaming, sports, student life and technology. Overall, there are over three hundred thousand explicitly defined groups in LiveJournal. Similarly, users in Friendster as well as in Orkut explicitly labeled in the data. Thus, we use such functional communities to define ground-truth communities.
and YouTube define topic-based groups that others then join. The Friendster and Orkut networks have more than a million explicitly defined groups and each user can join to one or more groups. We consider each such explicitly created group as a ground-truth community.

The LiveJournal network was provided to us by Lars Backstrom [10], the Friendster network was made public by the Internet Archive[1], and the Orkut and YouTube networks were kindly provided to us by Alan Mislove [107].

Amazon product co-purchasing network. The second type of network data we consider is the Amazon product co-purchasing network [92]. The nodes of the network represent products and edges link commonly co-purchased products. Each product (i.e., node) belongs to one or more hierarchically organized product categories and products from the same category define a group which we view as a ground-truth community. This means members of the same community share a common function or role, and each level of the product hierarchy defines a set of hierarchically nested and overlapping communities. We crawled this network using the Amazon API [92].

DBLP collaboration network. We consider the collaboration networks of DBLP [10], where nodes represent authors and edges connect nodes that have co-authored a paper. We use publication venues as ground-truth communities which serve as proxies for highly overlapping scientific communities around which the network then organizes. This network was provided to us by Lars Backstrom.

Ground-truth network characteristics. Table 3.1 gives the dataset statistics. Observe that the size of the networks ranges between hundreds of thousands to hundreds of millions of nodes and billions of edges. The number of ground-truth communities varies from hundreds to millions and there is also a range in ground-truth community sizes and node membership distribution.

For each of these networks we identified a sensible way of defining ground-truth communities that serve as organizational units of these networks. Even though our networks come from very different domains and individual labels may be noisy or even incomplete, the results we present in this thesis are robust and consistent across all the datasets. Our work is consistent with the premise that is implicit in all network community literature:
members of “real” communities share some (latent/unobserved) property or affiliation (i.e., functional definition of communities) that serves as an organizing principle of the nodes and makes them well-connected in the network (i.e., structural definition of communities).

All our networks are complete and are publicly available at [http://snap.stanford.edu/data](http://snap.stanford.edu/data).

**Data preprocessing.** To represent all networks in a consistent way we drop edge directions and consider each network as an unweighted undirected static graph. Because members of the group may be disconnected in the network, we consider each connected component of the group as a separate ground-truth community. However, we allow ground-truth communities to be nested and to overlap (i.e., a node can be a member of multiple groups at once).

**Figure 3.1:** Ground-truth community size distribution. Complementary cumulative distribution function $F_c(s)$ of the size of ground-truth communities, $s$. The size of a ground-truth community denotes the number of nodes belonging to the ground-truth community.
CHAPTER 3. GROUND-TRUTH COMMUNITIES

3.4 Distributions of Ground-truth Communities

Next we present the distribution of the various properties of ground-truth communities. Our goal here is to investigate properties of ground-truth communities and demonstrate that such sets of nodes in fact correspond to “real” network communities.

Previous literature found that the size of communities, i.e., the number of the nodes in communities, follows a heavy-tailed distribution [4, 118, 163]. Figure 3.1 shows the CCDF (complementary cumulative distribution function) of the sizes of ground-truth communities in the 6 networks. The distribution appears to follow a heavy-tailed distribution, which for LiveJournal, YouTube and Amazon appears to be power-law.

Figure 3.2 shows that the CCDF of the distribution of the number of communities a node belongs to. We observe it exhibits a power-law decay, but the distributions do not show a long tail in some data sets such as Orkut, DBLP, and Amazon. This is in accordance with Palla et al. [118] who reported that the distribution of node memberships, i.e., the number

Figure 3.2: Node membership distribution. Complementary cumulative distribution function $F_c(m)$ of the number of communities that a node belongs to, $m$. 
CHAPTER 3. GROUND-TRUTH COMMUNITIES

Figure 3.3: **Community overlap distribution.** Complementary cumulative distribution function \( F_c(o) \) of the size of overlaps between pairs of ground-truth communities, \( o \). The size of an overlap is the number of the nodes that belong to the overlap.

of the communities that a node belongs to, tends to follow a power-law.

Last, we also examine the statistics of the community overlaps. We focus on overlaps between a pair of ground-truth communities and report the absolute and fractional size of the overlap between two communities. Figure 3.3 shows the distribution of the absolute overlap sizes. We observe that the distributions follow a power-law, as also observed by Palla et al. [118] on detected (not ground-truth) communities. In addition, we investigate how ground-truth communities overlap: Do ground-truth communities overlap in a nested structure? Or, do they overlap only for a small fraction of members? To do this, we measure the fraction \( f \) of the size of the overlap \( A \cap B \) between two communities \( A, B \) to the size of the smaller community, \( \min(|A|, |B|) \) \( (f = |A \cap B|/\min(|A|, |B|)) \). \( f \) being close to 1 means a nested structure where the larger community includes the smaller one, and small \( f \) means overlap in the fringe. Figure 3.4 plots the distribution of the overlap
fraction $f$. The Amazon network shows high probability at $f = 1$ because the ground-truth communities form a nested structure by construction. In social networks and the DBLP network, most overlaps take a small fraction of individual communities, which is reasonable as each community has its own special interests.

3.5 Use Case: Evaluating Structural Definitions of Communities

In addition to the work in this thesis (i.e., understanding the connectivity structure of network communities and developing novel community detection methods), the ground-truth communities allow us to evaluate the existing structural definitions of network communities quantitatively.
A plethora of structural definitions for communities has been proposed yet it has been unclear which definitions should be preferred. With ground-truth communities, now we can examine how well various structural definitions of network communities correspond to ground-truth communities [153, 155]. A good structural definition of a community would be such that it would correspond to connectivity patterns of ground-truth communities.

In [153, 155], we study 13 commonly used structural definitions of communities. Each such definition corresponds to a scoring function that scores a given set of nodes how ‘community-like’ it is, i.e., a scoring function assigns high score to sets of nodes that closely resemble functional communities. We measure these 13 community scoring functions in two aspects; quality and robustness.

To measure the qualities of community scoring functions, we consider an axiomatic approach and define four intuitive properties that communities would ideally have. Intuitively, a ‘good’ community is 1) cohesive, 2) compact, 3) internally well connected, and 4) well separable from the rest of the network. We measure which scoring function tends to prefer good communities regarding to each of the four properties. This allows us to characterize which connectivity patterns a given structural definition detects and which ones it misses. To measure the robustness of community scoring functions, we measure how scoring functions behave when we perturb our ground-truth communities by using four different perturbation strategies.

Overall, evaluation shows that among the scoring functions considered here those that are based on triadic closure [148] and the Conductance score [135] best capture the structure of ground-truth communities. For more details, refer to [153, 155].

### 3.6 Related Work

We discuss the previous work that is related to the definitions of network communities. We distinguish the previous work by whether it was about functional definitions of communities or structural definitions of communities.

**Studies on functional definitions of communities.** There have been studies on the functional definitions of communities, which we use in this thesis to define ground-truth communities. These studies observe the properties of network communities that are defined
by functional definitions. For example, [10] focused on how users decide to join social
groups in the LiveJournal network. [107] studied the size distributions of social groups in
the Orkut and YouTube social networks. However, the key difference in our work lies in
the goal of the study; whereas the goal of existing work is to measure the statistical prop-
erties of communities, our goal here is to develop a novel community detection method
by understanding the structure of ground-truth communities. In the following chapters, we
build new community detection methods based on our observations of structural properties
of ground-truth communities. Another important difference in our work is that we focus on
how these ground-truth communities overlap with each other, whereas the previous work
did not focus on community overlaps.

Studies on structural definitions of communities. More broadly, there have been studies
on the structural definitions of network communities, either in theoretical ways or empirical
ways.

Theoretical analysis has been performed for a few most widely used structural defi-
nitions such as Modularity [51] and Conductance [7, 54]. For example, Gleich and Se-
shadhri [54] mathematically proved the existence of node sets with high conductance in
networks with high clustering coefficient. Fortunato and Barthelemy [51] showed that
Modularity may not detect communities that have too few edges compared to the total
number of edges in the network. More general theoretical analysis includes the work of Meilă [105] which studied the axiomatic criterion for community scores such as stabil-
ity [144].

Empirical approach analyzed the properties of structural definitions in real-world net-
works. One recent example is by Leskovec et al. [96], which evaluated a wide range of
structural definitions of network communities in large real-world networks. To evaluate
structural definitions of communities, [96] used detected communities by the Local Spec-
tral method [7]. However, using communities detected by a specific community detection
method would introduce a structural bias introduced by the detection method, as shown in
[1]. To avoid such bias, our work mentioned in Section 3.5 evaluated the structural def-
initions by using ground-truth communities, which are explicitly specified by individual
nodes rather than by community detection methods.


3.7 Conclusion

In this chapter, we defined ground-truth communities. To do this, we identifies six social and information networks where the nodes in the networks state their memberships to ground-truth communities. We showed the distributions of structural properties of ground-truth communities.

In the next chapter, we study the connectivity structure of the ground-truth communities and their overlaps. We observe that the ground-truth communities overlap with each other in a very different way than what has been assumed in previous literature.
Chapter 4

Observations of Community Overlaps

4.1 Introduction

In the previous chapter, we identified six different large scale networks where the nodes in the networks explicitly state their memberships to various groups. In the six networks, we defined ground-truth communities to be the groups to which the nodes state memberships. Then we discussed the basic statistics of the ground-truth communities.

The availability of reliable ground-truth communities allows us to observe the connectivity structure of real communities and their overlaps. We analyze how nodes of ground-truth communities connect to each other, how they connect to the rest of the network, and how they overlap. This way we can empirically study on a large scale how real communities map to the underlying network structure.

In particular, we find that the nodes residing in the overlaps of communities are more densely connected than the nodes that reside in the non-overlapping parts of communities. We view this finding particularly surprising because we show that it goes against the conventional wisdom that communities (both overlapping and non-overlapping) are more densely connected than their boundaries or the overlaps themselves [4, 118]. Thus, our findings suggest the need to revisit standard models of community structure to account for the fact that nodes in the overlap of communities are more densely connected.

To sum up, this chapter discusses two main contributions:

- We make a novel empirical observation that nodes in the community overlaps are
more densely connected than the nodes in the non-overlapping parts of communities.

- We show that the existing methods cannot identify densely overlapping communities. In other words, we show that the existing methods make an implicit assumption that the nodes in community overlaps are less densely connected than in the non-overlapping parts of communities.

### 4.1.1 Outline of Chapter

Here is the outline of the chapter:

- We study the connectivity structure of ground-truth communities and community overlaps, and we find that the nodes in the community overlaps have denser edge connectivity than the nodes in the non-overlapping part of communities. This observation is striking because it goes against the common assumption in the existing work on community detection: The existing community detection methods assume that community overlaps would have sparser connectivity than individual communities. (Section 4.2)

- We show that the existing community detection methods cannot correctly identify overlapping communities if the communities form dense community overlaps. We consider three classes of community detection methods: clique-based methods, link clustering method, and stochastic block models. (Section 4.3)

### 4.2 Empirical Observations on Community Overlaps

Communities in networks may overlap with each other because nodes may belong to multiple communities simultaneously. As demonstrated in the previous section, ground-truth communities overlap and many nodes belong to multiple communities at once (Figures 3.2 and 3.3).

We study the structure of community overlap simply by asking what is the probability that a pair of nodes is connected if they share membership to \( k \) common ground-truth
CHAPTER 4. OBSERVATIONS OF COMMUNITY OVERLAPS

34

communities. Figure 4.1 plots this probability for all six datasets for which we have the ground-truth community data.

We notice that all curves are steeply increasing. These increasing curves mean that, the more communities a pair of nodes has in common, the higher the probability of the nodes being connected. Notice the effect of shared memberships on the edge probability is very strong. For example, in LiveJournal, if a pair of nodes has 4 communities in common, the edge probability is nearly 50%. To appreciate how strong the effect of shared communities is on edge probability, one has to note that all of our networks have extremely sparse edge density; the overall edge probability is \( \approx 10^{-5} \). However, as soon as a pair of nodes shares two communities, their probability of linking increases by four orders of magnitude (from \( 10^{-5} \) to \( 10^{-1} \)).

Such high edge probability in the overlap is not just the consequence of the fact that the overlapping nodes have high degrees. The overlapping nodes have high degrees in general because they form edges across multiple communities, but this high degree alone cannot explain such high edge probability. For example, the average degree of the nodes in the overlaps in LiveJournal is around 50. If we assume the edges are created according to the node degrees only (i.e., the configuration model [109]), the edge probability between two overlapping nodes is still extremely low (\( \sim 10^{-5} \)).

We note that all other data sets exhibit similar behavior — the probability of a pair of nodes being connected approaches 1 as the number of common communities increases. Whereas in online social networks the edge probability exhibits a diminishing-returns-like growth, in DBLP, it appears to follow a threshold-like behavior.

In retrospective the above result is very intuitive. For example, in the context of social networks two students that belong to both a Tuesday salsa club and a Sunday Movie club are more likely to meet each other than if they would belong to only a single club. Thus, the more communities nodes share, the more likely they are to meet and interact. Communities thus serve as organizing principles of nodes in social networks and are created on shared affiliation, role, activity, social circle, interest or function.

What is perhaps more surprising is that we later show in Section 4.3 that most present overlapping community detection methods make an implicit assumption that is exactly the
Chapter 4. Observations of Community Overlaps

Figure 4.1: Community overlaps are more densely connected than the non-overlapping parts of communities. Edge probability $P(k)$ between two nodes given that two nodes share $k$ communities. We observe that $P(k)$ is an increasing function of $k$ in all the networks. For the purpose of this plot, we use the 5,000 ground-truth communities that are most cohesive in each data set [153].

Opposite. Most present methods and models of communities [4, 5, 131, 122, 118, 42] implicitly assume that nodes in community overlaps are less densely connected than nodes in the non-overlapping parts of communities. This assumption is prevalent in present approaches. This assumption further implies that pairs of nodes that share two or more communities have a lower probability of being connected. So, given that edge probability increases with the number of overlapping communities, this means that most present overlapping community detection methods are not able to correctly identify community overlaps — they either confuse the overlap as a separate community or identify multiple overlapping communities as a single one. We give further details in Section 4.3.
Figure 4.2: Three different definitions of network communities. Three networks (top) and corresponding adjacency matrices (bottom). We find (a) that as nodes share multiple communities, they are more likely to link which leads to densely connected community overlaps. However, most existing community detection methods either assume that (b) communities do not overlap or that (c) community overlaps are less well-connected than the non-overlapping parts of communities. Moreover, most existing community detection methods cannot properly detect communities with overlaps as in (a).

4.3 Consequences for Present Overlapping Community Detection Methods

Early works on community detection were heavily influenced by the strength of weak ties theory [58] and built on the intuition that communities are non-overlapping sets of nodes that are internally well connected with a few connections to the rest of the network. Such intuition led to development of graph partitioning approaches to community detection [38, 47, 61, 74, 77, 135, 137]. Refer to [143, 132] for surveys of this area. Using similar intuition the methods using betweenness centrality and the notion modularity were also defined [32, 114]. All these methods adopt the definition of communities as depicted in Figure 4.2b. Here we see two non-overlapping communities both depicted as a network and also in the form of an adjacency matrix. Each community has many edges among its members and few edges pointing outside the community. If the network follows the community structure depicted in Figure 4.2b then the graph partitioning methods will be able to correctly identify...
the two communities. Communities in real-world networks tend to overlap as nodes belong to multiple communities at once. However, the above approaches have a common limitation as they assume that communities do not overlap, i.e., nodes cannot belong to more than a single community at once.

However, applying the “strength of weak ties” view to overlapping communities leads to the (unnatural) structure of community overlaps as illustrated in Figure 4.2c. Notice that here where the nodes residing in the overlap of two communities, do not link to each other. This means that nodes that have multiple communities in common have a lower probability of linking. This means that the probability of an edge between a pair of nodes decreases as the number of shared community memberships of a pair increases. Later in the section we will show that this implicit assumption is made by most present community detection methods. For example, Clique Percolation Method [118], Link Clustering [4], Latent Dirichlet Allocation-based methods [5] [15] [65] [164] and many extensions of these methods [89] [134] all operate under the implicit assumption illustrated in Figure 4.2c.

However, our results in Figure 4.1 suggest exactly the opposite. The probability of an edge increases with the number of shared communities between a pair of nodes. This suggest the structure of community overlaps as illustrated in Figure 4.2a. Here the nodes that reside in the area of the overlap between two communities are more likely to be linked.

More importantly, as many methods implicitly assume sparse community overlaps (Figure 4.2c), this means that they are unable to correctly identify the overlaps between communities when communities overlap as illustrated in Figure 4.2a. In particular, most present community detection methods will either confuse the overlap as a separate community or merge the overlapping communities into a single giant community [32].

In the following we show that three widely used overlapping community detection methods all suffer from implicitly assuming sparse community overlaps. In particular, we analyze Clique Percolation Method [118], Link Clustering [4], and Mixed Membership Stochastic Block Model [5].
Figure 4.3: Clique Percolation Method cannot detect communities with dense overlaps. Given a network with two communities and a dense overlap, Clique Percolation Method would report a community that (depending on the parameter settings) either include both communities ((a)), or it would find a small community consisting only of the overlap ((b)).

4.3.1 Clique Percolation

First, we analyze Clique Percolation Method and show that it fails to properly detect two overlapping communities as illustrated in Figure 4.2a.

Clique Percolation Method (CPM) has a single input parameter $k$ that determines the size of the maximal cliques that the algorithm looks for. After finding all the $k$-cliques on the given network, the method merges two $k$-cliques if they share $k - 1$ nodes. Overlaps can happen when the nodes in the overlaps belong to multiple $k$ cliques that cannot be merged. When an overlap is denser, however, nodes in the overlap form many $k$-cliques themselves and the method would merge the $k$ cliques in the overlap. In this case, the method would either identify the overlap as a separate community, or merge adjacent communities through $k$-cliques in the overlap.

For example, Figure 4.3 shows the result of CPM on the network of Figure 4.2a where the overlap between the two (true) communities is denser than the individual communities. When $k = 3$, CPM sees the whole network as a single community because the clique in the overlap connects the cliques in the left community and the right community. When $k = 4$, CPM regards the overlap as a separate community.

In addition to Clique Percolation Method, many other overlapping community detection methods are based on expanding the maximal cliques. These methods (for example, Greedy...
Clique Expansion [89] and EAGLE [134] also suffer from the same problem.

By the same reasoning that we used for Clique percolation, we can see that none of these clique expansion methods is able to discover densely overlapping communities. For example, in the networks in Figure 4.3, neither EAGLE nor Greedy clique expansion could correctly identify the red overlapping nodes. Because all the red nodes form a maximal clique, both methods will regard the red nodes as a single community, and there is no way to tell that the red nodes belong to more than one community.

### 4.3.2 Link Clustering

Next we show that the Link Clustering [4] also suffers from similar problems as the Clique percolation. Link Clustering performs hierarchical clustering on the edges of the given network. For each pair of edges \((i, k)\) and \((j, k)\) that shares a single node \(k\), Link Clustering computes the Jaccard similarity \(JAC(n(i), n(j))\) between the sets of neighbors \(n(i)\) and \(n(j)\) of node \(i\) and \(j\) and builds a dendrogram by merging the pair of edges with the highest Jaccard similarity. Finally, Link Clustering cuts the dendrogram at the point where the partition density, a quality function proposed in [4], is maximized. In the following, we show that Link Clustering does not discover the true communities when their overlaps are more densely connected than each individual community.

We consider a network with two overlapping communities \(I\) and \(J\) with their overlap \(O\) (Figure 4.4). For simplicity, we assume that \(I\) and \(J\) each contain the same number of nodes. We denote the number of nodes in \(I\) or \(J\) by \(X + Y\) and the number of nodes in overlap \(O\) by \(X\). The total number of nodes in the network is thus \(X + 2Y\). Moreover, assume that the nodes in an individual community are connected with probability \(p\), and that the nodes in the overlap have a higher probability of being connected, say \(2p\). Now let’s consider the case where the number of nodes in the overlap is not larger than the number belonging a single community \((X \leq Y)\).

Now consider that Link Clustering computes Jaccard similarity between the neighbors of nodes \(u\) and \(v\). Without the loss of generality we can have one of the four cases:

- (1) \(u \in O\) and \(v \in O\)
- (2) \(u \in I \setminus O\) and \(v \in I \setminus O\)
Figure 4.4: **Link Clustering cannot detect communities with dense overlaps.** A network with two overlapping communities and the outcome of Link Clustering. Link Clustering builds a dendrogram with solid lines that finds the overlap as a separate community. The merger of the overlap and the single community regions (the right and the left communities), described by the dotted lines in the dendrogram, cannot happen as Link Clustering will stop the algorithm because of the decrease in the partition density.

- (3) \( u \in I \setminus O \) and \( v \in O \)
- (4) \( u \in I \setminus O \) and \( v \in J \setminus O \)

We now show that the Jaccard similarity between a pair of edges in case (1) is higher than in case (2), and that (2) is higher than (3), which is naturally higher than (4). This means that Link Clustering first merges edges between nodes in \( O \), and only then merge the edges between nodes in \( I \setminus O \) and edges between nodes in \( J \setminus O \). Last, Link Clustering merges the edges with one endpoint in \( O \) and the other in \( I \setminus O \) (\( J \setminus O \)). This process produces the dendrogram illustrated in Figure 4.4. In particular, this means that regardless where one cuts the dendrogram, Link Clustering fails to correctly identify the community structure of the simple network in Figure 4.4.

To show this more formally we proceed as follows. Let’s consider nodes \( I_1, a_2 \in I \setminus O \).
Chapter 4. Observations of Community Overlaps

and nodes \( o_1, o_2 \in O \), and their neighbors \( n(a_1), n(a_2), n(o_1), n(o_2) \). We show that in expectation the following is true:

\[
\frac{|n(o_1) \cap n(o_2)|}{|n(o_1)| + |n(o_2)|} > \frac{|n(a_1) \cap n(a_2)|}{|n(a_1)| + |n(a_2)|} \geq \frac{|n(a_1) \cap n(o_1)|}{|n(a_1)| + |n(o_1)|}
\]

The above inequalities are equivalent to \( \text{JAC}(n(o_1), n(o_2)) > \text{JAC}(n(a_1) \text{ and } n(a_2)) > \text{JAC}(n(a_1), n(o_1)) \). We have \(|n(o_1)| = |n(o_2)| = 2pX + 2pY\), \(|n(a_1)| = |n(a_2)| = pX + pY\) and we aim to compute the expected values of the sizes of the intersections between \( n(o_1), n(o_2), n(a_1), \) and \( n(a_2) \). For example, \(|n(o_1) \cap n(o_2)| = 4p^2X + 2p^2Y\) in expectation because \( o_1 \) and \( o_2 \) have a common neighbor in single community regions (2Y nodes) with probability \( p^2 \) and in overlap (X nodes) with probability \((2p)^2\). By the same logic, \(|n(a_1) \cap n(a_2)| = p^2X + p^2Y\), and \(|n(a_1) \cap n(o_1)| = 2p^2X + p^2Y\). From these, we derive the following:

\[
\frac{|n(o_1) \cap n(o_2)|}{|n(o_1)| + |n(o_2)|} = p \frac{2X + Y}{2X + 2Y} > \frac{p}{2} = \frac{|n(a_1) \cap n(a_2)|}{|n(a_1)| + |n(a_2)|} \geq \frac{p}{3} = \frac{2X + Y}{3X + 3Y} = \frac{|n(a_1) \cap n(o_1)|}{|n(a_1)| + |n(o_1)|},
\]

where the last inequality \( 1/2 \geq (2X + Y)/(3X + 3Y) \) comes from our assumption that \( X \leq Y \). Therefore, Link Clustering yields the dendrogram in Figure 4.4, which first merges edges in \( O \) and then merges edges in the two non-overlapping parts \((I \setminus O, B \setminus O)\) and only then merges edges between the overlapping and the non-overlapping parts \((O, A \setminus O), (O, B \setminus O)\).

Figure 4.5 gives a concrete example of a network where the result of Link Clustering gives counterintuitive results. Ideal outcome (the true communities) is that green nodes \((1, 2, 3, 4)\) belong to one community, blue nodes \((9, 10, 11, 12)\) belong to the second community and red nodes \((5, 6, 7, 8)\) belong to both communities. As we showed in our analysis, however, Link Clustering merges the edges inside the overlaps (between the red nodes) together and then it merges the nodes in a non-overlapping part (between the green nodes or between the blue nodes). Consequently, Link Clustering regards the overlap of the (true) communities as a separate community and is unable to find that the overlapping nodes belong to two communities at the same time.
4.3.3 Stochastic Block Models

Figure 4.6: The result of a Stochastic block model and the Mixed Membership Stochastic Block Model on a network of two communities with dense overlap. The adjacency matrix of the network in Figure 4.2a is shown and the bold lines denote the partitions discovered by stochastic block models. See main text for further discussion.

Last, we briefly mention that various kinds of stochastic block models \[5, 67, 76\] cannot correctly discover communities with dense overlaps. We show this for three variants of stochastic block models: the traditional stochastic block model \[67\], the Degree-corrected stochastic block model \[76\], and the Mixed Membership Stochastic Block Model \[5\].

The Stochastic block model \[67\] partitions a network into disjoint blocks and assigns an
edge probability to each block. The only way for the model to increase the edge probability among the nodes in the community overlap is to regard overlaps as separate communities with higher edge density than the individual non-overlapping parts of communities. For example, Figure 4.6 illustrates the adjacency matrix of the network from Figure 4.2a and the block structure as discovered by the stochastic block model. Instead of two overlapping communities, three communities are discovered.

The degree-corrected stochastic block model \cite{76} relaxes the assumption that nodes in the same community have similar degrees. By assigning a high degree to nodes in community overlaps, it might be possible to extend the model to increase the edge probability between overlapping nodes without treating the overlap as a separate community. However, the present version of the model assumes a non-overlapping community structure, and thus the block model cannot tell that the overlapping nodes belong to multiple communities.

Last, the Mixed Membership Stochastic Block Model can discover overlapping communities. However, the model cannot express dense community overlaps. In fact, based on the input matrix from Figure 4.2a, the model will identify three blocks as illustrated in Figure 4.6. The reason for this is that the edge probability between two nodes that belong to communities $I$ and $J$ is the weighted average of $P(A, A)$, $P(A, B)$, and $P(B, B)$, where $P(X, Y)$ is an edge probability between a node in community $X$ and a node in community $Y$. This means that the edge probability between the two nodes that share multiple communities is smaller than the maximum of $P(A, A)$ and $P(B, B)$ (because of the weighted summation). Therefore, the edge probability between overlapping nodes cannot be higher than the edge probability between nodes in an individual community.

4.3.4 Other Models of Network Communities

Our findings that communities form dense overlaps present a paradigm for overlapping communities: thinking of overlapping communities as overlapping tiles. In particular, in the same way as thickness of tiles increases in the areas where the tiles overlap, the “thickness” (i.e., density) of edges increases in the areas where two or more communities overlap. To the best of our knowledge our work here is the first to make these observations and build the conceptual understanding. However, we were able to identify a small number of other
community detection methods that should in principle be able to correctly identify densely overlapping communities (as illustrated in Figure 4.2a). We note that none of these methods has been rigorously tested whether it can detect densely overlapping communities.

First method that would detect densely overlapping communities is a statistical model of network communities by Ball et al. [12]. In particular, Ball et al. present an extension of stochastic block model where node community memberships are not modeled by a multinomial distribution but every node $i$ maintains a factor $k_{i,c}$ that models the amount by which node $i$ belongs to community $c$. This way one can think of node membership as described by a non-normalized multinomial distribution which allows for modeling an increased density of the edges in the community overlaps. However, the inference with the model is not particularly scalable due to the model’s generality.

Similarly, Mørup et al. [110] developed a non-parametric Bayesian multiple membership latent feature model for networks where edges of the network are generated by using a “soft-OR” function. And last, Gregory et al. [59] also developed a heuristic method for network community detection for which our analysis shows that it might be able to correctly identify densely overlapping communities.

4.4 Related work

There is a massive body of work on analyzing the overlaps between communities [4, 5, 102, 118]. However, our approach here is fundamentally different from these existing works. Whereas all the existing methods use the communities detected by their own methods to study the community overlaps, we study the overlaps between ground-truth communities that we defined in Chapter 3.

Very recently, researchers studied the characteristics of ground-truth communities (some were mentioned in Section 3.6.) [1] studied the structural properties of ground-truth communities (which they called annotated communities.) [73] studies the evolution of ground-truth communities. However, these approaches focused on each of individual communities, whereas our work focuses on the overlaps between the communities.
4.5 Conclusion

In this chapter, we investigated the structure and overlaps of ground-truth communities in networks. We observed that the overlaps of communities are more densely connected than the non-overlapping parts of communities, which is in contrast to assumptions made by present community detection models and methods. We also showed that the previous community detection methods cannot identify such densely overlapping communities correctly. This observation suggests a need for a new direction for research on community detection; we need to assume that communities form dense overlaps.

In the following chapters, we develop a family of novel community detection method following the new direction. As a first step, in the next chapter, we build a conceptual model of network community structure. Our model reliably captures the overall structure of networks as well as the overlapping nature of network communities. This model forms a basis for community detection methods that we develop in the later chapters.
Chapter 5

Community-Affiliation Graph Model (AGM)

5.1 Introduction

In the previous chapter, we illustrated that most commonly used state-of-the-art community detection methods fail to properly detect communities with dense overlaps, and thus a new approach is needed. In this chapter and the following chapters, we develop a family of novel community detection methods that can detect densely overlapping communities. Our work employs a model-based approach: In this chapter, we present a generative model that provides a natural explanation for why communities form dense overlaps. In the following chapters, we develop a family of novel community detection algorithms based on the model that we describe here.

From our observations in the previous chapter, we ask the following question: What underlying process causes community overlaps to be denser than the communities themselves? This question motivates our work in this chapter: We present a family of probabilistic generative models for graphs that capture the observed phenomena and produce graphs that have realistic community structure. We build on models of affiliation networks \[21,88\] and develop the Community-Affiliation Graph Model (AGM) which reliably reproduces the organization of networks into communities and the overlapping community structure. In the affiliation network, memberships of nodes to communities are modeled with a bipartite
CHAPTER 5. COMMUNITY-AFFILIATION GRAPH MODEL (AGM)

graph, where on the “bottom” we have the nodes of the underlying network, and on the “top” are the nodes representing communities. The edges of this bipartite graph model node-community affiliations. Given affiliation network between nodes and communities, our model generates the network (i.e., it generates the edges between the nodes). The central idea in generating networks based on the affiliation network is that links among nodes stem from one or more common or shared community affiliations [21, 46].

We mathematically analyze AGM to prove that AGM reproduces the emergence of dense community overlaps. We also evaluate the performance of AGM on six different data sets where we have ground-truth communities. Our experimental results establish that AGM reliably captures node community memberships and generates realistic community overlaps.

Our AGM has implications in several contexts:

- **Design of new community detection methods:** AGM is our first step for developing a novel community detection method. In Chapter [4] we showed that real communities form dense community overlaps and that the current community detection methods cannot detect such densely overlapping communities. Our AGM is a probabilistic model that can generate networks with dense community overlaps. Based on AGM, we develop a novel community detection methods that can detect overlapping communities with dense overlaps (Chapter [6]).

- **Synthetic benchmarks:** Our model can be used to generate synthetic benchmark datasets for evaluation and analysis of network community detection methods. Synthetic graphs are important for “what if” scenarios, for extrapolations, and for simulations, where real graphs are very difficult to collect.

- **Anomaly detection:** In many network settings, “normal” communities will produce overlaps that obey our observations. If we detect communities producing structures that deviate significantly from this, we can flag them as abnormalities; this ability can potentially help with the detection of fraud and spam.
5.1.1 Outline of Chapter

Here is the outline of the chapter:

- We present a generative model for networks and communities. Our model can generate networks with dense community overlaps in a very intuitive way. Also, our model can naturally express a variety of community structures such as non-overlapping, overlapping, and hierarchically nested structures. (Section 5.2)

- We empirically show that our model can generate realistic looking network and communities. In our experiments with six different data sets, our model generates more realistic networks than the state-of-the-art generative model for networks. (Section 5.3)

5.2 Community-Affiliation Graph Model

In this section, we present a simple, conceptual model of behavior, which naturally leads to the phenomena that we have observed.

We build on Breiger’s foundational work [21] that recognized that communities in networks arise as a result of shared group affiliations [21][136]. Furthermore, we build on the
focus theory of social ties \cite{46} which explained tie formation in social networks as a result of shared social foci between the individuals. Building on these sociological theories, we present the Community-Affiliation Graph Model (AGM), a family of simple probabilistic generative models for graphs that capture the phenomena observed in Chapter 4 and reliably reproduce the organization of networks into communities and the overlapping community structure.

The main phenomenon that we aim to capture is the increasing probability of linking as nodes are more and more common communities. For example, consider a pair of people that are members of several different interest based communities. We want to ensure that, by having more communities (\textit{i.e.}, interests) in common, this pair of people are more likely to meet and connect. From this example, we note two ingredients for our model: 1) a way to capture memberships of nodes to communities, and 2) a mechanism that gives nodes that share several communities multiple chances to create links to each other.

For the first ingredient, we use a bipartite affiliation network that links nodes of the social network to communities that they belong to. For the second ingredient, we assign a single probability to each community that specifies the probability that nodes belonging to the community are connected to each other. In this way, the more communities a pair of nodes shares, the higher is the probability of linking.

Figure 5.1a illustrates the essence of our model. We start with a bipartite graph where the nodes at the bottom represent the nodes of the network and the nodes on the top represent communities. The edges between nodes of the network and the communities indicate community memberships. We denote the bipartite affiliation network as $B(V, C, M)$, where $V$ denotes the set of nodes of the original network $G$, $C$ is a set of communities, and there is an edge $(u, c) \in M$ from node $u \in V$ to community $c \in C$ if node $u$ belongs to community $c$.

Now, given the affiliation network $B(V, C, M)$, we want to generate the underlying network $G(V, E)$. To achieve this we need to specify the process which generates the edges $E$ of $G$ given the affiliation network $B$. We consider a simple parametrization where we assign a separate parameter $p_c$ to each community $c \in C$. Parameter $p_c$ models the probability of an edge between two members of community $c$. In other words, we simply generate an edge between a pair of nodes that belong to community $c$ with probability $p_c$. If
nodes do not share any communities we assume they link with a small probability $\varepsilon$. Each community $c$ creates edges independently. However, if the two nodes have already been connected via some other common community membership, then the duplicate edge is not included in the graph $G(V, E)$.

**Definition 1.** Let $B(V, C, M)$ be a bipartite graph where $V$ is a set of nodes, $C$ is a set of communities, and an edge $(u, c) \in M$ connects node $u \in V$ to community $c \in C$ if $u$ belongs to community $c$ in the network $G$. Also, let $\{p_c\}$ be a set of probabilities for all $c \in C$. Given the affiliation network $B(V, C, M)$ and $\{p_c\}$, Community-Affiliation Graph Model generates a graph $G(V, E)$ where the node set $V$ and the edge set $E$ are defined as follows. For each pair of nodes $u, v \in V$, AGM creates edge $(u, v) \in E$ with probability $p(u, v)$:

$$p(u, v) = \begin{cases} 1 - \prod_{k \in C_{uv}} (1 - p_k), & \text{if } C_{uv} \neq \emptyset \\ \varepsilon, & \text{otherwise} \end{cases}$$

(5.1)

where $C_{uv} \subset C$ is a set of communities that $u$ and $v$ share ($C_{uv} = \{c|(u, c), (v, c) \in M\}$), and $\varepsilon > 0$ is a constant.

Note that this simple process already ensures that pairs of nodes that belong to multiple common communities are more likely to link. This is due to the fact that nodes that share multiple community memberships receive multiple chances to create a link. For example, pairs of nodes in the overlap of communities $A$ and $B$ (but not to $C$) in Figure 5.1a get two chances to create an edge. They can be connected first with probability $p_A$ (due to their membership in community $A$), and second with probability $p_B$ (due to membership in $B$).

While pairs of nodes residing in the non-overlapping region of $A$ link with probability $p_A$, nodes in the overlap link with probability $1 - (1 - p_A)(1 - p_B) = p_A + p_B - p_A p_B \geq p_A$, which means that overlaps of communities are more densely connected than the non-overlapping parts. We can also prove that AGM insures that the nodes sharing more communities have higher edge probability (the observations in Figure 4.1) for a simple case where communities have a same value of probability $p_c$:

**Proposition 1.** If $p_c = p$ for all communities $c$, the conditional edge probability between two nodes is an increasing function of the number of communities that the both nodes belong to (Observation in Figure 4.1).
Figure 5.1: **Flexibility of AGM.** AGM allows for rich modeling of network communities: (a) non-overlapping, (b) nested, (c) overlapping. In (a) we can assume that nodes in disjoint communities connect with small probability $\varepsilon$ which allows for sparse links between communities $A$ and $B$.

**Proof.** When two nodes belong to $k$ common communities, AGM connects the two nodes with probability $1 - (1 - p)^k$, which is an increasing function of $k$. \hfill \qed

**$\varepsilon$-Community.** As we described in Equation 5.1, AGM does not account for the edges between the nodes that do not share any common communities. To account for this, we simply assume that nodes which have no communities in common link with a small probability $\varepsilon$. In all our experiments we simply set $\varepsilon = 1/|V|$, where $|V|$ is the number of nodes in a given network.

### 5.2.1 Flexibility of AGM

Figure 5.1 illustrates the flexible nature of Community-Affiliation Graph Model that allows for modeling any combination of network community structures: Traditional non-overlapping communities can be modeled by the affiliation graph where each network node links only to a single community node (Figure 5.1a). Hierarchically nested communities can be modeled by the affiliation graph where subsets of network nodes belong to more and more specialized communities (Figure 5.1b). Communities with overlaps can be modeled by the affiliation graph where some nodes belong to multiple communities while other nodes belong to only one community.
5.3 Experimental Evaluations

Having defined AGM, we now proceed to investigate its properties. We perform a set of simulation experiments and compare our model to other models of network community structure. For each network, we generate a synthetic network with AGM, and then compare the properties of synthetic network and the real network. For a comparison, we use the model of network community structure proposed by Lancichinetti et al. [85]. We refer to the model as the LFR. The LFR model is the state-of-the-art model for generating networks with overlapping community structure that can then be used for evaluating community detection methods. Our goal is to understand whether AGM qualitatively reproduces structural properties of real networks and real communities.

5.3.1 Experimental setup

In order to compare real networks to the synthetic networks generated by AGM and LFR, we need to set parameters of both models. Both AGM and the LFR require bipartite affiliation networks, and for simplicity we construct the community affiliation network from the node community membership information. Then, we use maximum likelihood estimation to fit the parameters of the LFR as well as AGM. For AGM, we fit a set of probabilities \( \{p_c\} \). (We discuss the fitting of \( \{p_c\} \) via convex optimization in the following subsection.) The LFR requires the following parameters: the power-law coefficient of the network degree distribution and the fraction of external edges of each node. We estimate these parameters from the real network using maximum likelihood estimation. Note that when computing the community internal degree of a node, LFR penalizes nodes in the overlap: the internal degree is inversely proportional to the number of communities that the node belongs to, which means that LFR also assumes sparse community overlaps. In other words, in LFR the probability of a pair of nodes being connected in practice decreases with the increasing number of shared communities.

We then compare the networks synthesized by the two models to the real networks. We investigate two criteria: structure of community overlaps and structure of the networks themselves. For each of the six datasets we measure the structural properties of real and synthetic networks and examine the performance of the two models.
5.3.2 Estimating $p_c$ via convex optimization

Given a graph $G(V, E)$ and a bipartite community affiliation network $B(V, C, M)$, we aim to find parameters $\{p_c\}$ that maximize the likelihood of observed edges in $G$:

$$L(\{p_c\}) = \prod_{(u,v) \in E} p(u, v) \prod_{(u,v) \notin E} (1 - p(u, v)).$$  \hspace{1cm} (5.2)

By applying Eq. 5.1 we transform the optimization problem to:

$$\arg \max_{\{p_c\}} \sum_{(u,v) \in E} (1 - \prod_{k \in C_{uv}} (1 - p_k)) \sum_{(u,v) \notin E} \sum_{k \in C_{uv}} (1 - p_k)$$

with constraints $0 \leq p_c \leq 1$.

This optimization is nontrivial to solve. The objective function is non-convex as it involves a product over the variables $p_k$. Now, we show that it can be converted to a convex optimization problem.

We maximize the logarithm of the likelihood and perform a change of variables $1 - p_k = e^{-x_k}$:

$$\arg \max_{\{x_c\}} \sum_{(u,v) \in E} \log(1 - e^{-\sum_{k \in C_{uv}} x_k}) - \sum_{(u,v) \notin E} \sum_{k \in C_{uv}} x_k,$$

and constraints $0 \leq p_c \leq 1$ become $x_c \geq 0$. This transformed problem is a convex function of $\{x_c\}$ and thus the globally optimal values of $\{x_c\}$ can be efficiently found. Then, by the change of variables, we find the values of $\{p_c\}$.

By extending this estimation procedure, we can use AGM for community detection, i.e., we can also search for the optimal community affiliation $B(V, C, M)$ as well as $\{p_c\}$ so that the probability of $G$ is maximized. In the following chapters, we develop AGM based community detection methods where we find $B(V, C, M)$ as well as $\{p_c\}$ via maximum likelihood estimation. Our methods can successfully detect ground-truth communities on a range of networks \cite{152, 156}. 

5.3.3 Evaluation: Edge Probability as a Function of Shared Community Memberships

In Chapter 4, we observed that community overlaps have higher edge density than individual communities. We examine how well the two models mimic these patterns of community overlaps.

First, we found that the edge probability between a pair of nodes is an increasing function of the number of communities that the nodes share. Figure 5.2 plots the edge probability as a function of the number of common communities between a pair of nodes for each real network and compares it to the two models. Notice that AGM successfully reproduces the edge probability, while LFR fails to model the fact that nodes that share more communities are more likely to be connected. Notice that AGM is able to capture a wide range of behaviors — from diminishing-returns (Fig. 5.2a), S-shape (Fig. 5.2e) to a slowly rising pattern (Fig. 5.2f).

We now quantify the overall performance of AGM and the LFR. For each network,
Network  |  Deg  |  CCF  |  Hop  |  TP  |  EigVal  |  EigVec
---|---|---|---|---|---|---
LiveJournal  |  1.09  |  2.72  |  -0.03  |  1.72  |  2.52  |  -0.29
Friendster  |  0.35  |  2.37  |  0.04  |  1.61  |  1.74  |  0.24
Orkut  |  1.06  |  1.52  |  0.08  |  3.39  |  1.12  |  0.51
YouTube  |  -0.55  |  0.93  |  0.35  |  -1.99  |  -0.26  |  -2.01
DBLP  |  -1.10  |  2.31  |  0.02  |  1.07  |  0.59  |  -0.74
Amazon  |  -0.81  |  2.48  |  0.12  |  -1.29  |  -0.53  |  -0.24
Average  |  0.01  |  2.05  |  0.10  |  0.75  |  0.86  |  -0.42

Table 5.1: **Relative difference in the KS-statistic of AGM and LFR for network properties.** Positive values mean that AGM outperforms LFR.

we measure the quality of fit between that property in the synthetic AGM and LFR networks and the real data. We apply the Kolmogorov-Smirnov (KS) statistic, which is a non-parametric way of quantifying the distance between two distribution functions. Given two distribution functions \( f(x), g(x) \), *i.e.*, plots of the same structural property, the KS-statistic computes the maximum difference between the cumulative area under the two curves, \( KS(f, g) = \sup_x | \int_x^\infty f(t)dt - \int_x^\infty g(t)dt | \). We compute the KS-statistic between the AGM curve (or the LFR curve) and the true curve. We then measure the relative improvement in the KS-statistic between the two models (the difference between the two models normalized by the larger value of the two). AGM outperforms LFR by 73% in LiveJournal, 84% in Friendster, 95% in Orkut, and 66% in DBLP, and LFR outperforms AGM by 7% in YouTube and 79% in Amazon. Overall, AGM outperforms LFR by 39%.

### 5.3.4 Evaluation: Properties of the network

Next, we study whether AGM is able to generate overall realistic networks. We examine how well the global structural properties of the synthetic networks match the properties of the real network. For each of the networks synthesized by AGM and LFR, we quantify the degree of agreement between the real and synthetic network by computing the KS-statistic on the following network properties:

- **Degree distribution (Deg):** histogram of the number of edges of a node. Networks tend to have power-law degrees [43].
• **Clustering coefficient (CCF):** distribution of clustering coefficient of nodes [148].

• **Hop plot (Hop):** the number of reachable pairs of nodes in less than $x$ hops [94].

• **Triad participation (TP):** the number of triangles that a node participates in [140].

• **Eigenvalues (EigVal):** distribution of eigenvalues of the adjacency matrix [26, 44].

• **Eigenvector (EigVec):** distribution of components in the eigenvector associated with the largest eigenvalue. It has been also known to be heavy tailed [26, 28].

We first report KS-statistics, our evaluation metric. Table 5.1 shows the relative improvements in KS-statistics of AGM over LFR. The AGM network follows very closely the patterns of the real network for most properties, like the degree distribution, triad participation, and eigenvalues. The only exception where LFR outperforms AGM is the Eigenvector. The AGM exhibits 1% better fit (KS-statistic) than LFR for Degree distribution, 205% for Clustering coefficient, 10% for Hop distribution, 75% for Triad participation, and 86% for Eigenvalues. Only for the Eigenvector property, the LFR exhibits a 42% better value.

Overall, these results demonstrate that AGM is not only able to reliably capture the structure of network communities and community overlaps but that it also accurately generates the underlying networks.

We also show results in Figures 5.3, 5.4 and 5.5. These figures plot the distributions of each of the network properties of synthetic networks and the real networks. We observe that synthetic networks generated by AGM closely follow the statistics of the real networks in most properties.

Figure 5.3 is particularly surprising as it shows that AGM generates both heavy tailed degree distribution and high clustering coefficient. As described above, AGM generates the edges inside communities independently. If every node belongs to only one community (no community overlap), this process would generate homogeneous degree distribution and low clustering coefficient. However, since membership distribution is heavy-tailed (Figure 3.2), there exist a considerable number of high-membership nodes. These high-membership nodes will also have high network degree as they get multiple chances to create edges. This in turn results in a heavy-tailed degree distribution [88].
High clustering coefficient can be explained by that the edges generated by AGM will locally cluster in dense overlaps. In AGM, nodes in the same overlap (nodes who belong to multiple communities together) would have high probability of linking each other, and these nodes will have high clustering coefficient. Interestingly, the connection between community structure and high clustering coefficient that we observe here is supported by recent theoretical advances [54, 133].

5.4 Related work on Models for Overlapping Communities

Our work seeks to develop a generative model for network communities. In terms of probabilistic models for graph generation there has also been considerable work done. The discovery of degree power-laws and other properties of static and dynamic graphs led to the development of random graph models that exhibited such properties [82, 88, 95, 148]. See [26, 108] for surveys of this area. The main difference to our work here is that our goals are more ambitious as we aim to accurately model both the overall network structure as well as the node community memberships and the overlaps of communities.

Our Community-Affiliation Graph Model (AGM), which produces realistic graphs as well as community overlaps, is an example of a bipartite affiliation network model [21, 88, 133, 145, 163]. Affiliation networks have been extensively studied in sociology [21] as a metaphor of classical social theory concerning the intersection of persons with groups, where it has been recognized that communities arise due to shared group affiliations [21, 136]. In affiliation network models, nodes of the social network are affiliated with communities that they belong to and the links of the underlying social network are derived based on the community affiliation network.

The most related to our model is the work of Lattanzi and Sivakumar [88] who studied the macroscopic evolution of networks and proposed an affiliation network model for modeling evolution of social networks. They proved that networks arising from the model exhibit power-law degree distributions, densification power law and shrinking diameter. Our focus here is different as we aim to accurately model static network community structure.
To achieve this, we extend the model of [88] in order to allow for modeling non-overlapping, hierarchically nested, as well as overlapping community structures. This way AGM acquires the necessary flexibility to accurately model the community structure of real-world networks. And last, the model in [88] cannot be fit to the data and thus cannot be used for community detection. On the other hand, we can in fact perform community detection by fitting AGM to a given underlying network. (Chapter 6)

AGM is also related to the models for generating synthetic benchmarks for community detection [75, 85]. Similar to AGM, these models, given community affiliations, generate synthetic networks, in which we can evaluate the accuracy of community detection methods. One of the most notable models is Lancichinetti et al. [85], which is commonly referred to as the LFR model. LFR can generate networks with a power-law degree distribution and overlapping community structure. However, LFR assumes that the community overlaps are not denser than individual communities, whereas AGM assumes that community overlaps are denser. Our experiments in Section 5.3 show that, due to this difference, AGM generates more realistic networks than LFR.

5.5 Conclusion

In this chapter, we presented the Community-Affiliation Graph Model (AGM), a conceptual model of network community structure. We showed that AGM reliably captures how communities form dense overlaps. Our AGM can express a variety of complex community structures in a very natural way and can provide a realistic benchmark networks for development and evaluation of community detection methods.

As we briefly mentioned in Section 5.4, AGM has a unique strength compared to other affiliation network models: AGM allows us to fit the model. In the next chapter, we exploit this strength of AGM to take a natural step forward; we design novel community detection methods where we fit AGM to a given network to find the most likely community affiliations. Since AGM assumes dense community overlaps, our method can detect densely overlapping communities.
Figure 5.3: Model comparison: Network properties. (Degree distribution, Clustering Coefficient distribution)
CHAPTER 5. COMMUNITY-AFFILIATION GRAPH MODEL (AGM)

Figure 5.4: Model comparison: Network properties. (Hop distribution, Triad participation)
Figure 5.5: **Model comparison**: Network properties.(Eigenvectors, Eigenvalues)
Chapter 6

Community Detection with AGM

6.1 Introduction

In the previous chapter, we presented the Community-Affiliation Graph Model (AGM) which generates realistic looking networks given community affiliations of nodes. We showed that our AGM provides an intuitive explanation for our observations that community overlaps have high edge density. We also showed that the model parameters of AGM allow us to represent various kinds of community structures such as non-overlapping, overlapping, or hierarchically nested structures.

Based on AGM, in this chapter we develop a community detection method that successfully detects overlapping, non-overlapping, as well as nested communities in networks. We achieve this by fitting AGM (i.e., discovering the node-community affiliation graph) to an unlabeled undirected network. Using the Markov Chain Monte Carlo method and convex optimization, we develop a fitting algorithm for identifying node community affiliations. We also present a method that automatically determines the number of communities in a given network.

Figure 6.1 illustrates our approach. We asked a Facebook user to manually organize his Facebook friends into communities. The user classified his friends into four communities corresponding to his high-school, his internship, and two communities of university friends. Each friend is a member of one or more communities. Independently, our method detected communities in his Facebook network. Our method detected four communities, which is
Figure 6.1: **Facebook communities detected by AGM.** An example demonstrating the performance of AGM on a Facebook friendship network of a particular user. Communities detected by AGM are denoted by filled regions. Each detected community almost perfectly corresponds to the ground-truth community as labeled by the user: high-school friends, colleagues at the workplace (internship), and university friends with whom the user plays basketball and squash. Nodes are colored black if AGM community membership corresponds to the same community membership as determined by the user and gray otherwise. AGM assigns 94% of nodes to correct communities. Nodes in the overlap of communities have higher density of edges.

the same as the number identified by the user. Moreover, our method correctly determined community overlaps and community memberships for 94% of user’s friends.

We evaluate the performance of AGM on various social, collaboration, information and biological networks. Our experimental results reveal that AGM discovers overlapping as well as non-overlapping community structure more accurately than present state-of-the-art methods [4, 118, 5, 129]. AGM also outperforms all the baselines in the benchmark presented in Ahn et al. [4] which measures the quality and coverage of the detected community structure on biological, information, and word association network. The success of our approach relies on the flexibility of the AGM, which allows for modeling overlapping, non-overlapping as well as hierarchically nested communities in networks.

In addition to developing a novel and accurate community detection method, our work in this chapter presents a more rigorous way of evaluating community detection methods;
that is, we use the ground-truth communities that we described in Chapter 3 for our evaluation. Traditionally, the performance of community detection methods is evaluated by manual inspection yet such manual inspection can be subjective and is limited to small-sized networks. On the contrary, evaluation based on ground-truth communities allows for quantitative and large-scale evaluation of network community detection methods [153, 155].

In summary, our work has two main contributions:

- Based on AGM, we develop a novel community detection method that detects overlapping, non-overlapping, as well as nested communities in networks.

- We use social and information networks with explicit community memberships to evaluate network community detection methods. We believe that our work will bring more rigor to the standard for the evaluation of community detection methods.

6.1.1 Outline of Chapter

Here is the outline of the chapter:

- We present a novel community detection method where we find the best community affiliations for a given network by fitting AGM to the given network. Our fitting procedure is based on Markov-Chain Monte Carlo methods and convex optimization. We also present a separate method for estimating the number of communities in the network. (Section 6.2)

- We evaluate the performance of AGM by conducting three experiments. In the first experiment, we evaluate the accuracy of community detection using the ground-truth communities that we defined in Chapter 3. AGM discovers ground-truth communities more accurately than the state-of-the-art community detection methods. (Section 6.3)

- In the second experiment, we evaluate the quality of detected communities in protein-protein interaction networks. Our results show that AGM discovers more meaningful protein communities than other methods. (Section 6.4)
In the third experiment, we follow the benchmark in Ahn et al. [4] which evaluated the quality of detected communities based on node attributes. AGM outperforms other baselines. (Section 6.5)

### 6.2 Fitting AGM

In this section, we explain how we can detect communities based on AGM. Given network $G(V, E)$, we aim to detect communities by fitting AGM (i.e., finding affiliation graph $B$ and parameters $\{p_c\}$) to the underlying network $G$ by maximizing the likelihood $L(B, \{p_c\}) = P(G|B, \{p_c\})$:

$$
\arg \max_{B, \{p_c\}} L(B, \{p_c\}) = \prod_{(u,v) \in E} p(u, v) \prod_{(u,v) \notin E} (1 - p(u, v))
$$

(6.1)

To maximize the likelihood $L$ we employ coordinate ascent strategy where we update $\{p_c\}$ fixing $B$ and then we update $B$ with $\{p_c\}$ fixed.

For now we assume the number of communities $K$ ($K = |C|$) is known. We later show how to automatically determine $K$. We start the process by generating a random affiliation graph $B$ on $|V|$ network nodes and $K$ community nodes. We then proceed by updating $\{p_c\}$.

**Updating $\{p_c\}$.** With $B$ fixed, we aim to find $\{p_c\}$ by solving the following optimization problem:

$$
\arg \max_{\{p_c\}} \prod_{(u,v) \in E} (1 - \prod_{k \in C_{uv}} (1 - p_k)) \prod_{(u,v) \notin E} (1 - \prod_{k \in C_{uv}} (1 - p_k))
$$

(6.2)

with the constraints $0 \leq p_c \leq 1$. In Section 5.3.2 we showed that we can solve this problem using convex optimization.

**Updating $B$.** Now, given fixed $\{p_c\}$ we aim to update $B$, while maximizing the likelihood. To this end we use the Metropolis-Hastings [31, 116] algorithm where we update $B$ using small local modifications to it. Given the current community affiliation graph $B(V, C, M)$, we consider three kinds of local modifications to generate a new community affiliation graph $B'(V, C, M')$. Because $V$ and $C$ remain the same, we just need to update $M$ to $M'$:
Figure 6.2: **Local modifications for updating community affiliation graph** $B$. Given the current community affiliation graph on the left, we consider three local modifications. **LEAVE** considers a node quitting a community. **JOIN** considers a node joining a new community. **SWITCH** causes a node to replace one of its current communities with a new one.

- **LEAVE**: Choose a node-community pair $(u,c) \in M$ uniformly at random and let $M' = M \setminus \{(u,c)\}$.

- **JOIN**: Choose a node-community pair $(u,c) \notin M$ uniformly at random and let $M' = M \cup \{(u,c)\}$.

- **SWITCH**: Choose a node-community pair $(u,c1) \in M$, $(u,c2) \notin M$ at uniformly random and let $M' = (M \setminus \{(u,c1)\}) \cup \{(u,c2)\}$.

Once we have generated new community affiliation $B'$, we accept $B'$ with the Metropolis-Hastings rule. If $B'$ achieves higher likelihood than $B$, we accept $B'$. Otherwise, we accept $B'$ with probability $L(B',\{p_c\})/L(B,\{p_c\})$.

We experimented with many synthetic and real-world networks, and found that the Markov chain of fitting AGM exhibits relatively quick convergence as the likelihood does not increase after roughly $O(|V|^2)$ steps. Although this is not a rigorous performance guarantee, results show that the fitting method works quite well in practice. The algorithm can fit AGM to the networks with up to a few thousand nodes within several hours. Figure 6.3b shows how the log-likelihood converges for the Facebook ego-network in Figure 6.1.
Figure 6.3: **Convergence of fitting AGM to a given network.** The likelihood of AGM versus the number of iterations of Metropolis-Hastings measured on a synthetic network generated by AGM (a) and a Facebook ego-network (b) in Figure 6.1. In both cases AGM reaches converges after around 10,000 iterations and accurately discovers communities.

**Identifiability of AGM.** Our first test of AGM community detection method is to examine whether our fitting method can recover the model parameters given a synthetic network that was generated by AGM model itself. Assume a synthetic network $G^*$ that is generated by AGM using some input parameters $B^*$, $\{p_c^*\}$. Now our goal is to recover $B^*$, $\{p_c^*\}$ based only on the network $G^*$.

For example, we generated a network with the overlapping communities such that 100 nodes belong to community A, 100 nodes belong to community B, and 50 nodes belong to the both communities. We set $p_{A^*} = p_{B^*} = 0.3$ and generate the network $G^*$. Now given $G^*$ we identify communities by fitting AGM to the synthetic network $G$. AGM can discover communities A and B with perfect accuracy and estimate $p_A, p_B$ very closely ($p_A = 0.30, p_B = 0.29$). Figure 6.3a shows the likelihood as a function of the number of iterations. After 3,000 iterations, the likelihood reaches a plateau and converges.

We also considered a more general cases where we generated some random $B^*$ and randomly assigned parameters $\{p_c^*\}$. In nearly all cases our algorithm was successfully able to recover parameters $B^*$ and $\{p_c^*\}$ given only the synthetic network $G^*$. 
6.2.1 Automatically Finding the Number of Communities

To initialize $B(V, C, M)$, we need to set the number of communities $K = |C|$, which in practice is not known in advance. To resolve this, we develop a method to automatically estimate the number of communities. Our approach is based on statistical regularization techniques [139].

Our strategy begins with a candidate set of a large number of communities that might exist in a given network. We generate this candidate set by fitting AGM using a very large number of communities $K$. (In practice, we let $K = |V|/5$.) Then, we keep removing redundant communities using $l_1$-regularization until we observe a drop in the log-likelihood. We observe a threshold-like behavior of the log-likelihood as a function of the regularization parameter. We choose minimum number $k^*$ of communities before the log-likelihood drops. This way we find the minimum number $k^*$ of communities that is still sufficient to model the structure of a given network.

More precisely, we proceed as follows. First, we use a very large number of communities ($|C_0| = O(|V|)$) to fit AGM on the given network $G(V, E)$ and obtain the resulting bipartite community affiliation graph $B_0(V, C_0, M_0)$.

Note that not every community found in $B_0$ is important. This $B_0$ is a set of candidate communities from which we find a set of communities that are very likely to exist. Key intuition is that we can ignore communities if their corresponding $p_c$ is 0. Therefore, we aim to reduce the number of communities in $B_0$ by forcing more and more parameters $p_c$ to zero.

We apply $l_1$-regularization with parameter $\lambda$ to a problem of fitting $\{p_c\}$ to $B_0$. At each value of $\lambda$, we solve the following problem:

$$\{\hat{p}_c(\lambda)\} = \arg\max_{\{p_c\}} P(G|B_0, \{p_c\}) - \lambda \sum_c |p_c|$$ (6.3)

Solution $\{\hat{p}_c(\lambda)\}$ is a sparse vector with only few individual $\hat{p}_c \neq 0$. Non-zero $\hat{p}_c$ act as indicators of active communities in $B_0$. We construct $B(\lambda)(V, C(\lambda), M(\lambda))$ by taking the communities in $B_0$ with non-zero $p_c(\lambda)$. Each such $B(\lambda)$ represents a set of active communities at the regularization intensity $\lambda$.

Now our goal is to find the value of regularization parameter $\lambda$ such that we discover the
Figure 6.4: **Automatically determining the number of communities in a given network.** We plot various quantities as a function of regularization intensity $\lambda$ using two $Y$-axes. $K^*$: The true number of communities (using the left axis). $K(\lambda)$: The number of communities we estimate under regularization intensity $\lambda$ (using the left axis). $L'(B(\lambda))$: The normalized likelihood of $B(\lambda)$ (using right axis). $\sigma(\lambda)$: The sigmoid function fit to normalized $L(B(\lambda))$ (using right axis). Pink dotted vertical line: $\lambda^*$ at which $\sigma(\lambda^*)$ falls below 0.25 (using the right axis). Red horizontal line, the estimated (as well as the true) number of communities.

true number of communities in the network. We achieve this by measuring how well $B(\lambda)$ can represent $G(V, E)$ by measuring its likelihood $L(B(\lambda)) = \max_{p_c} P(G|B(\lambda), \{p_c\})$. Likelihood $L(B(\lambda))$ tells us how well $G(V, E)$ can be explained when we use only $C(\lambda)$ communities. For example, Figure 6.4 plots $L(B(\lambda))$ and $K(\lambda)$ measured on a network that has $K^* = 2$ true communities.

Notice that whereas $K(\lambda)$ is an almost strictly decreasing function of $\lambda$, $L(B(\lambda))$ seems to be a step function which is flat until $\lambda$ reaches some threshold and then suddenly drops. This suggests that no more than $K(\lambda)$ communities are needed to explain $G(V, E)$ when $\lambda$ is relatively small. In other words, $K(\lambda)$ with high $L(B(\lambda))$ gives us a upper bound for the number of communities that exist on the network. The tight upper bound happens at the point at which $L(B(\lambda))$ suddenly drops, and we report such $K(\lambda)$ measured at the quick drop of $L(B(\lambda))$ as our estimate for the number of communities.

Since we cannot examine all possible values of $\lambda$, detecting the exact value of $\lambda$ at which $L(B(\lambda))$ suddenly drops is a challenging task. To find such $\lambda$, we resort to the following
heuristics. We approximate $L(B(\lambda))$ by the sigmoid function $\sigma(\lambda) = \frac{1}{1 + e^{\alpha \lambda + \beta}}$. We first normalize $L(B(\lambda))$ into $L'(B(\lambda))$ so that the maximum over $\lambda$ is 1 and the minimum is 0, and then we fit the sigmoid function $\sigma(\lambda)$ to $L'(B(\lambda))$ by finding the optimal parameters $\alpha$ and $\beta$ [25]. Then we compute $\lambda^*$ such that $\sigma(\lambda^*) = \delta$ for some constant $\delta \ll 1$, and $K(\lambda^*)$ is our estimate for the number of communities. We experimented with various values of $\delta$ and found that setting $\delta = 0.25$ is a reasonable choice.

For example, in Figure 6.4 our strategy estimates the true number of communities correctly. In the experiments on the real-world networks, our strategy succeeds to estimate the true number of communities more accurately than other methods (Section 6.3).

The run time of this method mostly depends on fitting $B_0$ because solving Problem 6.3 can be done efficiently due to its convexity. Therefore the overhead of automatically finding the number of communities bring little computational overhead in practice. More importantly, with this method we can use AGM without any parameters using the following two-step strategy. When a network is given, we first estimate the number of communities $\hat{K}$ in the network, and then fit AGM using our estimate $\hat{K}$.
6.2.2 AGM Does Not Suffer From the “Resolution” Limit

Many community detection methods suffer from the “resolution limit” [17, 51, 57]. In particular, Fortunato et al. [51] showed that Modularity has a resolution limit in a sense that Modularity cannot detect communities if they are too small.

A ring of cliques (illustrated in Figure 6.5) is an example of a graph where the resolution limit can be reliably studied. The ring of cliques consists of \( n \) cliques \( K_m \), where each \( K_m \) is a complete graph on \( m \) nodes. The cliques are then connected into a ring by adding a single edge between two consecutive cliques. On such a graph we expect a given community detection method to \( n \) find communities—each \( K_m \) is a separate community. However, [51] proved that the Modularity-based methods fail to discover each \( K_m \) as a separate community if \( n \) is larger than the square root of the number of edges in the network, i.e., Modularity-based methods fail for a network consisting of many small modules.

We run AGM for the same values of \( n, m \) used in [51] \( (n = 30, m = 5) \), and find that AGM correctly identifies both the number of communities \( n \) as well as detects each \( K_m \) as a separate community. We also experimented with many other values for \( n \) and \( m \) (e.g., \( n = 50, m = 10, n = 100, m = 10 \)) and observed that AGM perfectly identifies each \( K_m \) as a community. From these experiments we conclude that AGM does not suffer from the resolution limit.

6.3 Experiments: Networks with Ground-truth Communities

In this section we evaluate the performance of AGM and compare it to the state-of-the-art community detection methods on a range of networks from a number of different domains and research areas. We perform experiments on the six networks described in Section 3 (Table 3.1) for which we have explicitly labeled ground-truth communities. Availability of ground-truth communities allows us to quantify the “accuracy” of community detection methods by comparing the level of correspondence between the detected and the ground-truth communities.
Figure 6.6: Sampling subnetworks with community overlaps. On the left is a part of a full network $G(V, E)$ and on the right is the subnetwork that we sample. We randomly pick a node (the red node on the left) and then construct a subnetwork consisting of the communities that the red node belongs to.

### 6.3.1 Experimental Setup

We evaluate community detection methods on their ability to correctly identify overlapping communities. With this purpose in mind, running community detection algorithms on a whole network is not an effective way for two reasons. First, for some nodes we have no ground-truth community labels. And more importantly, none of the community detection algorithms that we consider is scalable to the size of networks we consider here.

To resolve this issue, we proceed by finding subnetworks with highly overlapping community structure from a given network $G(V, E)$. To obtain one such subnetwork we pick a random node $u \in V$ that belongs to more than one ground-truth community and then take the induced subgraph of $G$ consisting of all the nodes that share at least one ground-truth community membership with $u$. Figure 6.6 illustrates how a subnetwork (right) is created from $G(V, E)$ (left) when the red node $u$ is chosen. We identify all the member nodes of the communities that the red node belongs to and then construct the induced subgraph on the right. This way we obtain subgraphs that are of reasonable size and contain fully labeled overlapping communities.

We sample 500 subnetworks for each of the six networks from Table 3.1. We control the sampled networks to have similar number of ground-truth communities across the data sets. In particular, we sample 100 networks for each of two, three, four, and five communities on each network. And the last 100 sampled networks have more than five communities.

We also considered many alternative ways of obtaining small enough subnetworks so
that community detection algorithms could be run. For example, we considered a strategy
where given a network $G(V, E)$, we pick a random node $u \in V$ and find a set of nodes
$V_u$ that are less than 2-hop away from $u$. We then construct the induced subgraph of $V_u$
and take communities that have more than 50% of their members in this induced subgraph.
We also considered the approach where we created a random set of “connected” commu-
nities that either share an edge or a node. In all these cases the results we obtained were
qualitatively similar.

### 6.3.2 Methods for Comparison

We compare AGM to several state-of-the-art community detection methods. We choose
the three most prominent community detection methods: Link Clustering (LC) [4], Clique
Percolation Method (CPM) [118], and the Mixed-Membership Stochastic Block model
(MMSB) [5].

These methods have a number of parameters that need to be set. For CPM, we have to
choose the clique size $k$ yet there is no well-defined criteria for setting $k$. We use $k = 5$
since the number of communities discovered by CPM with $k = 5$ best approximates the
true number of communities. MMSB also requires the number of communities $K$ as an
input parameter. We use the Bayes Information Criterion as described in [5] to choose $K$.
MMSB outputs a stochastic vector for each node representing partial memberships to each
of the $K$ communities. To generate “hard” memberships we assign a node to a community
if the corresponding stochastic membership is non-zero. For CPM and LC we used the
implementation in the Stanford Network Analysis Platform [1] while for MMSB we used
the publicly-available ‘LDA’ R package. We note that we also considered Infomap [129],
which is the-state-of-the-art non-overlapping community detection method. We omit the
results as the performance of the method was not competitive.

### 6.3.3 Evaluation Metrics

To quantify the performance, we measure the level of agreement between the detected and
the ground-truth communities. Given a network $G(V, E)$, we consider a set of ground truth

---

communities $C^*$ and a set of detected communities $\hat{C}$ where each ground-truth community $C_i \in C^*$ and each detected community $\hat{C}_i \in \hat{C}$ is defined by a set of its member nodes. To compare $\hat{C}$ and $C^*$, we use four performance metrics:

- **Average F1 score** \[101\] is the average of the F1 score of the best-matching ground-truth community for each detected community, and the F1 score of the best-matching detected community for each ground-truth community. In particular, we compute $F_g(C_i) = \max_j F1(C_i, \hat{C}_j)$ for each ground-truth community $C_i$ and $F_d(\hat{C}_i) = \max_j F1(C_j, \hat{C}_i)$ for each detected community $\hat{C}_i$, where $F1(S_1, S_2)$ is the harmonic mean of precision and recall between two node sets $S_1, S_2$. The average F1 score is $\frac{1}{2}(\bar{F}_g + \bar{F}_d)$ where $\bar{F}_g = \frac{1}{|C^*|} \sum_i F_g(C_i)$ and $\bar{F}_d = \frac{1}{|\hat{C}|} \sum_i F_d(\hat{C}_i)$.

- **Omega Index** \[60\] is the accuracy of estimating the number of communities that each pair of nodes share. For each pair of nodes $u, v \in V$ we define $C_{uv}$ to be the set of ground-truth communities to which both $u$ and $v$ belong and $\hat{C}_{uv}$ to be the set of detected communities to which the both nodes belong. Then the Omega Index is $\frac{1}{|V|^2} \sum_{u,v \in V} 1\{|C_{uv}| = |\hat{C}_{uv}|\}$.

- **Normalized Mutual Information** \[86\] adopts the criterion used in information theory. The Normalized Mutual Information is $1 - \frac{1}{2}(H(C^*|\hat{C}) + H(\hat{C}|C^*))$ where $H(A|B)$ is the extension of entropy when $A, B$ are sets of sets \[86\].

- **Accuracy in the number of communities** is $1 - \frac{|C^*| - |\hat{C}|}{|C^*|}$, which is the relative error in predicting the number of communities.

Note that all performance metrics take values on the interval $[0, 1]$ and higher values correspond to better performance. In all metrics, a score of 1.0 is achieved when the detected communities $\hat{C}$ are exactly the same as the ground-truth communities $C^*$.

Figure 6.7 gives a simple example of two ground-truth communities $C_1, C_2$ and two detected communities $\hat{C}_1, \hat{C}_2$. $\hat{C}_1$ and $\hat{C}_2$ are denoted by shared ellipses. The nodes marked with “X” belong to $C_1$ and the nodes with yellow background belong to $C_2$.

In this example, the detected community $\hat{C}_1$ perfectly corresponds to $C_1$ but the detected community $\hat{C}_2$ only partially corresponds to $C_2$. For this particular case the F1 score is 0.94 as we see $F_g(C_1) = 1.0, F_g(C_2) = 0.89$ and $F_d(\hat{C}_1) = 1.0, F_d(\hat{C}_2) = 0.89$. The Omega
Figure 6.7: **An example of two detected communities** (shaded regions $\hat{C}_1$ and $\hat{C}_2$) and two ground-truth communities $C_1, C_2$ (“X”-marked nodes belong to $C_1$ and yellow nodes belong to $C_2$). In this example detected communities $\hat{C}_1$ and $\hat{C}_2$ achieve good correspondence to the ground-truth communities $C_1, C_2$ (only a single mistake is made).

Index is 0.85 and the Normalized Mutual Information is 0.78. And the accuracy in the number of communities is 1.

### 6.3.4 Results

For each of six networks and the 500 subnetworks (3,000 subnetworks total) we run AGM as well as the four other methods. For each subnetwork and method we compute the four evaluation metrics. We then compute the average value of a given performance metrics for a given method and network. Now, for each metric, we normalize the scores of methods so that the best performing method for each score has the value of 1.0. Finally, we compute the composite performance by summing up the four normalized scores. If a method achieves better value than any other method in all the scores, then the composite performance of the method is 4.0.

Figure 6.8 displays the composite performance of each of the 5 methods over the six networks with ground-truth communities. Overall, we notice that AGM gives superior overall performance on all networks except the Amazon, where it ties with MMSB. Furthermore, AGM detects the highest quality communities for most individual performance metrics in all networks. On average, the composite performance of AGM is 3.40, which is 61% higher than that of Link Clustering (2.10), 50% higher than that of CPM (2.41), 30% higher than
### Measures

<table>
<thead>
<tr>
<th></th>
<th>Normalized Mutual Information</th>
<th>Number of Communities</th>
<th>-index</th>
<th>F1-score</th>
<th>Composite Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGM</td>
<td>0.57</td>
<td>0.42</td>
<td></td>
<td>0.46</td>
<td>3.25</td>
</tr>
<tr>
<td>Link Clustering</td>
<td>0.57</td>
<td>0.42</td>
<td></td>
<td>0.46</td>
<td>3.25</td>
</tr>
<tr>
<td>CPM</td>
<td>0.57</td>
<td>0.42</td>
<td></td>
<td>0.46</td>
<td>3.25</td>
</tr>
<tr>
<td>Infomap</td>
<td>0.57</td>
<td>0.42</td>
<td></td>
<td>0.46</td>
<td>3.25</td>
</tr>
<tr>
<td>MMSB</td>
<td>0.57</td>
<td>0.42</td>
<td></td>
<td>0.46</td>
<td>3.25</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Methods</th>
<th>Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>Link Clustering</td>
</tr>
<tr>
<td>C</td>
<td>Clique Percolation</td>
</tr>
<tr>
<td>I</td>
<td>Infomap</td>
</tr>
<tr>
<td>M</td>
<td>Mixed-Membership Block Model</td>
</tr>
<tr>
<td>A</td>
<td>AGM</td>
</tr>
</tbody>
</table>

### Figure 6.8: The composite performance of the community detection methods on six networks with ground-truth communities. AGM gives the overall best performance.

The composite performance of the community detection methods on six networks with ground-truth communities. AGM gives the overall best performance.

Higher than that of Infomap and 8% higher than that of MMSB (3.25). The absolute average value of the Omega Index of AGM over the 6 networks is 0.46, which is 21% higher than Link Clustering (0.38), 22% higher than CPM (0.37), 5% higher than Infomap (0.44) and 26% higher than MMSB (0.36).

In terms of absolute values of scores, AGM archives an average F1 score of 0.57, an average Omega index of 0.46, a Mutual Information of 0.15 and an accuracy of the number of communities 0.42. We also note that AGM also heavily outperforms CPM with other values of $k$ (e.g., CPM with $k = 3, 4, 6$).

### 6.4 Experiments: Biological Networks

Having found that AGM can discover the community structure on the social and information networks accurately, we now evaluate AGM on biological networks.

#### 6.4.1 Dataset description

We consider the protein-protein interaction (PPI) networks of *Saccharomyces cerevisiae*, which are one of the most complete protein-protein interaction networks available today [161, 81, 4]. The PPI networks are compiled into three different genome-scale networks: yeast two-hybrid (Y2H), affinity purification followed by mass spectrometry (AP/MS), and literature curates (LC). Edges correspond to statistically significant interactions among proteins. In addition to these three networks already mentioned, we use also the union of
the three networks (PPI (All)). Basic statistics of the networks are shown in Table 6.1.

6.4.2 Evaluation metrics

We measure the quality of detected communities by using high quality node meta-data. We use the Gene Ontology terms (GO terms) as meta-data [9]. The GO terms provide the most elaborate annotations for the biological roles of groups of proteins in the protein-protein interaction network for three different types: Biological process, Cellular component, and Molecular function. Moreover, there are statistical tools [14, 20] that find the most relevant GO-term for a group of proteins. We quantify the correspondence between the protein communities detected by a method and the significance of the associated GO terms.

Given a PPI network, we detect communities $\hat{C}$ over the whole network. For each detected community $\hat{C}_i \in \hat{C}$, we find the most statistically relevant GO term and its $p$-value, $p_v(\hat{C}_i)$ using the GO term finder [20]. We then report the average $p$-value over the detected communities: $\bar{p}$ ($\bar{p} = \frac{1}{|\hat{C}|} \sum_i p_v(\hat{C}_i)$). We compute the average $p$-value $\bar{p}$ for the three types of GO terms (biological process, cellular component and molecular function) and use $-\log(\bar{p})$ for the each GO term type as a separate score. We take the negative logarithm of the $p$-value to transform it so that the performance score is a nonnegative increasing function of the quality. Finally, we normalize the scores so that the best method achieves the value of 1.0 as we did in Section 6.3.

6.4.3 Results

Figure 6.9 displays the composite performance for biological networks for each of the five methods (LC, CPM, Infomap, MMSB, and AGM). Note that AGM attains the best composite performance in all four networks by a huge margin. CPM is the second best, Link Clustering is the third, and Mixed Membership Stochastic Block Model scores the worse.

Since the comparison is made on the logarithms of $p$-values, this result suggests that communities detected by AGM are far more statistically relevant than those detected by other methods. For example, the average $p$-value of AGM communities over all the networks is 0.008, which is 13-times better than that of Link Clustering (0.11), 15-times better
### Dataset

<table>
<thead>
<tr>
<th>Network Type</th>
<th>Nodes</th>
<th>Edges</th>
<th>Clustering Coefficient</th>
<th>Shortest Path Length</th>
<th>Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Protein-protein interaction networks</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPI (Y2H) [161]</td>
<td>1,647</td>
<td>2,518</td>
<td>0.10</td>
<td>6.60</td>
<td>3.06</td>
</tr>
<tr>
<td>PPI (AP/MS) [33]</td>
<td>1,004</td>
<td>8,319</td>
<td>0.72</td>
<td>6.51</td>
<td>16.57</td>
</tr>
<tr>
<td>PPI (LC) [125]</td>
<td>1,213</td>
<td>2,556</td>
<td>0.46</td>
<td>8.56</td>
<td>4.21</td>
</tr>
<tr>
<td>PPI (All) [4]</td>
<td>1,647</td>
<td>12,784</td>
<td>0.41</td>
<td>6.24</td>
<td>8.60</td>
</tr>
<tr>
<td><strong>Networks from Ahn et al. [4]</strong> (Section 6.5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metabolic [45]</td>
<td>1,042</td>
<td>8,756</td>
<td>0.74</td>
<td>3.15</td>
<td>16.81</td>
</tr>
<tr>
<td>Philosophers [4]</td>
<td>1,218</td>
<td>5,972</td>
<td>0.30</td>
<td>4.25</td>
<td>9.81</td>
</tr>
<tr>
<td>Word Association [112]</td>
<td>5,018</td>
<td>55,232</td>
<td>0.19</td>
<td>4.04</td>
<td>22.01</td>
</tr>
</tbody>
</table>

Table 6.1: **Network statistics.** *N*: Number of nodes, *E*: Number of edges, ⟨C⟩: Average clustering coefficient [148], ⟨D⟩: Average shortest path length, ⟨k⟩: Average node degree. The four networks in the first block are the protein-protein interaction networks of *Saccharomyces cerevisiae* which are described in Section 6.4. The next three networks in the second block are the networks used in Ahn et al. [4]. We describe these networks in Section 6.5.

than of Clique percolation (0.12), 12-times better than of Infomap (0.096) and also 119-times better than that of MMSB (0.95). We further investigated the poor performance of MMSB on this network and found it is due to the fact that MMSB tends to find very large communities, which in turn leads to very poor *p*-values.

In terms of absolute values of *p*-values AGM performs quite well. For example, in the AP/MS network, AGM achieves the average *p*-value of $1.9 \times 10^{-5}$, which suggest the high significance of the detected communities.

Discovering interactions among proteins still remains an active research area; only ≈ 20% of all protein-protein interactions in yeast have been currently reported [161]. The high-quality protein communities detected by AGM can suggest very plausible candidates that biologists can investigate for undiscovered protein-protein interactions [119, 126].

## 6.5 Experiments: Networks in Ahn et al.

Finally, we also evaluate the performance of AGM under exactly the same conditions as used in the original link Clustering paper by Ahn et al. [4]. Ahn et al. [4] kindly shared with us the exact networks, metadata and the code. Ahn et al. provide objective evaluations
CHAPTER 6. COMMUNITY DETECTION WITH AGM

of community detection methods with data-driven measures. We replicate the experiment in [4] with the same data sets and the same evaluation methodology.

6.5.1 Dataset Description

We consider the seven networks used in [4]. The first four networks are the 4 PPI networks that were described in Section 6.4.

In addition to four PPI networks, we consider three more networks. As the fifth network, we consider the metabolic network of *E. coli* K-12 MG1655 strain (iAF1260), which is regarded as one of the state-of-the-art metabolic network reconstructions [45]. Two metabolites have an edge if they share a cellular reaction. In total, we have five biological networks. Next, we consider the network of famous philosophers constructed based on Wikipedia [4]. If the Wikipedia page of a philosopher has a hyperlink to the Wikipedia page of the other philosopher, then the two philosophers have an edge between them. Finally, we consider the Word association network from the data sets from the University of South Florida and the University of Kansas [42, 112], which observed which words human subjects associate to given words. As the data set provides weighted, directed graph among words, we convert the graph into undirected and unweighted version [4, 118]. Basic statistics of the networks is in Table 6.1. Further details are in [4].

---

2We thank Sune Lehmann for generously providing data.
6.5.2 Evaluation Metrics

We adopt the 4 data-driven measures defined in [4]:

- The community coverage is the fraction of the nodes that belong to at least one detected community.

- The overlap coverage is the average value of the number of communities a node belongs to. If the method detects many communities that share large overlaps, then the overlap coverage will be high.

- The community quality assumes that the similarity of the two nodes $\mu(i, j)$ is available for any pair of nodes $i$ and $j$. Given the similarities, the community quality is the average similarity between all pairs of nodes that share a community, divided by the average similarity between all pairs of nodes [4].

- The overlap quality requires for each node $i$ the information $W(i)$ which is related to the number of true communities that $i$ belongs to. On the protein-protein interaction networks, for example, a protein annotated with many GO terms is expected to belong to many protein communities. On the word association network, words with many definitions are likely to belong to many communities of words. The overlap quality is the mutual information between $W(i)$ and the number of detected communities that $i$ belongs to.

For each network, we apply AGM, Link Clustering, Clique percolation, Infomap and Mixed-membership stochastic block model. For evaluation we use exactly the same meta data and the same parameters as in [4].

6.5.3 Results

Similar to the previous experiments, we compute the composite performance by normalizing the scores the same way as we did in the experiments with ground-truth communities. Figure 6.10 shows the composite performance of the four methods. AGM achieves best composite performance in the 3 networks (PPI (Y2H), PPI (LC) and Philosophers), Link Clustering performs slightly better in the Word association and the metabolic network, and
Figure 6.10: The data-driven benchmark presented in Y.Y. Ahn et al. [4]. Community coverage, Overlap coverage, Community quality, and Overlap quality measure the quality of the communities detected by the algorithms. AGM gives overall best performance.

MMSB is the best in the PPI (Y2H) and PPI (All) networks. On average, AGM achieves a composite performance score of 3.06, outperforming Link Clustering (2.67) by 14%, Clique percolation (1.49) by 104%, Infomap (1.82) by 67% and MMSB (2.84) by 8%. Thus, AGM gives overall best performance on this diverse set of networks and evaluation metrics.

6.6 Conclusion

In this chapter, we developed a novel community detection method that accurately discovers the overlapping community structure of real-world networks. In particular, we presented an efficient algorithm to fit AGM to a given network whose community structure is unknown. Experiments show that AGM outperforms the state-of-the-art community detection methods in accurately discovering network communities as well as the overlaps between communities.

Our work has important implications: First, densely overlapping communities that are detected by AGM help us examine the organization of complex networks in a fundamentally different way. Second, our evaluation methodology with ground-truth communities offers a reliable way for evaluating community detection methods.
As mentioned in Section 6.2, AGM takes several hours for networks with a few thousands of nodes. This range of runtime is comparable to those of the existing community detection methods. To deal with networks with millions of nodes, however, we need a faster method than the method presented in this chapter. In the next chapter, we provide a solution for this problem; we develop an approximate fitting algorithm for AGM which can handle million-node networks within a few hours.
Chapter 7

Fast Algorithm for Fitting AGM

7.1 Introduction

One of the biggest challenges in the current state of community detection is the lack of scalable methods [50]. Almost all overlapping community detection methods [4, 5, 118] (including our method presented in the previous chapter) are computationally constrained to scale to large networks. As a result, while non-overlapping communities in large networks are relatively well understood thanks to fast non-overlapping community detection methods [7, 16, 77], it is still not clear how to identify overlapping communities in very large networks.

In this chapter, we develop a fast community detection method that can easily scales to networks with millions of nodes. We build on our AGM (Chapter 5) and develop the Cluster-Affiliation Model for Big networks (BigCLAM), a generative model similar to AGM. In BigCLAM, we assign each node-community pair a nonnegative real-valued latent factor which represents the strength of membership of a node to the community. We then model the probability of an edge between a pair of nodes in the network as a function of the shared community affiliations. Like AGM, BigCLAM assumes that the nodes sharing more communities have higher edge probability between the nodes.

Similar to what we did with AGM in the previous chapter, we detect communities with BigCLAM by fitting the model to a given network. However, the key difference from the previous chapter is that fitting AGM requires combinatorial optimization whereas fitting
BigCLAM is continuous optimization. Moreover, we show that fitting BigCLAM (i.e., to estimate nonnegative community membership factors) is a variant of non-negative matrix factorization [90], which has been heavily studied recently. By combining the state-of-the-art nonnegative matrix factorization methods [99], we achieve gains both in the quality of detected communities as well as in scalability of the method. Experiments show that BigCLAM achieves near linear running time while other methods exhibit quadratic or exponential running time.

Similar to AGM, BigCLAM detects densely overlapping communities more accurately than the previous state-of-the-art. Moreover, BigCLAM is far more scalable than the existing methods (including AGM) as it is 100 times faster than the existing methods on average. We process networks of more than 35 million edges which improves by a factor of 10 the size of the largest networks that overlapping community detection methods could process in the past. Overall, BigCLAM improves over the current state of the art in both the scalability as well as the quality of detected communities. Code is available at http://snap.stanford.edu.

7.1.1 Outline of Chapter

Here is the outline of the chapter:

- We first present BigCLAM, a generative model which can be thought of as an approximation of AGM. Whereas AGM uses binary variables to represent community affiliations, our model in this chapter uses real-valued affiliation factors. (Section 7.2)

- We then describe how we can detect communities from a given network by fitting BigCLAM, i.e., we estimate the best-fit community affiliation factors. We develop a fitting procedure based on block coordinate gradient descent. Our method has linear running time in the number of edges (not the number of total pairs of nodes) and supports parallelization. (Section 7.3)

- We evaluate the performance of BigCLAM by conducting four experiments. The first experiment is measuring the running time on synthetic networks, where BigCLAM achieves 100 times faster speed than the fastest baseline. The next three
experiments are about the accuracy of community detection, where we evaluate the accuracy using ground-truth communities and the benchmark in Ahn et al. [4]. In all three experiments, BigCLAM outperforms the state-of-the-art baseline methods.

(Section 7.4)

7.2 Cluster-Affiliation Model for Big networks

In this section, we present the Cluster-Affiliation Model for Big networks (BigCLAM), a probabilistic generative model for networks. Similar to AGM, BigCLAM reliably captures the organization of networks based on community affiliations. However, whereas the AGM considers binary community affiliations, BigCLAM introduces real-valued factors to represent node-community affiliations. In this regard, BigCLAM approximates the generative process in the AGM using real-valued variables, i.e., we relax the formulation of AGM using continuous variables.

We now show how we derive BigCLAM by relaxing AGM. We begin by stating Eq. 5.1 in a new form:

\[
p(u, v) = 1 - \prod_{c \in C_{uv}} (1 - p_c) = 1 - \prod_{c} (1 - p_c)^{M_{uc}M_{vc}},
\]

where \( M_{uc} \) is an indicator variable whether node \( u \) belongs to community \( c \). By replacing \( 1 - p_c = \exp(-\alpha_c) \) with \( \alpha_c \geq 0 \), we can express the equation as a linear form of \( M \) and \( \alpha_c \):

\[
p(u, v) = 1 - \exp(- \sum_{c} M_{uc}\alpha_c M_{vc}).
\]

We then further simplify the equation by letting \( \tilde{M}_{uc} = \sqrt{\alpha_c} M_{uc} \).

\[
p(u, v) = 1 - \exp(-\tilde{M}_u\tilde{M}_v^{T}).
\]

Note that we did not use any approximation so far. So the maximum likelihood estimation of the model is still a combinatorial optimization problem (\( \tilde{M}_{uc} \in \{\sqrt{\alpha_c}, 0\} \)). \( \tilde{M}_{uc} \in \{\sqrt{\alpha_c}, 0\} \) means that if node \( u \) belongs to \( c \), it would be connected to other member nodes
in $c$ with the factor $\sqrt{\alpha_c}$. Therefore, we can interpret $\tilde{M}_{uc}$ as the level of participation of $u$ in community $c$, which then determines edge probability of $u$ to other nodes in $c$.

Now, we replace discrete-valued $\tilde{M}_{uc}$ with a continuous membership $F_{uc}$ which can be any nonnegative number. This way we actually model a level of participation of each node in a particular community as members with the higher value of $F_{uc}$ will be more likely to connect to other members of $c$. In other words, we assign a nonnegative weight $F_{uc}$ between node $u \in V$ and community $c \in C$ (Figure 7.1b). Using $F_{uc}$, we formally define the generative process for networks of BigCLAM as follows:

**Definition 2.** Let $F$ be a nonnegative matrix where $F_{uc}$ is a weight between node $u \in V$ and community $c \in C$. Given $F$, the BigCLAM generates a graph $G(V,E)$ by creating edge $(u,v)$ between a pair of nodes $u,v \in V$ with probability $p(u,v)$:

$$p(u,v) = 1 - \exp(-F_u \cdot F_v^T), \quad (7.1)$$

where $F_u$ is a weight vector for node $u$ ($F_u = F_{u\cdot}$).

Another way to derive Eq. (7.1) is the following. We assume that each community $c$ connects its member nodes $u, v$ with probability $1 - \exp(-F_{uc} \cdot F_{vc})$. Each community $c$ creates edges independently. However, if a pair of nodes gets connected multiple times, the duplicate edges are not included in the graph $G(V,E)$. Since each community $c$ connects $u, v$ independently with probability $1 - \exp(-F_{uc} \cdot F_{vc})$, the edge probability between $u$ and $v$ is $1 - \exp(- \sum_c F_{uc} \cdot F_{vc})$. 

---

**Figure 7.1:** (a) Bipartite community affiliation graph. Circles: Communities, Squares: Nodes of the underlying network. Edges indicate node community memberships. Edges with zero weight are not shown. (b) Each affiliation edge from node $u$ to community $c$ has strength $F_{uc} \geq 0$. 

CHAPTER 7. FAST ALGORITHM FOR FITTING AGM

This derivation of Eq. 7.1 clearly suggests that BigCLAM generates dense community overlaps, i.e., it generates an increasing relationship between edge probability and the number of shared communities. This is due to the fact that nodes that share multiple community memberships receive multiple chances to create a link. For example, pairs of purple nodes in the overlap of communities A and B in Figure 7.1a get two chances to create an edge. First they create an edge with probability \(1 - e^{-F_{uA} \cdot F_{vA}}\) (due to the membership to community A) and then also an edge with probability \(1 - e^{-F_{uB} \cdot F_{vB}}\) (due to membership to community B). The edge probability between these nodes is \(1 - e^{-F_{uA} \cdot F_{vA} + F_{uB} \cdot F_{vB}}\). If they were to reside in the non-overlapping region of A, they would be linked with probability \(1 - e^{-F_{uA} \cdot F_{vA}}\), which is smaller than \(1 - e^{-F_{uA} \cdot F_{vA} + F_{uB} \cdot F_{vB}}\).

The Eq. 7.1 also has the following probabilistic interpretation. Assume an undirected weighted network where pairs of nodes have a latent interaction of non-negative strength \(X_{uv}\). However, we only observe an undirected unweighted version of network \(G(V, E)\) where a pair of nodes \(u, v\) is connected if the corresponding \(X_{uv} > 0\). Now consider that nodes \(u, v\) generate an interaction of strength \(X_{uv}^{(c)}\) within each community \(c\) using a Poisson distribution with mean \(F_{uc} \cdot F_{vc}\). Then the total amount of interaction \(X_{uv}\) between nodes \(u\) and \(v\) is the sum of \(X_{uv}^{(c)}\):

\[X_{uv} = \sum_c X_{uv}^{(c)}, \quad X_{uv}^{(c)} \sim \text{Pois}(F_{uc} \cdot F_{vc}).\]

Then, due to the additivity of the Poisson random variable, \(X_{uv} \sim \text{Pois}(\sum_c F_{uc} \cdot F_{vc})\), and the edge probability \((P(X_{uv} > 0))\) is the same as \(p(u, v)\) in Eq. 5.1

\[P(X_{uv} > 0) = 1 - P(X_{uv} = 0) = 1 - \exp(-\sum_c F_{uc} \cdot F_{vc})\]

Note that node \(u\) with higher \(F_{uc}\) is more likely to be connected to other members of \(c\) as \(X_{uv}^{(c)}\) will have a higher mean.

Finally, we briefly mention that BigCLAM can naturally model non-overlapping, overlapping or hierarchically nested community structure (which were described in Figure 5.1.) \(\varepsilon\)-Community. Similar to AGM, in Eq. 7.1 BigCLAM does not allow for the edges between the nodes \(u\) and \(v\) that do not share any common communities, because for such
nodes $F_{uc} \cdot F_{vc} = 0$ for all $c$. To allow for edges between nodes that do not share any community affiliations, we assume an additional community, called the $\varepsilon$-community, as we did in Sec. 5.2 for the AGM. $\varepsilon$-community connects any pair of nodes with a very small probability $\varepsilon$. We use the same value for $\varepsilon$ as we did for the AGM ($\varepsilon = 1/|V|$).

### 7.3 Fitting Algorithm for BigCLAM

Now that we defined the BigCLAM model, we explain how to detect network communities using the model. Given an unlabeled undirected network $G(V, E)$, we aim to detect $K$ communities by fitting the BigCLAM (i.e., finding the most likely affiliation factor matrix $\hat{F} \in \mathbb{R}^{N \times K}$) to the underlying network $G$ by maximizing the likelihood $l(F) = \log P(G|F)$ of the underlying $G$:

$$\hat{F} = \arg\max_{F \geq 0} l(F),$$

where

$$l(F) = \sum_{(u,v) \in E} \log(1 - \exp(-F_{u}F_{v}^{T})) - \sum_{(u,v) \notin E} F_{u}F_{v}^{T}.$$ 

For now, we assume the number of communities $K$ is given. We will describe later how to automatically estimate $K$.

The optimization problem of Eq. 7.2 can be viewed as a variant of nonnegative matrix factorization (NMF) [90] where we learn $F \in \mathbb{R}^{N \times K}$ that best approximates the adjacency matrix $L$ of a given network $G$. By representing a negative log-likelihood $-l(F)$ as a loss function $D$ and $1 - \exp(\cdot)$ as a link function, we can represent the problem as follows:

$$\hat{F} = \arg\min_{F \geq 0} D(L, f(FF^{T}))$$

The benefit of using matrix factorization approach is increased scalability. Overlapping community detection methods have been developed to analyze small networks [4, 5], and most methods rely on combinatorial optimization which is hard to scale. On the other hand, for nonnegative matrix factorization many efficient techniques exist [71, 99].

BigCLAM modifies the existing NMF methods [71, 90, 99] and adapts them to large networks. While NMF methods use $l_2$ norm as an objective function, $l_2$ norm is not suitable
for modeling binary adjacency matrices [70]. Instead, BigCLAM employs log-likelihood as a loss function. Additional benefit is that for sparsely connected networks (which real networks are) our formulation allows for near-constant time gradient computation ($l_2$ takes linear time) which in practice speeds up our algorithm for a factor of 100.

### Solving the optimization problem.

To solve the problem in Eq. 7.2, we adopt a block coordinate gradient ascent algorithm [71, 99]. In particular, we update $F_u$ for each $u$ with the other $F_v$ fixed, i.e., we update the memberships of one node with fixing the membership of all other nodes. The main reason is that if we fix all $F_v$, then the problem of updating $F_u$ becomes a convex optimization problem. We solve the following subproblem for each $u$:

$$
\arg\max_{F_u \geq 0} l(F_u),
$$

where

$$
l(F_u) = \sum_{v \in \mathcal{N}(u)} \log(1 - \exp(-F_u F_v^T)) - \sum_{v \not\in \mathcal{N}(u)} F_u F_v^T,
$$

where $\mathcal{N}(u)$ is a set of neighbors of $u$. To solve this convex problem, we use projected gradient ascent. The gradient can be computed straightforwardly.

$$
\nabla l(F_u) = \sum_{v \in \mathcal{N}(u)} F_v \frac{\exp(-F_u F_v^T)}{1 - \exp(-F_u F_v^T)} - \sum_{v \not\in \mathcal{N}(u)} F_v
$$

We compute a step size using backtracking line search [19]. After update, we project $F_u$ into a space of nonnegative vectors by setting $F_{uc} = \max(F_{uc}, 0)$.

For a large network with more than a million nodes, this coordinate ascent is not very scalable as making a single step of coordinate ascent (i.e., computing $l(F_u)$ and $\nabla l(F_u)$) takes linear time $O(N)$. However, we reduce the complexity to $O(|\mathcal{N}(u)|)$ by computing $\sum_{v \not\in \mathcal{N}(u)} F_v$ efficiently. In particular, we notice:

$$
\sum_{v \not\in \mathcal{N}(u)} F_v = (\sum_v F_v - F_u - \sum_{v \in \mathcal{N}(u)} F_v)
$$

By storing $\sum_v F_v$, we can compute $\sum_{v \not\in \mathcal{N}(u)} F_u F_v^T$ in time $O(|\mathcal{N}(u)|)$. Given that real-world networks are extremely sparse ($|\mathcal{N}(u)| \ll N$), we can update $F_u$ for a single node
u in near-constant time. We iteratively update $F_u$ for each $u$ and stop the iteration if the likelihood does not increase (increase less than 0.001%) after we update $F_u$ for all $u$. In practice this speeds up our algorithm for two orders of magnitude and makes it practical to run it on networks with millions of nodes and edges.

**Determining community affiliations.** After we learn $\hat{F}$, we still have to determine whether $u$ belongs to community $c$ or not from the value of $F_{uc}$. To achieve this, we ignore the membership of node $u$ to community $c$ if $F_{uc}$ is below some threshold $\delta$. Otherwise ($F_{uc} \geq \delta$), we regard $u$ as belonging to $c$. We set $\delta$ so that if two nodes belong to community $c$, then their edge probability is higher than the background edge probability $\varepsilon = 1/|V|$ (see Section 7.2).

$$1/|V| \leq 1 - \exp(-\delta^2)$$

Solving this inequality, we set the value of $\delta = \sqrt{-\log(1 - 1/|V|)}$. Note we also experimented with other values of $\delta$ and found that our choice for $\delta$ gives overall good performance.

**Initialization.** To initialize $F$, we use locally minimal neighborhoods [54]. Neighborhoods $N(u)$ of node $u$ is a community of $u$ and its neighbors, and $N(u)$ is locally minimal if $N(u)$ has lower conductance than all the $N(v)$ for nodes $v$ who are connected to $u$. Recently, Gleich et al. [54] empirically showed that the locally minimal neighborhoods are good seed sets for community detection algorithms. For a node $u'$ who belongs to a locally minimal neighborhood $k$, we initialize $F_{u'k} = 1$, otherwise $F_{u'k} = 0$.

**Choosing the number of communities.** To find the number of communities $K$, we adopt the approach used in [5]. We reserve 20% of node pairs as a hold out set. Varying $K$, we fit the BigCLAM model with $K$ communities on the 80% of node pairs and then evaluate the likelihood of BigCLAM on the hold out set. The $K$ with the maximum hold out likelihood will be chosen as the number of communities.

**Implementation details.** Since the objective function of our optimization problem is not the $l_2$ norm, the methods for least squares NMF such as multiplicative update [90] or alternating least squares [71] are not applicable. We experimented with the cyclic coordinate descent method (CCD) [71] which optimizes $F_{uc}$ for each $u$ and each $c$ by the Newton’s method, but the method converged slower than our block coordinate ascent method. The
main reason for this is that the number of subproblems that we have to solve in CCD grows linearly with $K$, the number of communities. In matrix factorization, usually $K$ (the rank of $F$) is assumed to be a very small constant \[71, 99\]; however, in our problem $K$ increases as the size of the underlying network grows.

## 7.4 Experiments

We proceed by evaluating the performance of BigCLAM and comparing it to the state-of-the-art community detection methods on a range of networks from a number of different domains and research areas.

### 7.4.1 Experiments on Synthetic Networks

Using synthetic networks we investigate the scalability and convergence of the BigCLAM optimization problem.

**Convergence of BigCLAM.** Non-negative matrix factorization is non-convex which means that gradient based approaches do not guarantee to find an optimal solution. To verify that our fitting algorithm does not suffer too much from local optima, we conduct the following experiment on synthetic networks. We generated 100 synthetic networks using the AGM model \[154\]. For each of these networks, we then fit BigCLAM using 10 different random starting points and attempt to recover the true community affiliations.

In 98% of cases our fitting algorithm finds true communities with reliable accuracy (F1 score of node community memberships higher than 0.85), and in 27% of cases our algorithm discovers the communities almost perfectly (F1 score $> 0.95$). This result suggests that the optimization space has several local optima which almost equivalent to the global optimum.

**Scalability of BigCLAM.** We also evaluate the scalability of BigCLAM by measuring the running time on the networks of increasing sizes. For comparison, we compare the runtime of the following overlapping community detection methods:

- NMF: Least squares non-negative matrix factorization. We solve the following problem: $\text{argmax}_{F_{uk} \geq 0} \|L - F \cdot F^T\|_F$ where $L$ is an adjacency matrix of a given network.
CHAPTER 7. FAST ALGORITHM FOR FITTING AGM

Figure 7.2: **Algorithm runtime comparison.** BigCLAM runs 10 to 100 faster than competing approaches.

We used a projected gradient descent as we do with BigCLAM.

- **BigCLAM(Naive):** BigCLAM without the optimization in Eq. 7.4
- **LC:** Link Clustering method [4].
- **CPM:** Clique Percolation method [118].
- **MMSB:** Mixed-Membership Stochastic Block model [5].

Note that Link Clustering, Clique Percolation Method and Mixed-Membership Stochastic Block models have been explained in Section 6.3

Figure 7.2 shows the results. NMF, BigCLAM(Naive) and MMSB scale to networks of around 1,000. LC and CPM scale to networks of about 10,000 and then their runtime becomes prohibitively large. On the other hand BigCLAM can process networks with hundreds of thousands of nodes within 20 minutes. This means that BigCLAM can easily process networks 10 to 100 times larger than other approaches (and while also more accurately detecting communities). Last, note that the optimization of BigCLAM defined in Eq. 7.4 speeds up the algorithm for around 100 times and is thus essential for making BigCLAM scale to large networks.
CHAPTER 7.  FAST ALGORITHM FOR FITTING AGM

7.4.2 Experiments on Networks with Ground-truth Communities

We also examine the performance of BigCLAM using the 6 networks with ground-truth communities that we described in Chapter 3. We follow the same experimental setup as we did in Section 6.3 (the experiments for Figure 6.8). We use the same data sets, the same baselines, and the same evaluation metrics.

Figure 7.3 displays the composite performance of the methods over all six networks. On average, the composite performance of BigCLAM is 3.60, which is 79% higher than that of Link clustering (2.01), 45% higher than that of CPM (2.47), and 15% higher than that of MMSB (3.14). The absolute average value of Omega Index of BigCLAM over the 6 networks is 0.47, which is 24% higher than Link Clustering (0.38), 26% higher than CPM (0.37), and 30% higher than MMSB (0.36). In terms of absolute values of scores, BigCLAM archives the average F1 score of 0.60, average Omega index of 0.47, Mutual Information of 0.22 and accuracy of the number of communities of 0.43.

Overall, BigCLAM gives superior overall performance. This means that, while BigCLAM is two orders of magnitude more scalable than competing approaches, it also achieves superior performance in the quality of detected communities. On 4 out of 6 networks BigCLAM performs best by a big margin. However, we note that on DBLP and Amazon MMSB is the winning method mostly due to BigCLAM scoring very badly on a single individual metric (Number of communities on DBLP, Ω-index on Amazon). This occurs due to the fact that BigCLAM uses a single parameter ε to model the edge probability between
all pairs of different communities (ε-community in Section 7.2), while MMSB uses one parameter for each pair of communities. With more parameters, MMSB can fit these networks better. Note that BigCLAM could be easily extended to include a distinct parameter for the edge probability between each pair of communities.

### 7.4.3 Experiments on Networks in Ahn et al.

We further evaluate BigCLAM using performance benchmarks from Ahn et al. [4]. For this experiment, we follow the same procedure as we did in Section 6.5.

**Results.** Following [4] we compute the composite performance by normalizing the scores the same way as we did in the experiments with ground-truth communities. Figure 7.4 shows the composite performance of the four methods. The BigCLAM achieves best composite performance in 4 networks, and the second best in three networks. In all these cases MMSB slightly outperforms BigCLAM due to BigCLAM’s bad performance on the Overlap Coverage metric. Overlap Coverage is defined as the average number of communities that a node belongs to [4]. This metric is extremely ill posed since assigning nodes to more communities always improves the score. Since any non-zero stochastic membership found by MMSB is regarded as a valid community membership, the MMSB achieves extremely high score on the Overlap Coverage metric. Nevertheless, on average, the BigCLAM achieves a composite performance score of 3.06, outperforming Link Clustering (2.67) by 14%, Clique percolation (1.50) by 102%, and MMSB (2.84) by 8%.

### 7.4.4 Experiments on Large Networks

In addition to better accuracy, another strength of BigCLAM is its scalability. To test this, we apply BigCLAM to large real-world networks. We were able to run BigCLAM on 4 networks from in Chapter 3: LiveJournal, YouTube, Amazon, and DBLP.

To reduce the memory requirements of our method, we aim to find sparse latent factors. We achieve this by adding $l_1$ regularization term to Eq. 7.2 and optimize:

$$\arg\max_{F_{uc} \geq 0} l(F) - \lambda \sum_{u,c} |F_{uc}|$$
Since $l_1$ regularization introduces sparsity to matrix $F$, we only need to keep track of latent factors with non-zero value, which decreases the memory requirements of our method. We use $\lambda = 10$ for Amazon, YouTube, and DBLP and $\lambda = 5000$ for LiveJournal. We update $F_u$ (Solving Eq. 7.3) for multiple nodes in parallel. With 20 threads, it takes about one day to fit BigCLAM to the LiveJournal network (4M nodes, 35M edges).

As our baselines from the previous experiments do not scale to these networks, we consider two well-known graph partitioning methods as baselines: Metis [77] and Graclus [38]. For Graclus and Metis, we set the number of communities to detect $K$ to be the number of ground-truth communities and use the same $K$ for BigCLAM as well.

Similarly to experiments in Figure 7.3, we measure the accuracy of detected communities using F-1 score and Omega index (NMI is omitted as all the methods perform the same). Moreover, notice that ground-truth communities in our data are partially annotated as some nodes might not indicate their memberships. This means it is important to quantify the Recall of a given method. We define Recall as the average Recall of best-matching detected communities:

$$Recall(C^*, \hat{C}) = \frac{1}{|C^*|} \sum_{C_i \in C^*} Rc(C_i, C^*_g(i))$$
Table 7.1: Relative improvement of BigCLAM over Metis and Graclus in detecting communities in large scale networks. Positive value indicates that BigCLAM outperforms the baselines.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Ω-Index</th>
<th>F-1</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>LiveJournal</td>
<td>2.70</td>
<td>0.21</td>
<td>0.43</td>
</tr>
<tr>
<td>YouTube</td>
<td>1.60</td>
<td>0.39</td>
<td>0.82</td>
</tr>
<tr>
<td>Amazon</td>
<td>0.00</td>
<td>0.00</td>
<td>0.23</td>
</tr>
<tr>
<td>DBLP</td>
<td>0.10</td>
<td>0.03</td>
<td>0.29</td>
</tr>
<tr>
<td>Average</td>
<td>1.10</td>
<td>0.16</td>
<td>0.44</td>
</tr>
</tbody>
</table>

where \( R_c(C_i, \hat{C}_j) \) is the recall of \( \hat{C}_j \) under the best matching \( g \).

Since the two baselines (Graclus and Metis) perform very similarly in all metrics, we take just the best value among the two in each case rather than showing the result of baselines separately. For each network and each score, we pick the best score \( x \) among the two baselines and compute the relative improvement of BigCLAM over the \( x \), i.e., \( \frac{\text{Score}(\text{BigCLAM})-x}{x} \). Table 7.1 shows the relative improvement of BigCLAM over the baselines. For example, 0.21 for F-1 in LiveJournal means that BigCLAM achieves 21% higher F-1 score than the best baseline (Metis in this case).

Overall, BigCLAM outperforms the baselines in nearly all cases. On average, BigCLAM achieves 110% higher Omega index, 16% higher F-1 score, and 44% higher average Recall, which means that BigCLAM achieves 57% relative improvement on average among the three scores. Furthermore, BigCLAM outperforms the baselines in every measure and every network. The absolute value of the scores of BigCLAM is 0.11 (Omega index), 0.13 (F-1 score), and 0.32 (Recall). Overall, the results emphasize the need for a scalable and accurate overlapping community detection method as graph partitioning methods fail to detect overlapping communities. Results demonstrate that BigCLAM could be the needed solution.

7.5 Related Work

We discuss the previous work that is related to BigCLAM.

Low rank approximation of matrices. BigCLAM formulates community detection as a
variant of low rank approximation of matrices. Given a network with $N$ nodes, its adjacency matrix $L \in \{0, 1\}^{N \times N}$, and the number of communities to detect $K$, BigCLAM aims to learn a low rank matrix $F \in \mathbb{R}^{N \times K}$ (which specifies non-negative affiliation factors) so that we can closely reproduce the adjacency matrix $L$. The closest kind of low-rank approximation to BigCLAM is non-negative matrix factorization (NMF) \cite{90} where one aims to estimate non-negative factors $W, H \in \mathbb{R}^{N \times K}$ that minimize the Frobenious norm of the reconstruction error $\|L - WH^T\|$. However, BigCLAM has two important improvements. First, most of NMF research pays relatively little attention to interpreting the latent factors \cite{3}. The primary goal there is to estimate the missing entries of the matrix (e.g., as in the Netflix competition \cite{80}). On the other hand, BigCLAM aims to learn latent factors which represent community affiliations of nodes. Second, instead of using a Gaussian distribution \cite{71, 90} or logistic link function \cite{70}, we optimize the model likelihood of explaining the links of the observed network. Our choices result in better accuracy and better scalability of the method compared to NMF. In terms of accuracy, in \cite{160}, we showed that our formulation leads to better detection accuracy than using NMF for community detection. As for scalability, our formulation of likelihood allows us to compute a gradient of the factor matrix in near-constant time, which is significant improvement over existing NMF methods where the complexity of computing such gradient is linear in the number of rows of the matrix (i.e., nodes of the network). In practice, computing the gradient in near-constant time makes our algorithm about 1,000 times faster.

More broadly, BigCLAM is also related to the low-rank singular value decomposition (SVD) where the goal is to learn low-rank matrices $U, S, V$ so that $\|L - USV^T\|_F$. The constraints for the solution are that the columns of $U, V$ are orthonormal and $S$ is a diagonal matrix. Whereas NMF and BigCLAM resort to finding a locally optimal solution, SVD allows us to find the globally optimal solution for the problem \cite{40}. The time complexity for finding the exact solution for SVD is $O(N^3)$. Since this complexity is too slow for large $N$, researchers developed faster algorithms that can find an approximated solution in a near-linear time in $N$ \cite{2, 106}. However, the key difference in SVD compared to NMF and BigCLAM is that SVD does not have the non-negativity constraint on its solution. Because of this difference, the factors learned by SVD $(U, S, V)$ are hard to interpret due to mixed signs \cite{90}, and thus SVD is not suitable for our task of community detection.
Large scale overlapping community detection. We briefly discuss the current state of large scale overlapping community detection. In terms of scalability, most overlapping community detection methods scale to networks with at most thousands of nodes [5, 102, 118]. The largest network processed with overlapping community detection methods is a mobile phone network of 800,000 nodes and 2.8 million edges [4]. Non-overlapping community detection algorithms, which solve a simpler problem, have been applied to networks with millions of nodes [38, 77]. Our methods presented here can process networks with tens of millions of edges while also obtaining state of the art quality of detected communities.

7.6 Conclusion

In this chapter, we developed a novel scalable community detection method that accurately discovers the overlapping community structure of large scale real-world networks. We developed the Cluster-Affiliation Model for Big networks (BigCLAM), a relaxation of community detection with AGM. Our algorithm builds on the research of nonnegative matrix factorization and scale to networks with million nodes. Experiments show that the BigCLAM outperforms the state-of-the-art community detection methods in accurately discovering network communities as well as the overlaps between communities. Furthermore, BigCLAM can detect community structure in the LiveJournal network which is more than 10 times bigger than the previously largest network considered for overlapping community detection.

Our work has several implications: First, BigCLAM opens up a new possibility to combine the advances in community detection and nonnegative matrix factorization. Second and more importantly, our BigCLAM can broaden our understanding of organizing principles of large scale networks, whose community structures were unclear due to the lack of scalable community detection methods. In the next chapter, we investigate the overlapping community structure of various large networks by applying BigCLAM to the networks.
Chapter 8

Implications to Core-Periphery

8.1 Introduction

In the previous chapter, we developed BigCLAM, a novel, accurate, and scalable method for overlapping community detection. With BigCLAM, we can now identify overlapping communities in networks with millions of nodes, which was not possible so far due to lack of scalable overlapping community detection methods.

In this section, we study how communities are organized in large real-world networks and how communities overlap with each other. Our analysis provides a novel insight for the organizing principles of large networks; we find that densely overlapping communities lead to a core-periphery structure [18, 68, 127]. In the core of the network, many communities overlap and thus the edge density in the core becomes high, whereas in the periphery communities rarely overlap and the edges are sparse. We find consistent evidence for this insight in our experiments on many large real-world networks including social networks, information networks, biological networks, and ecological networks.

The core-periphery structure means that the nodes in the “core” (i.e., the central part of a network) are densely connected to each other, whereas the nodes in the “periphery” are sparsely connected. This structure was first mathematically formulated by Borgatti and Everett [18], and then many methods [68, 127] have been developed to quantify the core-periphery structure in networks. All these methods confirm that core-periphery is a pervasive and crucial characteristic of large real-world networks. However, despite many
observations on how core-periphery manifests in the networks, there is little explanation on why core-periphery emerges. In this chapter, we provide a solution to this question for the first time. Our results — core-periphery stems from dense community overlaps — give the first conceptual explanation for core-periphery.

One reason why the formation of the core-periphery structure is hard to explain is because core-periphery is in conflict with the notion of network communities. Previous community detection methods tend to regard a core of the networks as a gigantic single community and thus it was conjectured that the core is absent of network communities. For example, Clauset et al. [32] detected 1,684 communities with a Modularity-based community detection method in the Amazon product network (409,687 nodes), and the authors found that 1,682 communities (99.9% of the total communities) exist in the periphery that consists of about 50% of the nodes. On the other hands, the authors detected only two communities in the core (50% of nodes). Leskovec et al. [97] also observed that the communities detected by the local spectral clustering method [7] exist only in the periphery of networks. However, our results with BigCLAM disprove this previous conjecture; rather, we observe that the core in the networks contains more communities than the periphery.

The success of BigCLAM (and AGM) for core-periphery networks is due to our correct assumption that communities form dense overlaps. Previous methods, assuming that community overlaps are sparse, cannot interpret network cores as overlaps, and thus these methods make mistake by regarding the cores as separate communities (as we show in Section 4.3). On the other hand, our BigCLAM correctly recognizes dense network cores as the results of many overlapping communities.

8.1.1 Outline of Chapter

Here is the outline of the chapter:

- We first describe the networks that we consider. We consider thirteen networks from social networks to biological networks. (Section 8.2)

- We then present our findings that the global core-periphery structure stems from densely overlapping communities. We find the same phenomena consistently in all the networks that we consider. (Section 8.3)
• Using the connection between core-periphery and overlapping communities, we dis-
cover that biological networks and the Amazon product network have local cores whereas social networks, web graphs, and foodwebs have only global core. (Section 8.4)

• Finally, we compare our methodology to the state-of-the-art methods for identifying the core-periphery structure. We validate that our methodology correspond very well to the existing methods, which suggests that we correctly identify the core-periphery structure of networks. (Section 8.5)

8.2 Dataset Description

We begin by describing the networks that we consider for these experiments (Table 8.1). Because our goal is to discover network communities in these networks, we do not need ground-truth communities to be defined in these networks. Therefore, we can extend our focus on the domains where ground-truth communities are hard to define (such as foodwebs or webgraphs). Overall, we consider the following thirteen different large-scale networks that include social, information, biological, and the web networks.

• Three social networks. LiveJournal online social network was described in Section 3.3. MSN Finland is the MSN network of users in Finland, and the LinkedIn network is the snapshot of the LinkedIn social network when it had 254,151 users.

• The Amazon product network described in Section 3.3.

• Three web graphs where nodes are web pages and edges mean hyperlinks [97]. Web-
Stanford is a network of the web pages from Stanford University, Web-NotreDame is of the web pages from University of Notre Dame, and Web-BerkStan is of the pages from Stanford University and University of California Berkeley.

• 4 protein-protein interaction networks described in Section 6.4.

• 2 networks of the Florida Bay food web networks [142]: In wet season (Foodweb-wet) and in dry season (Foodweb-dry).
Table 8.1: Network statistics. $N$: Number of nodes, $E$: Number of edges, $\langle C \rangle$: Average clustering coefficient [148], $\langle D \rangle$: Average shortest path length, $\langle k \rangle$: Average node degree. The 3 networks in the first block are social networks, the second block shows the Amazon product network described in Section 3.3, and the 3 networks in the third block are web graphs. The 4 networks in the fourth block are the protein-protein interaction networks of *Saccharomyces cerevisiae* which are described in Section 6.4, and the 2 networks in the last block are foodweb networks.

8.3 Dense Community Overlaps Lead to Global Core-periphery

In this section, we gain well-founded insight on the organizing principles of networks by applying BigCLAM to the networks that we described in Section 8.2. In particular, we observe that densely overlapping network communities lead to a global core-periphery network structure. We observe that a network core forms as a result of a convolution of many overlapping communities.

In the thirteen networks, we measure the following two quantities: How central a node is, and how many communities (detected by BigCLAM) that the node belongs to. More
CHAPTER 8. IMPLICATIONS TO CORE-PERIPHERY

Figure 8.1: Overlapping communities lead to global core-periphery network structure. The average (and the 90-th percentile) of the number of community memberships $m$ of a node as a function of the average shortest path length $d$ to all other nodes of the network. The number of community memberships increases with the centrality of a node. Nodes that reside in the center of the network, and have small shortest path distances to other nodes of the network, tend to belong to the highest number of communities.
Densely overlapping communities lead to global core-periphery. A hypothesis example that shows how overlapping communities lead to the global core-periphery structure. Communities (colored ellipses) overlap pervasively in the center and do not overlap in the periphery. As a result, a dense global core forms in the center of the network.

Formally, given the network $G(V, E)$, we measure the average number of BigCLAM communities $m$ that a node belongs to as a function of the farness centrality $d$ of the node. The farness centrality $d$ of node $u$ is the average shortest path length from $u$ to all other nodes in the network, i.e., $d = \frac{1}{|V|} \sum_{v \in V} d(u, v)$ where $d(u, v)$ is the shortest path length from $u$ to $v$.

Figure 8.1 displays the plots of $m$ and $d$ for all thirteen networks. In all networks but the two web graphs, the number of community memberships of a node decreases with the farness centrality of a node, which implies that nodes residing in the center of the network, which have small shortest path distances to other nodes of the network, tend to belong to the highest number of communities. In addition, we previously observed that the edge density increases as a function of community memberships (Figure 4.1). Thus, our findings imply that the nodes in the center of network have higher density of edges than the node in the periphery because they belong to more communities.
Our findings in this section suggest that a network with a core-periphery structure organizes like Figure 8.2. In the network in the figure, communities overlap heavily in the center and rarely in the periphery. The nodes in the core (where many communities overlap) have high edge densities, while the nodes in non-overlapping part of communities (i.e., periphery of the network) have sparse edge densities.

To the best of our knowledge, Figure 8.2 provides the first conceptual explanation why the core-periphery structure arises. Moreover, this is the first successful attempt to reconcile the core-periphery structure and community structure; we show that even in the presence of many communities, dense community overlaps lead to a global core-periphery network structure.

8.4 Existence of Local Cores

In Section 8.3, we found that high-membership nodes form network cores. Using this finding, in this section we develop a methodology to determine how many cores exist in a given network. In other words, we examine whether a given network has local (secondary) cores as well as the global core or not.

We determine the existence of local cores in the networks by the following methodology. We measure the fraction of the largest connected component (LCC) in the induced subgraph of the nodes who belong to at least $l$ communities. Thinking of a network as a valley where peaks correspond to cores and peripheries to lowlands, our methodology is analogous to flooding lowlands and measuring the fraction of the largest island (which was a peak before flooding). High $c(l)$ means that there is a single dominant core (peak). Whereas, low $c(l)$ suggests the existence of nontrivial secondary cores, because flooding results in several isolated islands if there are multiple peaks.

Figure 8.3 displays $c(l)$ for each of the 5 types of networks. We observe that the protein-protein interaction networks and the Amazon product network have local cores, while the other types have a global core.

Maximum overlap fraction. We also characterize how much communities overlap with
Figure 8.3: **Largest connected component size on an induced subgraph of nodes belonging to at least \( l \) communities.**

Each other in different types of networks. Maximum overlap fraction \( o_c \) of a given community \( c \) quantifies the fraction of \( c \)'s members in the largest overlap with any other community.

Figure 8.4 shows the distribution of \( o \) in the 5 types of networks. Communities in the protein-protein interaction networks, social and product co-purchasing network are mainly non-overlapping whereas the communities in the foodweb and the web graph are pervasively overlapping.

**Implications for different types of networks.** Our results reveal that communities in protein and product networks have small overlaps and also form many local cores (Figure 8.5). In particular, protein communities only slightly overlap and form local cores as well as a small global core (Figure 8.5b). Small overlaps of protein communities can be explained by the fact that communities act as functional modules and it would be hard for the cell to independently control heavily overlapping modules \([81, 4]\). Communities of co-purchased products can also be thought of as functional modules since the products in a community are
CHAPTER 8. IMPLICATIONS TO CORE-PERIPHERY

bought together for a specific purpose. On the other hand, foodweb communities overlap pervasively while forming a single dominant core. This leads to a flower-like overlapping community structure (Figure 8.5a) where tiles (communities) overlap centrically to form a central core of the network. The heavily overlapping foodweb communities form due to the closed nature of the studied Florida bay ecosystem [142]. Web communities overlap moderately and form a single global core. Many of these communities form around common interests or topics, which may overlap with each other [49].

8.5 Comparison to other notions of core-periphery

In order to argue about the core-periphery structure of networks we so far used the fact that communities behave as tiles in the sense that overlap of two communities leads to higher edge density (higher tile thickness). Combining this with the observation that communities overlap most pervasively in the center of the network leads to the conclusion about
the global core-periphery structure. However, there are many other methods that identify core-periphery structure in networks and our goal is to quantify the agreement of our methodology and existing methods.

We aim to quantify the agreement between cores we find here and the cores detected by existing methods. In particular, we compare the method invented by Rombach et al. [127]. Since BigCLAM is proposed for community detection rather than core detection, our goal here is to measure the correspondence between the core determined by [127] and the core (i.e., high membership nodes) detected by the BigCLAM.

Rombach et al. computes a real-valued “core score” $CS(i)$ for each node $i$ which specifies how likely $i$ belongs to a core. In our experiments we used the number of community memberships $m(i)$ of node $i$ to indicate whether $i$ belongs to the core or not. Since $m(i)$ and $CS(i)$ are scores rather than binary indicators, we aim to measure the Pearson correlation coefficient $[30]$ between $m(i)$ and $CS(i)$.

For these experiments we consider two networks that were also considered in Rombach et al. [127]. First, we use the Zachary’s karate club network [162]. And second, we also
Table 8.2: **Comparison with the cores detected by Rombach et al. [127].** The Pearson’s correlation coefficient between the core score computed by Rombach et al. [127] and the number of communities the node belongs to as determined by BigCLAM. \( p \)-values are also computed for the null hypothesis using the Student’s \( t \)-test. High correlation coefficient implies that high membership nodes under the BigCLAM are more likely to belong the network core as detected by Rombach et al.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Zachary</th>
<th>London underground</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correlation coefficient</td>
<td>0.774</td>
<td>0.408</td>
</tr>
<tr>
<td>( p )-value</td>
<td>4.01 \times 10^{-8}</td>
<td>6.12 \times 10^{-4}</td>
</tr>
</tbody>
</table>

We consider the London underground network between the metro stations. Since the London underground network is a weighted network, we build an unweighted network for BigCLAM by connecting two nodes when the edge weight is larger than 2. In Table 8.2, we observe the correlation coefficient for the Zachary’s karate club network is 0.774, and 0.408 for the London underground network. In the second row of the table, we also compute the \( p \)-value for the null hypothesis that there is no positive correlation between the two values. We use Student’s \( t \)-test to achieve this. As \( p \)-values are far lower than the standard 0.05, we confirm that the cores that we find by BigCLAM (i.e., the high membership nodes) correspond well to the cores found by the state-of-the-art methods. The level of correlation is lower in the London underground network, which can be explained by the fact that some information is lost when converting the weighted network to an unweighted network.

### 8.6 Related Work

The core-periphery structure has been observed predominantly in many large real-world networks [68][127]. Most approaches on the core-periphery structure focused on identifying how central the nodes in the network are. [18] developed an simulated-annealing method to determine whether the nodes are in the core or in the periphery. Building on this method, [127] proposed a “Core score” which specifies how central a node is. Rather than using simulated annealing, [68] employed a data-driven measure for the centrality of the nodes. In our work, we use the community memberships of a node as a novel indicator for the centrality of the node. However, the crucial difference in our work is that our work
sheds light on why the core-periphery structure arises. We find that core-periphery stems from dense community overlaps and it is the first conceptual explanation for core-periphery to the best of our knowledge.

The existing community detection methods tend to fail to detect communities in the dense cores of networks [32]. As a result, it was conjectured that the core is a void of communities [97]. However, in this thesis, we find a new paradigm that network communities form dense community overlaps. Based on the new paradigm, we discover that the network cores are the overlaps of communities rather than a void of communities. Our results successfully reconcile two important but conflicting organizing principles of networks: The core-periphery structure and the community structure.

8.7 Conclusion

In this section, we observed that densely overlapping communities lead to a global core-periphery structure. We also find that biological networks and product networks have local cores whereas webgraphs, social networks, and foodwebs have only global core. Our results provide a novel explanation for the formation of core-periphery structures and reconcile the core-periphery structure and community detection. We believe our methodology can be used and extended to examine the core-periphery structures of networks in finer details.

This chapter concludes our work with AGM and BigCLAM. In the next chapters, we discuss how we can extend AGM and BigCLAM for community detection in directed networks or in networks with node attributes.
Chapter 9

Detecting 2-mode Communities as Well as Cohesive Communities

So far, we presented AGM and BigCLAM, novel overlapping community detection methods. For AGM and BigCLAM, the input to the methods is an undirected network and the output is the community memberships of nodes. In this chapter and the next chapter, we seek to extend the AGM and the BigCLAM for more diverse settings. In this chapter, we develop a novel method for detecting communities where the member nodes are connected in a bipartite fashion, rather than being densely connected each other. In addition, our method can handle both directed networks and undirected networks. To our surprise, we can develop this extension by making a trivial modification on AGM and BigCLAM.

9.1 Introduction

Traditional community detection methods thought of communities as groups of nodes with many connections among the group’s members, but few to the rest of the network [4, 34, 50, 113]. Our AGM and BigCLAM do the same. However, we note that dense communities are but one kind of mesoscale structure in networks, and there may be other structures that help us to understand networks better. As an example, consider a Twitter follower network and the ‘community’ of candidates in the 2012 U.S. presidential election. This community is not densely interlinked, in the sense that the candidates—being adversaries—do not follow
CHAPTER 9. 2-MODE COMMUNITIES

Figure 9.1: Two types of networks (directed and undirected) and two types of communities (cohesive and 2-mode). While research focus has predominantly been on undirected-cohesive communities (top left), we develop a method that can detect cohesive as well as 2-mode communities in both directed and undirected networks.

each other; thus we would not identify these individuals as a ‘community’ if we were to search for densely connected sets of nodes. However, such communities are uniquely characterized by a certain structural signature: in particular, they form around nodes whose outgoing edges have similar endpoints, or around nodes whose incoming edges have similar sources. Continuing our example, presidential candidates form a community in Twitter precisely because they have many followers in common.

Thus we argue that communities may be characterized by their external rather than their internal connectivity: e.g., members of a community may be linked to the same set of endpoints, even if they do not link to each other. Generally, we expect communities such as these to arise when sets of ‘fans’ link to sets of ‘celebrities’. We refer to such communities, which are characterized by connectivity between, rather than within groups, as ‘2-mode’. Similar examples also exist beyond social networks; for example, in protein-protein interaction networks, some protein complexes act as bridges or regulators, i.e., they do not interact among themselves but regulate/interact with the same set of proteins [120].
Another common assumption made by traditional methods is that networks are undirected \[118, 152\], which implies that relationships between connected nodes are symmetric or reciprocal. However, in directed networks relationships are asymmetric, as with our previous example about ‘fans’ who follow ‘celebrities’. Present methods can often be adapted to handle directed networks, though this is often done in an ad-hoc fashion (e.g., by treating directed edges as though they were undirected), though this can lead to unexpected or undesirable results \[50, 91, 129\]. We believe that by ignoring edge directedness we may lose important information, especially if relationships are predominantly non-reciprocal. This is certainly the case in predator-prey networks \[91\] or in online social networks like Twitter \[84\].

**Present work: Approach.** Here we consider new notions of community linking structure that go beyond thinking of communities as internally well-connected sets of nodes. Our work stems from social network literature on structural equivalence \[23\], where it has been noted that social homogeneity (i.e., social communities) arises not only between nodes that link to each other (i.e., internal group connectivity), but also between nodes that link to the rest of the network in a coordinated way (i.e., external group connectivity). In particular, we consider different notions of “communities” that are depicted in Figure 9.1. We differentiate between cohesive communities (Fig. 9.1 Cohesive) and 2-mode communities (Fig. 9.1 2-mode) where nodes link in a bipartite fashion with links predominantly appearing between partitions rather than inside them.

While existing community detection methods typically focus on Undirected-Cohesive or Directed-Cohesive communities \[34, 50, 91, 118, 152\], the focus of our paper is on developing methods that can detect communities of all four different types depicted in Figure 9.1. By modeling each of these definitions in concert, we are able to capture the complex structure present in networks.

**Present work: Communities through Directed Affiliations.** We present CoDA (Communities through Directed Affiliations), a BigCLAM’s extension that scales to networks with millions of nodes and tens of millions of edges. CoDA exhibits the following three properties: First, it naturally detects both cohesively connected as well as 2-mode communities. Second, CoDA allows cohesive and 2-mode communities to overlap or be hierarchically nested. And third, CoDA naturally allows for community detection in directed as well as
CHAPTER 9. 2-MODE COMMUNITIES

undirected networks.

Like what we did for AGM (Chapter 5) and BigCLAM (Chapter 7), we develop our community detection method by first presenting a generative model for networks. Then we fit the model to a given network and thus discover communities.

Our model builds on the AGM (Chapter 5), where nodes of the underlying network represent one ‘layer’ of the bipartite graph and communities represent the other. Edges between network-nodes and community-nodes in the affiliation graph represent memberships of nodes to communities. However, our model has one critical modification: while memberships of nodes to communities have previously been modeled as undirected, we model directed memberships.

Though simple on the surface, this modification leads to substantial changes in the modeling capability of affiliation network models. In particular, a directed affiliation between a node and a community models whether the node receives or creates links (or both) to other members of the community. Directed affiliations allow us to simultaneously model cohesive as well as 2-mode communities. In cohesive communities node affiliations are bidirectional (a node links to other members and also receives links from them); 2-mode communities are modeled with unidirectional memberships where some members mostly create links (i.e., fans) while others mostly receive them (i.e., celebrities).

We then develop a fast method to fit our node-community affiliation model to a given network, by employing similar approximation techniques as used in BigCLAM (Chapter 7). Our experiments demonstrate that CoDA can reliably detect both 2-mode and cohesive communities in real-world networks. Moreover, CoDA can easily handle networks with millions of nodes and CoDA can be easily parallelized.

Present work: Experimental results. We evaluate CoDA on a number of networks from several domains. We consider social, biological, communication, and ecological networks and evaluate CoDA on networks with explicitly labeled ground-truth communities [102, 153] as well as on networks where communities can be manually examined.

Our experiments demonstrate that CoDA’s ability to detect 2-mode as well as cohesive communities leads to improved performance over the existing state-of-the-art. For example, when detecting social circles in the Google+ follower network, our method gives a relative improvement in accuracy of 36% over Link Clustering [4] (28% over MMSB [5],
25% over Clique Percolation Method [118] and 21% over DEMON [34]).

Moreover, CoDA facilitates novel discoveries about the community structure of real-world networks. For example, we find that 2-mode communities in foodwebs describing predatory relationships correspond to groups of predators who rely on similar groups of prey. In citation networks among scientific papers, protein-protein interaction networks, as well as web graphs, a majority of detected communities are 2-mode. In social networks where edges signify reciprocal friendships, cohesive communities are more frequent. However, in Twitter or Google+, where relationships are asymmetric, 2-mode communities represent a significant portion of the network (20% in Twitter and 30% in Google+).

Further related work. Whereas communities were thought of as densely connected sets of nodes with few connections to the rest of the network [4, 36, 118, 123], the notion of structural equivalence suggests that nodes with similar connectivity patterns may be considered a community even if they do not connect to each other [23, 83, 66]. Our work builds on both definitions of network ‘communities’, and attempts to resolve them using a single, unified model.

It was also noted in [50, 91, 129] that algorithms developed for undirected networks may produce unexpected results if naively applied to directed networks. We argue that by modeling both cohesive and 2-mode communities in concert, we are able to model both types of network using a single model.

Trawling [83] is a method for extracting 2-mode communities in large directed networks. The critical difference with our work here is that Trawling identifies complete bipartite subgraphs of a given directed network, while our method relaxes this and is able to identify any bipartite community of significant edge density in directed as well as undirected networks.

CoDA is related to existing work on block models, which are in principle capable of detecting cohesive as well as 2-mode communities [5, 55, 67]. Our work differs from such approaches in terms of how communities overlap and are hierarchically nested. We also emphasize the scalability of CoDA compared to these approaches.
CHAPTER 9. 2-MODE COMMUNITIES

9.1.1 Outline of Chapter

Here is the outline of the chapter:

- We build on AGM and present a generative model for directed (and undirected) networks. Our model can naturally represent 2-mode communities as well as cohesive communities. (Section 9.2)

- We then present CoDA, a novel community detection method for 2-mode communities and cohesive communities. By extending BigCLAM, we develop an efficient fitting procedure to fit our generative model to a given network. (Section 9.3)

- We perform experiments to evaluate the performance of CoDA. We measure how well CoDA discovers hand-labeled ground-truth communities in social networks and email networks. The networks we consider contain both directed and undirected networks. Overall, CoDA outperforms the state-of-the-art baseline methods. (Section 9.4)

- Finally, we measure the fraction of 2-mode communities and cohesive communities in networks coming from various domains such as webgraphs, citation networks, and biological networks. Our results provide valuable insights on how networks organize around communities in different domains. (Section 9.5)

9.2 Models for Directed Community Affiliations

We start by presenting a generative model of networks with 2-mode and cohesive communities. We then proceed to develop an efficient model fitting procedure which allows us to detect community affiliations of nodes in a given network.

We describe our model in the context of directed networks and then show how it can straightforwardly be adapted to undirected networks. Our model builds upon the AGM (Chapter 5). However, whereas the AGM focuses on modeling only cohesive communities in undirected networks, our work here aims to find 2-mode communities as well as cohesive communities in both directed and undirected networks.
**Directed Affiliation Network Model.** We begin with the intuition that a desirable model of communities in directed networks should exhibit two properties. First, communities should be modeled not only in terms of their internal connectivity, but also in terms how members connected to non-members. Second, the model should account for asymmetries, *i.e.*, directedness, of relationships between nodes. We later demonstrate that accounting for these two properties is important. Perhaps surprisingly, we observe that it leads to better performance even when modeling communities in *undirected* networks. This is due to the fact that when edge directions are not explicit, relationships in the network may still be (implicitly) asymmetric, and identifying such asymmetries leads to improved performance.

We proceed by formulating a simple conceptual model of networks that we refer to as a *Directed Affiliation Network Model*. Our work builds on AGM (Chapter 5) and other affiliation network models [21], however, the AGM and other affiliation models are typically designed to handle cohesive communities in undirected networks [88, 152, 156, 163]; here we extend such models in order to capture cohesive as well as 2-mode communities in directed as well as undirected networks.

Similar to AGM, we consider a bipartite affiliation graph where the nodes of the network connect to communities to which they belong (Figure 9.2a). Edges of the underlying network (Figure 9.2b) then arise due to shared community affiliations.

Consider for a moment an undirected network; when a node belongs to a community in such a network it typically means that the node has (undirected) relationships to other members of the community. This type of community affiliation can be modeled by the AGM where undirected affiliations are formed between nodes and communities [88, 152, 156, 163].

In directed networks, however, we need a richer notion of community affiliation (Figure 9.2a): a node may have many edges that go to other members of a community, and also many edges from other members of a community, or both. Therefore, we assume that nodes in directed networks can have two ‘types’ of community affiliation: Outgoing affiliations from nodes to communities mean that in the network the nodes send edges to other members of the community. And, incoming affiliations from communities to nodes mean that nodes receive edges from other community members. We model this using *directed*
memberships between nodes and communities: outgoing memberships and incoming memberships.

Formally, we denote a bipartite affiliation graph as $B(V, C, M)$, where $V$ is the set of nodes of the underlying network $G$, $C$ the set of communities, and $M$ the directed edge set. An outgoing membership of a node $u \in V$ to a community $c \in C$ is denoted as $(u, c) \in M$ and, and an incoming membership is denoted as $(c, u) \in M$.

Now, given the affiliation graph $B(V, C, M)$, we need to specify a process that generates the edges $E$ of the underlying directed network $G(V, E)$. As we did in the AGM, we consider a simple parameterization where we assign a single parameter $p_c$ to every community $c \in C$. The parameter $p_c$ models the probability of a directed edge forming from a member node $u$ with an outgoing membership to community $c$ to another member $v$ of $c$ with an incoming membership. In other words, we generate a directed edge between a pair of network nodes with probability $p_c$ if they are connected in $B$ with a 2-step directed path via community-node $c$.

**Definition 3** (Directed Affiliation Network Model). *Let $B(V, C, M)$ be a directed bipartite graph where $V$ is a set of nodes, $C$ is a set of communities, and $M$ is a set of directed edges between $V$ and $C$. Also, let $\{p_c\}$ be a set of probabilities for all $c \in C$. Given $B(V, C, M)$ and $\{p_c\}$, the model generates a directed graph $G(V, E)$ by creating directed edges $(u, v)$
CHAPTER 9. 2-MODE COMMUNITIES

Figure 9.3: Affiliation graph (top) of the Directed Affiliation Network Model that corresponds to the network adjacency matrix (bottom). It contains two overlapping cohesive (A, B) and two overlapping 2-mode (C, D) communities. Black edges in the affiliation graph denote bidirectional community memberships and red edges denote unidirectional memberships.

from node $u \in V$ to node $v \in V$ with probability $p(u, v)$:

$$ p(u, v) = 1 - \prod_{k \in C_{uv}} (1 - p_k), \quad (9.1) $$

where $C_{uv} \subseteq C$ is a set of communities through which $u$ has a 2-step directed path to $v$ ($C_{uv} = \{c|(u, c), (c, v) \in M\}$). If $C_{uv} = \emptyset$ then we set $p(u, v) = 1/|V|$.

The directed affiliation network model and the underlying network it generated are illustrated in Figure 9.3. Directed affiliations are able to explain the overlapping nature of cohesive as well as 2-mode communities. For example, imagine a Twitter network among community of music fans (A), a community of movie fans (B), a group of famous singers (C) and a group of famous actors (D). Members in communities A and B build
bi-directional social relationships inside their respective communities. Some nodes may belong to both communities A and B as they are interested in both movies and music. As for one-directional relationships, we can easily see that music fans would follow singers (C) and movie fans would follow actors (D). Together, these relations would form the adjacency matrix at the bottom of Figure 9.3. Our model captures this complex community structure very naturally, as shown in the community affiliation graph above the adjacency matrix, where green nodes represent music fans (A), blue nodes are movie fans (B), red nodes are fans of both movies and music, ivory nodes are singers (C), and purple nodes are actors (D). Affiliations between nodes and cohesive communities A and B flow in both directions because members of those communities have reciprocal relationships with each other, whereas fans and celebrities belonging to 2-mode communities C and D have edges flowing in only one direction (fans follow celebrities, celebrities are followed by fans).

More generally, our model has two important advantages over existing approaches [88, 152, 156, 163]. First, our model generates natural overlaps between communities. In Chapter 4 and 5, we showed that our AGM can model community overlaps accurately, which traditional models of overlapping communities fail to capture [4, 5, 118]. Like AGM, our model also captures realistic community overlaps because its modeling power generalizes that of AGM for undirected networks. The second advantage of our model is its ability to model 2-mode communities. By modeling such communities, we can better capture the interaction between groups of nodes. This is a significant improvement over present approaches that model only the interaction within communities.

9.3 Community Detection by CoDA

Given an unlabeled, directed network $G(V, E)$, our goal is to identify cohesive as well as 2-mode communities. We achieve this by fitting our Directed Affiliation Network Model to $G(V, E)$, i.e., by finding an affiliation graph $B$ and parameters $\{p_c\}$ that maximize the likelihood. For now, we assume that the number of communities $K$ is given; we will later discuss how to automatically determine $K$. We aim to solve the following Maximum
Likelihood Estimation problem:

$$\arg\max_{P, \{p_c\}} \sum_{(u,v) \in E} \log p(u,v) + \sum_{(u,v) \notin E} \log(1 - p(u,v)), \quad (9.2)$$

where the edge probability $p(u, v)$ is defined in Eq. 9.1.

To solve the problem in Eq. 9.2, we develop a fast approximation algorithm that is similar to BigCLAM. In particular, we develop an approximate algorithm to optimize the problem in Eq. 9.2 by first relaxing binary memberships into real-valued memberships. We begin by introducing variables to represent the memberships of the nodes. As noted earlier, we distinguish incoming memberships and outgoing memberships. We let $M_{uc}$ indicate whether the node $u$ belongs to community $c$ with an outgoing membership, and $L_{vc}$ indicate whether node $v$ has an incoming membership for $c$. Using these notations, Eq. 9.1 can be represented in a new form:

$$p(u,v) = 1 - \prod_{c \in C^{uv}} (1 - p_c) = 1 - \prod_{c} (1 - p_c)^{M_{uc}L_{vc}},$$

By applying the change of variables $1 - p_c = \exp(-\alpha_c)$ with $\alpha_c \geq 0$, the equation becomes linear in $M$, $L$, and $\alpha_c$:

$$p(u,v) = 1 - \exp(-\sum_{c} M_{uc}\alpha_cL_{vc}).$$

We then further simplify the equation by letting $\tilde{M}_{uc} = \sqrt{\alpha_c} M_{uc}$ and $\tilde{L}_{vc} = \sqrt{\alpha_c} L_{vc}$.

$$p(u,v) = 1 - \exp(-\sum_{c} \tilde{M}_{uc}\tilde{L}_{vc}).$$

So far, we restrict $\tilde{M}_{uc} \in \{\sqrt{\alpha_c}, 0\}$ and $\tilde{L}_{vc} \in \{\sqrt{\alpha_c}, 0\}$. Note that we can interpret $\tilde{M}_{uc}$ as the strength of membership for user $u$ to community $c$; the condition $\tilde{M}_{uc} \in \{\sqrt{\alpha_c}, 0\}$ means that if node $u$ belongs to $c$, it would be connected to other member nodes in $c$ with the factor $\sqrt{\alpha_c}$, which determines $p_c$. The same argument also applies to $\tilde{L}_{vc}$.

We replace $\tilde{M}_{uc}$ and $\tilde{L}_{vc}$ with continuous valued memberships $F_{uc}$ and $H_{vc}$ (respectively), which can be any nonnegative number. A high value of $F_{uc}$ means that the node $u$
has many outgoing edges towards other members of $c$, while high $H_{vc}$ means that node $v$ has many incoming edges from other members of $c$. We write:

$$p(u, v) = 1 - \exp(-F_u H_v^T).$$

Now, we transform Eq. 9.2 into a continuous optimization problem:

$$\{\hat{F}, \hat{H}\} = \arg\max_{F, H \geq 0} l(F, H)$$

(9.3)

where

$$l(F, H) = \sum_{(u, v) \in E} \log(1 - \exp(-F_u H_v^T)) - \sum_{(u, v) \not\in E} F_u H_v^T.$$

In other words, to detect network communities we fit our model by estimating affiliation matrices $\hat{F}, \hat{H} \in \mathbb{R}^{N \times K}$ that maximize the likelihood $l(F, H) = \log P(G|F, H)$.

**Solving the optimization problem.** To solve the problem in Eq. 9.3, we adopt a block coordinate ascent approach as we did in Section 7.3. In particular, we update $F_u$ for each $u$ with $H$ fixed and update $H_v$ for each $v$ with $F$ fixed, i.e., we update either incoming or outgoing memberships of one node while fixing the other type of memberships. This approach has the advantage that each subproblem of updating $F_u$ and $H_v$ is convex. For brevity we describe only how to update $F_u$. Updating $H_v$ is analogous. We solve for each $u$:

$$\arg\max_{F_{uc \geq 0}} l(F_u),$$

(9.4)

where

$$l(F_u) = \sum_{v \in N(u)} \log(1 - \exp(-F_u H_v^T)) - \sum_{v \not\in N(u)} F_u H_v^T,$$

where $N(u)$ is a set of neighbors of $u$. To solve this convex problem, we use projected gradient ascent with the following gradient:

$$\nabla l(F_u) = \sum_{v \in N(u)} H_v \frac{\exp(-F_u H_v^T)}{1 - \exp(-F_u H_v^T)} - \sum_{v \not\in N(u)} H_v$$

We compute a step size using backtracking line search. After each update, we project $F_u$.
into a space of nonnegative vectors by setting $F_{uc} = \max(F_{uc}, 0)$.

Naive computation of $\nabla l(F_u)$ takes time $O(|V|)$. As we did in Eq. 7.4, however, we can reduce the complexity to the degree of $u$, $O(|\mathcal{N}(u)|)$, which increases the scalability of CoDA significantly. In particular, we can compute the second term $\sum_{v \notin \mathcal{N}(u)} H_v$ in $O(|\mathcal{N}(u)|)$ by storing $\sum_v H_v$:

$$\sum_{v \notin \mathcal{N}(u)} H_v = (\sum_v H_v - H_u - \sum_{v \in \mathcal{N}(u)} H_v).$$

Similarly, the update rule for $H_v$ can be easily derived by symmetry and takes near-constant time $O(|\mathcal{N}(v)|)$. In practice, we iteratively update $F_u, H_u$ for each $u$ and stop iterating once the likelihood does not increase (by 0.01%) after we update $F_u, H_u$ for all $u$.

**Parameter settings.** For other parameter settings, we follow the same settings as BigCLAM. To determine hard community memberships from the real-valued $\hat{F}, \hat{H}$ that we learn, we threshold $F_{uc}$ and $H_{uc}$ with a constant $\delta = \sqrt{-\log(1 - 1/|V|)}$, i.e., we regard $u$ has an outgoing membership for community $c$ if $F_{uc} \geq \delta$, and an incoming membership for $c$ if $H_{uc} \geq \delta$. To initialize $F, H$, we employ locally minimal neighborhoods. To automatically determine the number of communities $K$, we perform a cross-validation. Refer to Section 7.3 for the further details.

**Parallelization.** We note that our CoDA naturally allows for parallelization. When updating $F_u$ for each node $u$ (Eq. 9.4), each subproblem is separable since all other variables in Eq. 9.4 ($H$) remain fixed. That is, updating the value of $F_u$ for a specific node $u$ does not affect updates to $F_v$ for any other node $v$. In the parallelized version of CoDA, we solve Eq. 9.4 for multiple nodes in parallel. This parallelization does not affect the results of the method. Updating $H_u$ for each node $u$ can be parallelized in the same way.

**CoDA for undirected networks.** So far, we have explained CoDA under the context of directed networks. However, CoDA can easily be applied to undirected networks as well. We make a simple observation: undirected networks model symmetric relationships and thus an undirected relationship is equivalent to two directed relationships. Thus, given an undirected network, we simply convert the network into a directed one by regarding every edge as reciprocal, and then apply CoDA to detect communities.
CHAPTER 9. 2-MODE COMMUNITIES

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Directed</th>
<th>N</th>
<th>E</th>
<th>C</th>
<th>S</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google+</td>
<td>✓</td>
<td>250,469</td>
<td>30,230,905</td>
<td>437</td>
<td>143.51</td>
<td>0.25</td>
</tr>
<tr>
<td>Twitter</td>
<td>✓</td>
<td>125,120</td>
<td>2,248,406</td>
<td>3,140</td>
<td>15.54</td>
<td>0.39</td>
</tr>
<tr>
<td>Facebook</td>
<td>✗</td>
<td>4,089</td>
<td>170,174</td>
<td>193</td>
<td>28.76</td>
<td>1.36</td>
</tr>
<tr>
<td>Enron</td>
<td>✓</td>
<td>45,266</td>
<td>185,172</td>
<td>4,572</td>
<td>63.93</td>
<td>6.46</td>
</tr>
<tr>
<td>LiveJournal</td>
<td>✗</td>
<td>3,997,962</td>
<td>34,681,189</td>
<td>287,512</td>
<td>22.31</td>
<td>1.59</td>
</tr>
<tr>
<td>Youtube</td>
<td>✗</td>
<td>1,134,890</td>
<td>2,987,624</td>
<td>8,385</td>
<td>13.50</td>
<td>0.10</td>
</tr>
</tbody>
</table>

Table 9.1: **Dataset statistics.** Directed: Yes/no, \( N \): number of nodes, \( E \): number of edges, \( C \): number of ground-truth communities, \( S \): average ground-truth community size, \( A \): ground-truth community memberships per node. Further datasets used in this study are described in Table 9.5.

Now, CoDA will easily detect cohesive communities in this converted network as edges in cohesive communities are typically assumed to be reciprocal. Detecting 2-mode communities is also simple. Consider the case where we are given an undirected 2-mode community \( X \) where nodes in group \( A \) are connected to nodes in group \( B \). Once we convert \( X \) into a directed network with reciprocal edges between \( A \) and \( B \), CoDA will estimate two 2-mode communities from this community \( X \): \( \hat{X}_1 \) for edges from \( A \) to \( B \), and \( \hat{X}_2 \) for edges from \( B \) to \( A \). Since both \( \hat{X}_1 \) and \( \hat{X}_2 \) correspond to \( X \), CoDA discovers \( X \) correctly.

### 9.4 Experiments

We evaluate the performance of CoDA and compare it to state-of-the-art community detection methods on a range of directed as well as undirected networks. We measure the quality of community detection by computing the detection accuracy based on gold-standard ground-truth communities. We also evaluate the scalability of the methods as the network size increases.

#### 9.4.1 Dataset Description

We begin by briefly describing the networks that we consider in this chapter. Overall, we consider 5 undirected and 9 directed networks from a wide spectrum of different domains; in particular, we consider social, communication, information, biological and ecological...
Table 9.2: **Performance on Facebook, Google+, and Twitter.** Higher is better. Standard errors are shown in parentheses. The best and second best methods are annotated as ‘1’ and ‘2’. CPM: Clique Percolation Method [118], LC: Link Clustering [4], CoDA, undir: CoDA on the undirected versions of networks, CoDA, dir: CoDA on the directed versions of networks.

<table>
<thead>
<tr>
<th>Method</th>
<th>F1 score Google+</th>
<th>F1 score Twitter</th>
<th>F1 score Facebook</th>
<th>Jaccard similarity Google+</th>
<th>Jaccard similarity Twitter</th>
<th>Jaccard similarity Facebook</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMSB</td>
<td>0.314 (0.011)</td>
<td>0.262 (0.005)</td>
<td>0.379 (0.012)</td>
<td>0.249 (0.020)</td>
<td>0.169 (0.004)</td>
<td>0.266 (0.019)</td>
<td>0.208</td>
</tr>
<tr>
<td>CPM</td>
<td>0.331 (0.036)</td>
<td>0.246 (0.006)</td>
<td>0.429 (0.051)</td>
<td>0.240 (0.032)</td>
<td>0.163 (0.005)</td>
<td>0.342 (0.050)</td>
<td>0.292</td>
</tr>
<tr>
<td>LC</td>
<td>0.304 (0.016)</td>
<td>0.334 (0.003)</td>
<td>0.372 (0.027)</td>
<td>0.226 (0.016)</td>
<td><strong>0.238 (0.003)</strong></td>
<td>0.275 (0.024)</td>
<td>0.291</td>
</tr>
<tr>
<td>BigCLAM</td>
<td>0.334 (0.017)</td>
<td>0.344 (0.005)</td>
<td>0.442 (0.042)</td>
<td>0.217 (0.004)</td>
<td>0.234 (0.004)</td>
<td>0.325 (0.038)</td>
<td>0.315</td>
</tr>
<tr>
<td>DEMON</td>
<td>0.341 (0.029)</td>
<td>0.308 (0.005)</td>
<td>0.418 (0.046)</td>
<td>0.255 (0.027)</td>
<td>0.210 (0.005)</td>
<td>0.311 (0.041)</td>
<td>0.307</td>
</tr>
<tr>
<td>CoDA, undir</td>
<td><strong>0.414 (0.027)</strong></td>
<td><strong>0.348 (0.005)</strong></td>
<td><strong>0.470 (0.042)</strong></td>
<td><strong>0.314 (0.026)</strong></td>
<td>0.237 (0.004)</td>
<td><strong>0.357 (0.039)</strong></td>
<td>0.357</td>
</tr>
<tr>
<td>CoDA, dir</td>
<td><strong>0.406 (0.025)</strong></td>
<td><strong>0.363 (0.005)</strong></td>
<td><strong>0.470 (0.042)</strong></td>
<td><strong>0.334 (0.024)</strong></td>
<td>0.250 (0.004)</td>
<td><strong>0.357 (0.039)</strong></td>
<td>0.360</td>
</tr>
</tbody>
</table>

Networks with ground-truth communities. For the experiments in this section, we consider a subset of 6 publicly available networks where we have explicit ground-truth communities [153]. Table 9.1 shows the statistics of the networks and the ground-truth communities. The networks come from three different domains: The first three networks are the collection of ego-networks from the online social communities of Facebook, Twitter and Google+ [102], the Enron email communication network [79], and LiveJournal and Youtube social networks that we described in Chapter 3. We describe the nature of ground-truth communities in each of these datasets in more detail later.

9.4.2 Experimental Setup

**Baselines.** For comparison we consider the following baseline methods: MMSB (Mixed Membership Stochastic Block Model) [5], which can detect both cohesive and 2-mode communities; Clique Percolation Method, [118] link Clustering [4], BigCLAM [152] [156] are state of the art overlapping cohesive community detection techniques for undirected networks; DEMON [34] is a scalable local community detection method. We use publicly available implementations of each of the above methods.

The above baselines represent the current state-of-the-art in community detection. However, we also considered other baselines, including those that make use of node features [1]We use the publicly available data from the Stanford Large Network Collection: [http://snap.stanford.edu](http://snap.stanford.edu)
network topology \[129\], or both \[11, 102\]; however experiments demonstrate that none of these alternatives outperforms CoDA.

Some methods require input parameters. MMSB requires the number of communities $K$. We use the Bayes information criterion suggested by the authors \[5\] to choose $K$. DEMON requires $\varepsilon$, the threshold value for merging two communities. As there exists no standard criterion for $\varepsilon$, we set $\varepsilon$ so that DEMON detects the same number of communities as CoDA does.

**Evaluation.** We evaluate the performance of the above methods we quantify the degree of correspondence between the ground-truth and the detected communities. To compare a set of ground-truth communities $C^*$ to a set of predicted communities $C$, we adopt an evaluation procedure previously used in Section 6.3, where every detected (ground-truth) community is matched with its most similar ground-truth (detected) counterpart community. More formally, our evaluation function is:

\[
\frac{1}{2|C^*|} \sum_{C_i^* \in C^*} \max_{C_j \in C} \delta(C_i^*, C_j) + \frac{1}{2|C|} \sum_{C_j \in C} \max_{C_i^* \in C^*} \delta(C_i^*, C_j),
\]

where $\delta(C_i^*, C_j)$ is some measure of the similarity between the communities $C_i^*$ and $C_j$. We consider two standard measures of the similarity between sets, namely the F1 score and the Jaccard similarity. Thus, we obtain a value between 0 and 1, where 1 indicates perfect recovery.

### 9.4.3 Discovering Social Circles

We first consider the problem of automatically discovering users’ social circles, a problem previously considered in \[102\]. Circles (or ‘lists’ in Facebook and Twitter) give users a means of categorizing their immediate neighbors, or in the case of directed networks, the users whom they follow. Thus the problem of automatically identifying users’ social circles can be posed as a community detection problem on each user’s ego-network \[102\].

In Table 9.2 we evaluate the performance of CoDA and baselines on social circle detection. Across all three datasets and both evaluation metrics, CoDA (the last row) is the best or second-best performer. On average, CoDA outperforms MMSB by 34%, Clique
Percolation Method by 23%, Link Clustering by 24%, BigCLAM by 14% and DEMON by 17%.

It is important to notice that the 3 data sets possess very different reasons for community (i.e., circle) formation. Facebook is an undirected network and in Facebook circles are driven by dense mutual friendships among users with homogeneous backgrounds [102]; therefore, we would expect cohesive communities in Facebook. Google+ and Twitter are directed networks and there circles are not necessarily based on friendship, because edges in these networks denote follower relationships: The fraction of mutual edges is only 29% in Google+ and 54% in Twitter. For example, a ‘circle’ in Twitter might consist of authors who publish in the same genre, or candidates in the same election. As we will see later in Section 9.5, many detected circles in Google+ and Twitter follow such 2-mode structure.

Nevertheless, despite the different nature of the data sets, CoDA is the best performer in each of them. This result means that CoDA recovers 2-mode circles in Google+ or Twitter as well as cohesive circles in Facebook, i.e., CoDA can detect both kinds of communities more accurately than the baselines.

Directed vs. undirected networks. To further examine the performance out method on directed and undirected networks we perform an experiment with the goal of understanding whether CoDA is still able to recover 2-mode communities even when edge directions are dropped and networks are considered as undirected. To test this, we conduct the following experiment: We convert the directed networks of Twitter and Google+ into undirected ones by removing the edge direction, then apply CoDA (CoDA, undirected, the second to last row in Table 9.2). Surprisingly, CoDA achieves similar performance even without explicit edge directions. This experiment provides evidence that CoDA is capable of accurately finding 2-mode communities even in undirected networks.

9.4.4 Discovering Recipient Lists in Email Networks

We also define a task of email recipient list discovery. Here we also apply CoDA to the problem of automatically discovering recipient lists in the the Enron email communication network [79]. Here the idea is that such lists exhibit a distinct structural pattern in the network and so recipient lists may have 2-mode community structure as a set of users who
CHAPTER 9. 2-MODE COMMUNITIES

<table>
<thead>
<tr>
<th>Method</th>
<th>F1 score</th>
<th>Jaccard similarity</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMSB</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>CPM</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Link Clustering</td>
<td>0.195</td>
<td>0.294</td>
<td>0.245</td>
</tr>
<tr>
<td>BigCLAM</td>
<td>0.478</td>
<td>0.358</td>
<td>0.418</td>
</tr>
<tr>
<td>DEMON</td>
<td>0.464</td>
<td>0.350</td>
<td>0.407</td>
</tr>
<tr>
<td>CoDA, undirected</td>
<td>0.538</td>
<td>0.431</td>
<td>0.485</td>
</tr>
<tr>
<td>CoDA, directed</td>
<td>0.617</td>
<td>0.516</td>
<td>0.567</td>
</tr>
</tbody>
</table>

Table 9.3: **Performance of recipient discovery on the Enron network.** Algorithms that do not scale to the size of the dataset are labeled as “N/A”.

receive the same email may not necessarily email each other [103].

We consider all Enron emails with 20 or more recipients. This gives us a set of 4,572 unique recipient lists in the Enron dataset, which we treat as ground-truth communities (Table 9.1). Now we are given an unlabeled directed Enron email communication network, where an edge $i \rightarrow j$ means that $i$ sent at least one mail to $j$, and the goal is to discover email recipient lists.

We apply CoDA to this network and in Table 9.3 we measure how accurately the communities detected by CoDA correspond to these ground-truth email recipient lists. We report both the F1 score and Jaccard similarity (for methods that do not scale to networks of this size, we report N/A). Table 9.3 shows that CoDA outperforms other methods by a significant margin. CoDA outperforms Link Clustering by 131%, DEMON by 39%, and BigCLAM by 36%.

9.4.5 **Experiments on Large Networks**

We also apply CoDA to two large social networks described in Section 3.3. We consider the LiveJournal and Youtube social networks, and regard user-created groups as ground-truth communities. We ignore groups containing fewer than 10 nodes, yielding 71,093 communities in LiveJournal and 2,078 in Youtube.

Of the baselines previously mentioned, only BigCLAM could scale to both networks and DEMON could scale to the Youtube network. Therefore, we also consider two large-scale graph partitioning methods as baselines for this experiment: Metis [77] and Graclus [38]. For all methods we set the number of communities $K$ to be the number of
Table 9.4: Relative accuracy (compared to the worst performing method) of detected communities on large scale social networks.

<table>
<thead>
<tr>
<th>Method</th>
<th>Relative F\text{1} score</th>
<th>Absolute F\text{1} score</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LiveJournal</td>
<td>Youtube</td>
</tr>
<tr>
<td>Metis</td>
<td>100%</td>
<td>200%</td>
</tr>
<tr>
<td>Graclus</td>
<td>100%</td>
<td>185.7%</td>
</tr>
<tr>
<td>BigCLAM</td>
<td>121.0 %</td>
<td>278.1 %</td>
</tr>
<tr>
<td>DEMON</td>
<td>N/A</td>
<td>100%</td>
</tr>
<tr>
<td>CoDA</td>
<td>129.4%</td>
<td>307.1%</td>
</tr>
</tbody>
</table>

In Table 9.4 shows the results. For this experiment we focus on the score relative to that of the worst-performing baseline in each network (so that the worst-performing baseline has a score of 100%). The reason for focusing on the relative score rather than the absolute score is that the networks are only partially labeled and overall performance is thus artificially low. We find that CoDA outperforms its nearest competitor by 8.4% on LiveJournal and 29% on Youtube.
9.4.6 Scalability

Last, we evaluate the scalability of CoDA by measuring its running time on synthetic networks with increasing size. We generate synthetic networks using the Forest fire model [94] with the forward and backward probabilities set to 0.36 and 0.32 respectively. Since CoDA is easily parallelizable as described in Section 9.3, we also consider a single machine parallel implementation running with 24 threads (CoDA-24).

Scalability results are shown in Figure 9.4. Link Clustering and Clique Percolation Method scale to networks of at most a few thousand nodes. DEMON is a fast and scalable overlapping community detection method. DMEON tends to be faster than CoDA (single-threaded implementation) for small networks up to 100,000 nodes, however, once the network becomes larger, CoDA becomes much faster.

When comparing single-threaded implementations we also note that BigCLAM is the fastest method in our experiments. However, we note that CoDA takes only 30% more time than BigCLAM while it is also solving a more complicated problem, namely detecting cohesive as well as 2-mode communities. Last, we also measure a parallelized version of CoDA (CoDA-24). Using 24 threads on a single machine, we achieve nearly 24x speedup. Ultimately, CoDA takes just 6 minutes to process a 300,000 node network.

9.5 Qualitative Analysis on 2-mode communities and Cohesive Communities

So far we have demonstrated that CoDA can reliably detect both cohesive and 2-mode communities in directed as well as undirected networks.

In this section, we shall demonstrate that 2-mode communities take an important role in networks. We shall use CoDA to perform a qualitative study of various networks in order to determine the extent to which community structures vary across real-world networks from different domains.

Network data. In addition to the datasets already introduced, we also analyze biological networks, foodwebs, web graphs, and citation networks (Table 9.5). For biological networks, we consider the protein-protein interaction network of *Saccharomyces cerevisiae*. 
yeast two-hybrid (PPI-Y2H) and literature-curated (PPI-LC) (Section 6.4). We also consider the Chesapeake and Florida Bay food webs (Section 8.2), the web graph of Stanford University web pages (web-Stanford), the web graph released by Google in 2002 (web-Google) (Section 8.2), and the arXiv citation networks from high-energy physics phenomenology (cit-HepPh) and theory (cit-HepTh) (Section 8.2).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Directed</th>
<th>N</th>
<th>E</th>
<th>T</th>
<th>S</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPI-Y2H</td>
<td>✗</td>
<td>1,647</td>
<td>2,518</td>
<td>40</td>
<td>90.75</td>
<td>2.20</td>
</tr>
<tr>
<td>PPI-LC</td>
<td>✗</td>
<td>1,213</td>
<td>2,556</td>
<td>40</td>
<td>42.08</td>
<td>1.39</td>
</tr>
<tr>
<td>web-Stanford</td>
<td>✓</td>
<td>281k</td>
<td>2,312k</td>
<td>19k</td>
<td>70.63</td>
<td>4.59</td>
</tr>
<tr>
<td>web-Google</td>
<td>✓</td>
<td>875k</td>
<td>5,105k</td>
<td>39k</td>
<td>41.79</td>
<td>1.86</td>
</tr>
<tr>
<td>cit-HepTh</td>
<td>✓</td>
<td>27k</td>
<td>353k</td>
<td>2,000</td>
<td>70.00</td>
<td>5.04</td>
</tr>
<tr>
<td>cit-HepPh</td>
<td>✓</td>
<td>34k</td>
<td>422k</td>
<td>4,976</td>
<td>51.52</td>
<td>7.42</td>
</tr>
<tr>
<td>Florida Bay</td>
<td>✓</td>
<td>121</td>
<td>1,745</td>
<td>6</td>
<td>45.33</td>
<td>2.25</td>
</tr>
<tr>
<td>Chesapeake</td>
<td>✓</td>
<td>33</td>
<td>72</td>
<td>5</td>
<td>9.20</td>
<td>1.39</td>
</tr>
</tbody>
</table>

Table 9.5: Dataset statistics. Directed: Whether the network is directed or not, N: number of nodes, E: number of edges, T: number of detected communities, S: average size of detected communities, R: community memberships per node.

Figure 9.5: Two detected communities in a Foodweb (Chesapeake Bay). Among other communities, CoDA identifies sets of nodes with similar predators (A, blue nodes) and with similar prey (B, red nodes), both of which have low internal connectivity.
9.5.1 Biological and Foodweb Communities

We first present 2-mode communities in food webs, where an edge from a node $u$ to $v$ means that $u$ is preyed upon by $v$. We apply CoDA on the Chesapeake Bay food web network shown in Figure 9.5 and display an induced subgraph of detected 2-mode communities in Figure 9.6.

In food web networks, we find 2-mode communities of groups of predators who rely on similar groups of prey (Figure 9.6). The blue 2-mode community ($B-D$) represents predators and prey in the the Chesapeake Bay sands: *nereis, macoma spp.*, and *mya arenaria* (in $B$) are small, sand-dwelling clams or worms that are fed on by fish (in $D$). Alternately, the red community ($A-C$) shows predator-prey relationships among fish: small fish ($A$) are eaten by bigger fish ($C$). CoDA also discovers the overlap between two predator groups where *white perch* and *spot* prey on both fish and clams.

CoDA allows us to gain insights into biological PPI networks. In particular, 2-mode communities can also be frequent in undirected networks where relations are modeled as

\[ \text{http://snap.stanford.edu} \]
Figure 9.7: Overlapping 2-Mode communities detected by our method in a Protein-Protein interaction network. See main text for the explanation of community structure.

CoDA discovers 2-mode communities in the undirected protein-protein interaction network determined by yeast two-hybrid screening (PPI-Y2H).

Figure 9.7 displays the induced subgraph of two communities that CoDA detects. 2-mode communities detected by CoDA clearly reveal the interaction between different protein groups. For example, proteins in group $C$ of Figure 9.7 heavily interact with proteins in group $A$, even though these proteins do not interact within the same group (with $A$ or within $C$).

To further analyze the role of these communities, we used gene ontologies to identify relevant terms/functions of proteins in $A$, $B$, $C$, and $D$ using the GO Term Finder [20]. The proteins in the large groups ($C$, $D$) are generally associated with catalytic activity and ion binding ($p$-value $\sim 10^{-4}$); however, they are regulated by different protein groups ($A$, $B$) which have different functions.

Proteins in $A$ (e.g., YLR347C and YNL189W) are protein transporters, whereas proteins in $B$ (e.g., YLR291C) are regulators. Perhaps more interestingly, YPL070W belongs to both $A$ and $B$ and regulates both $C$ and $D$. However, its role is not yet known. But based on known functions of proteins in groups $A$ and $B$ we can extrapolate the function of YPL070W. This example shows how network analysis and community detection in particular can provide research directions for experimental biology [24].
9.5.2 2-mode vs. Cohesive Communities

Since CoDA can detect both cohesive and 2-mode communities, we can use it to measure the extent to which real network data exhibits cohesive and 2-mode behavior. This analysis allows us to characterize the mesoscale structure of real-world networks as the proportion of 2-mode versus cohesive communities can be used as a ‘signature’ of the community structure in the network.

**Experimental setup.** For this experiment, we consider 12 networks that come from 6 domains to characterize their different mesoscale structures. We consider ego networks (Twitter, Google+) and social networks (LiveJournal, Youtube) from Section 9.4. We also include 8 networks from 4 different domains: Biological networks, web graphs, food webs, and citation networks among research papers (Table 9.5).

To classify whether a detected community is 2-mode or cohesive, we measure the Jaccard similarity \( J(c) = \frac{|O(c) \cap I(c)|}{|O(c) \cup I(c)|} \) between the set of member nodes with outgoing memberships \( O(c) \), and the set of member nodes with incoming memberships \( I(c) \). In a completely cohesive community, this Jaccard similarity is 1 because two sets of members are identical, whereas it is 0 in a completely 2-mode community. We regard a community \( c \) as 2-mode if \( J(c) \) is lower than some threshold \( \gamma \) or as cohesive otherwise \( (J(c) \geq \gamma) \). Here, we show the case with \( \gamma = 0.2 \) as this setting gives the most interpretable results.

**Experimental results.** Figure 9.8 shows the fraction of 2-mode and cohesive communities in 12 networks from different domains. Ego networks (Twitter and Google+) exhibit a relatively high fraction of cohesive communities. As noted earlier Facebook ego networks (not shown) have an even higher fraction (over 95%) of cohesive communities. This result
is in line with [34] where the authors show that Facebook ego networks can be easily divided into cohesive communities. However, it is important to note that a significant fraction of Twitter (20%) and Google+ (30%) communities exhibit 2-mode structure.

Literature-curated protein-protein interaction networks (PPI-LC) exhibit many cohesive communities. On the other hand, in PPI networks generated based on yeast two-hybrid screening (PPI-Y2H) about 50% of the communities are 2-mode. This difference is consistent with a previous study on PPI networks [161], which provided the following explanation: edges of PPI-LC are extracted from scientific papers that report experimentally validated interactions. However, current biological experiments have mainly been guided by research on cohesive communities and thus it seems as though most interactions that have been explored take place in “cohesive” communities [161]. On the other hand, the PPI-Y2H network is created by an automatic process and more faithfully represents the interaction network. In this case many 2-mode communities emerge [161].

In social networks we find interesting results. In LiveJournal, communities are more cohesive, which can be explained by the fact that edges in LiveJournal indicate ‘friendships’ (e.g., sharing private blog content). Alternately, Youtube communities are predominantly 2-mode. Youtube differs from other social networks in one important way: edges in Youtube are essentially ‘subscriptions’ for content rather than mutual friendships; consequently, high degree nodes tend to connect to low degree nodes [107].

We also consider web graphs, which are of interest because Kumar et al. [83] used the existence of 2-mode communities as indicators or signatures for cohesive communities. Our results nicely suggest the co-existence of cohesive communities and 2-mode communities by showing that web graphs have an equal proportion of 2-mode and cohesive communities.

Finally, food webs as well as citation networks consist almost entirely of 2-mode communities. These results are natural as reciprocal relationships are extremely unlikely in these networks. In food webs, for example, few species prey upon each other. Citation networks are directed acyclic graphs and reciprocal citation is impossible by definition. Intuitively, cohesive communities in directed networks contain some number of bidirectional edges among their members, therefore a lack of such reciprocal edges naturally leads to the dominance of 2-mode communities, as we observe in Fig. 9.6.
9.6 Conclusion

An accurate notion of a community is critical when studying the mesoscale structure of networks. Traditional models consider ‘communities’ to be sets of densely connected nodes. In addition, here we also consider 2-mode communities, which are groups of nodes who have similar external links, even though they may be sparsely connected internally.

By extending our AGM and BigCLAM, in this chapter we presented CoDA, a community detection method which naturally detects both densely connected and 2-mode communities. Unlike BigCLAM, CoDA handles both directed and undirected networks, while maintaining the ability to capture overlapping and hierarchical structure among communities. Our experimental findings reveal that CoDA outperforms the current state-of-the-art in detecting ground-truth communities, and that CoDA reveals how 2-mode and cohesive communities co-exist in real networks. Surprisingly, all these additional features were made possible by making a very simple modification to BigCLAM.

The versatility of CoDA to detect both cohesive and 2-mode communities accurately in directed and undirected networks raises many interesting avenues of future work. For example, understanding the interaction between 2-mode communities and cohesive communities is a fruitful direction. Inferring the role of nodes from their community affiliations would be useful.

In the next chapter, we present another extension of AGM and BigCLAM where we detect communities using not only a network but also node attributes.
Chapter 10

Community Detection in Networks with Node Attributes

10.1 Introduction

Our main task in this thesis is network community detection, where the input is a network and the goal is to infer the community memberships of the nodes. To detect community memberships of the nodes, we rely on the network connectivity structure. In this chapter, our goal is to extend AGM and BigCLAM so that we can incorporate the information for the node attributes as well as the network connectivity for community detection.

The main reason why we use the network structure for community detection is the intuition that the network connectivity structure organizes around the community affiliations. We can naturally develop a similar intuition for the node attributes; nodes in communities would share certain attributes. For example, known properties of proteins, users’ social network profiles, or authors’ publication histories may tell us which objects are similar, and to which communities or modules they may belong. Thus, there are two sources of data that can be used to infer community affiliations: network connectivity and node attributes.

However, many existing methods for community detection typically focus only one of these two data modalities. Most community detection algorithms (including our AGM, BigCLAM, and CoDA) aim to find communities based on the network structure [50] [150] yet they typically ignore node attributes. In terms of attributes, clustering algorithms [15]...
identify sets of objects whose attributes are similar, while ignoring relationships between objects.

By considering only one of these two sources of information independently, an algorithm may fail to account for important structure in the data. For example, attributes might tell us to which community a node with very few links belongs to; this would be difficult to determine from the network structure alone. Conversely, the network might tell us that two objects belong to the same community, even if one of them has no attribute information. Thus, it is important to consider both sources of information together and consider network communities as sets of nodes that are densely connected, but which also share some common attributes. Node attributes can complement the network structure, leading to more precise detection of communities; additionally, if one source of information is missing or noisy, the other can make up for it. However, considering both node attributes and network topology for community detection is also challenging, as one has to combine two very different modalities of information.

Only recently have approaches for detecting communities based on both sources of information been developed [11, 102] (Table 10.1). Many existing methods that combine network and node attribute information use single-assignment clustering [6, 41, 111, 130, 165]; however, the applicability of these methods is limited, as they cannot detect overlapping communities. Approaches based on topic models [11, 100, 138, 151] allow overlapping communities to be detected. However, they assume “soft” node-community memberships, which are not appropriate for modeling communities because they do not allow a node to have high membership strength to multiple communities simultaneously (i.e., like Mixed Membership Block model in Section 4.3). Finally, all existing methods are only able to handle relatively small networks: the networks typically analyzed consist only of thousands of nodes [29, 100, 102, 138].

**Present work:** Community detection in networks with node attributes. Here, we develop a high-performance (accurate and scalable) overlapping community detection method for networks with node attribute information. We build on our BigCLAM and develop Communities from Edge Structure and Node Attributes (CESNA), which is based on a generative model for networks with node attributes. Our model advances existing approaches (summarized in Table 10.1) by making several innovations that ultimately lead to better
CHAPTER 10. COMMUNITY DETECTION WITH NODE ATTRIBUTES

<table>
<thead>
<tr>
<th>Method class</th>
<th>O</th>
<th>H</th>
<th>D</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heuristics</td>
<td></td>
<td></td>
<td>X</td>
<td>100,000</td>
</tr>
<tr>
<td>LDA-based</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>85,000</td>
</tr>
<tr>
<td>Clique-based heuristics</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>100,000</td>
</tr>
<tr>
<td>Social circles</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>5,000</td>
</tr>
<tr>
<td>CESNA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1,000,000</td>
</tr>
</tbody>
</table>

Table 10.1: Methods for community detection in networks with node attributes. O: Detects overlapping communities?, H: Assigns hard node-community memberships?, D: Allows for dependence between the network and the node attributes? (Fig. 10.1), N: Largest network that can be processed in 10 hours (Fig. 10.4). Refer to Sec. 10.6 for further details.

performance both in terms of accuracy as well as scalability. First, our model allows us to detect overlapping communities by employing hard node-community memberships. This way, we can avoid the assumption of soft-membership methods that nodes sharing multiple common communities are less likely to be connected (Section 4.3). Second, in contrast to a line of previous work [62, 102], which assumed that communities and attributes are marginally independent, we assume that communities “generate” both the network as well as attributes (Figure 10.1). This way we allow for dependence between the network and the attributes. Third, to fit the model and thus discover communities, we develop a block-coordinate ascent method like BigCLAM where we can update all model parameters in time linear in the number of edges in the network (Chapter 7). This approach makes our method scale to networks an order of magnitude larger than what was possible by previous methods.

To the best our knowledge, CESNA is the first overlapping community detection method that models both hard node-community memberships and the dependency between the communities and attributes. Moreover, CESNA can detect overlapping, non-overlapping, as well as hierarchically nested communities in networks, while considering both node attributes and graph structure.

We evaluate CESNA on six online social, information, and content-sharing networks: Facebook, Google+, Twitter, Wikipedia, and Flickr. We quantify CESNA’s accuracy in
detecting communities by comparing its predictions to hand-labeled ground-truth communities. We compare CESNA to state-of-the-art community detection methods, including those that detect communities based only on the network structure, methods based only on node attributes, and methods that model both network structure and attributes jointly. Overall, CESNA achieves a 47% improvement in the accuracy of detected communities over the baselines we consider. We also examine whether node attributes can boost the performance of community detection algorithms in cases where the network is noisy or not fully observed. We add noise to the network and we find that the performance gap between CESNA over competing methods increases as the network structure becomes noisier and therefore less reliable. This means that CESNA is able to successfully leverage node attributes to compensate for missing or noisy information in the network structure.

To quantify the scalability of CESNA we measure its running time on synthetic networks of increasing size. Compared to existing methods, the size of networks that CESNA can process far exceeds the current state-of-the-art: CESNA can handle networks 100 times larger than LDA-based methods [11] given the same runtime budget. Even when compared to methods that consider only the network structure (i.e., which handle strictly less information), CESNA is faster than most baselines.

Last, we also inspect communities detected by CESNA on Facebook networks, and on a network of Wikipedia articles about famous philosophers. We find that, on Facebook data, CESNA automatically learns that education-based attributes (“School name” or “Major”) are very highly correlated with a communities, whereas other people’s attributes, such as “Work start date” and “Work end date” are not related to community structure. On philosophers data, CESNA learns natural attributes for communities: e.g., subjects about Islamic culture are associated with a community of Islamic philosophers. While methods that ignore node attributes assign very influential philosophers (e.g., Aristotle) to most communities, CESNA circumvents this issue by modeling attributes, and discovering that Aristotle, while well connected to many philosophers, does not share common attributes with all of them.
Figure 10.1: Two ways of modeling the statistical relationship between a graph $G$, attributes $X$, and communities $F$. Circles represent latent variables that need to be inferred and squares represent manifest (observed) variables.

10.1.1 Outline of Chapter

Here is the outline of the chapter:

- We present a generative model for the network structure and the node attributes. Our intuition is that the community affiliations of the nodes generate the network connectivity and the node attributes. (Section 10.2)

- We develop CESNA, a novel community detection method for networks with node attributes. Given a network and the node attributes, we fit our generative model to the data. We use a block-coordinate gradient ascent approach which is similar to BigCLAM. (Section 10.3)

- We quantitatively evaluate the performance of CESNA by performing several experiments. CESNA detects network communities more accurately than the previous methods that include those using only network, those using only the node attributes, and those using both network and the attributes. CESNA is robust when the network edges are partially observed, and CESNA is far more scalable than the previous methods that consider both network and the node attributes. (Section 10.4)

- Finally, we qualitatively analyze the communities detected by CESNA. CESNA allows us to identify which attributes are associated with each community. (Section 9.5)
10.2 Models for Communities in Networks with Node Attributes

Here, we develop a probabilistic model that combines community memberships, the network topology, and node attributes. We present the Communities from Edge Structure and Node Attributes (CESNA), a probabilistic generative model for networks and node attributes that satisfies the desiderata mentioned above. Our model is based on the following intuitive properties:

- Nodes that belong to the same communities are likely to be connected to each other.
- Communities can overlap, as individual nodes may belong to multiple communities.
- If two nodes belong to multiple common communities, they are more likely to be connected than if they share only a single common community (Our observation in Chapter 4).
- Nodes in the same community are likely to share common attributes — for example, a community might consist of friends attending a same school.

We formally describe the generative process of CESNA as follows. We assume that there are $N$ nodes in the network $G$, each node has $K$ attributes, and there are $T$ communities in total. We denote the network by $G$, the node attributes by $X$ ($X_{uk}$ is $k$-th attribute of node $u$), and community memberships by $F$. For community memberships $F$, we assume that each node $u$ has a non-negative affiliation weight $F_{uc} \in [0, \infty)$ to community $c$. ($F_{uc} = 0$ means that node $u$ does not belong to community $c$.)

We shall now proceed by describing these components of the model in further detail.

Modeling the links of the network. To model a network, we employ the generative process of BigCLAM (Chapter 7), where the graph $G(V, E)$ arises from node community memberships $F$. In particular, we assume that two member nodes $u, v$ are connected with the following probability (Eq. 7.1):

$$P_{uv} = P(u, v) = 1 - \exp(- \sum_c F_{uc} \cdot F_{vc}).$$
In other words, we assume the following generative process for each entry $A_{uv} \in \{0, 1\}$ of the network’s adjacency matrix:

$$P_{uv} = 1 - \exp(-\sum_{c} F_{uc} \cdot F_{vc}),$$

$$A_{uv} \sim \text{Bernoulli}(P_{uv}).$$

(10.1)

Note that the above generative process satisfies our three aforementioned requirements. The network edges are created due to shared community memberships (Requirement (1)). Furthermore, each membership $F_{uc}$ of a node $u$ is regarded as an independent variable to allow a node to belong to multiple communities simultaneously (Requirement (2)). This is in stark contrast to “soft-membership” models (such as Mixed Membership Stochastic Block Model [5]), which add constraints $\sum_{c} F_{uc} = 1$ so that $F_{uc}$ is a probability that a node $u$ belongs to a particular community. Finally, because each community $c$ generates connections between its members independently, nodes belonging to multiple common communities have a higher probability of connecting than if they share just a single community (Requirement (3)).

**Modeling node attributes.** Just as community affiliations can be used to model network edges, they can also be used to model node attributes. We next describe how node attributes are generated from community memberships.

We assume binary-valued attributes where for each attribute $X_{uk}$ of a node $u$, we consider a separate logistic model. Our intuition is that, based on a node’s community memberships, we should be able to predict the value of each of the node’s attribute values. Thus, we regard group memberships $F_{u1}, \ldots, F_{uC}$ as input features of the logistic model with the associated logistic weight factor $W_{kc}$ (for each attribute $k$ and community $c$). We also add an intercept term $F_{u(C+1)} = 1$ to the input feature of each node $u$:

$$Q_{uk} = \frac{1}{1 + \exp(-\sum_{c} W_{kc} \cdot F_{uc})},$$

$$X_{uk} \sim \text{Bernoulli}(Q_{uk}).$$

(10.2)

where $W_{kc}$ is a real-valued logistic model parameter for community $c$ to the $k$-th node attribute and $W_{k(C+1)}$ is a bias term. The value of $W_{kc}$ represents the relevance of each
CHAPTER 10. COMMUNITY DETECTION WITH NODE ATTRIBUTES

Figure 10.2: Plate representation of CESNA. $X_{uk}$: $k$-th attribute of node $u$; $W_k$: Logistic weight vector for attribute $k$; $Q_{uk}$: Probability that $X_{uk} = 1$; $F_{uc}$: Membership strength of node $u$ to community $c$; $A_{uv}$: Indicator for whether the nodes $u$ and $v$ are connected; $P_{uv}$: Probability that $A_{uv} = 1$.

Last, we also note that depending on the type of attribute, there are also other choices for modeling attributes $X$ based on $F$. For example, for real-valued attributes linear regression could be used. Also, note that we assume that the number of attributes is relatively small compared to the number of nodes; as such, we can use a separate logistic model for each attribute.
Chapter 10. Community Detection with Node Attributes

10.3 Community Detection with CESNA

Next, we shall describe how we detect network communities by estimating CESNA model parameters from given data. We are given an undirected graph \( G(V, E) \) with binary node attributes \( X \). We aim to detect \( T \) communities as well as the relation between communities and attributes. For now, we shall assume the number of communities \( T \) is given. Later, we will describe how to automatically estimate \( T \).

We aim to infer the values of latent variables \( F \) and \( W \) based on the observed network and the attributes. This means we need to estimate \( N \cdot C \) community memberships (i.e., \( \hat{F} \in \mathbb{R}^{N \times C} \)), and \( K \cdot (C + 1) \) logistic weight parameters (i.e., \( \hat{W} \in \mathbb{R}^{K \times (C+1)} \)).

We find the optimal \( \hat{F} \) and \( \hat{W} \) by maximizing the likelihood \( l(F, W) = \log P(G, X|F, W) \) of the observed data \( G, X \):

\[
\hat{F}, \hat{W} = \arg\max_{F \geq 0, W} \log P(G, X|F, W). \tag{10.3}
\]

Because \( G \) and \( X \) are conditionally independent given \( F \) and \( W \), we can decompose the log-likelihood \( \log P(G, X|F, W) \) as follows:

\[
\log P(G, X|F, W) = \mathcal{L}_G + \mathcal{L}_X
\]

where \( \mathcal{L}_G = \log P(G|F) \) and \( \mathcal{L}_X = \log P(X|F, W) \). We compute \( \mathcal{L}_G \) and \( \mathcal{L}_X \) simply using Equations 10.1 and 10.2.

\[
\mathcal{L}_G = \sum_{(u,v) \in E} \log(1 - \exp(-F_u F_v^T)) - \sum_{(u,v) \notin E} F_u F_v^T
\]

\[
\mathcal{L}_X = \sum_{u,k} (X_{uk} \log Q_{uk} + (1 - X_{uk}) \log(1 - Q_{uk}))
\]

where \( F_u \) is a vector \( \{F_{uc}\} \) for node \( u \) and \( Q_{uk} \) is defined in Equation 10.2.

Last, we also invoke \( l_1 \)-regularization on \( W \) to avoid overfitting and to learn sparse relationships between communities and attributes. Thus, our optimization problem that we
aim to solve is:

\[ \hat{F}, \hat{W} = \arg\max_{F \geq 0, W} \mathcal{L}_G + \mathcal{L}_X - \lambda \|W\|_1, \]  

(10.4)

where \( \lambda \) is a regularization hyperparameter.

To solve the problem in Eq. (10.4) we adopt a block coordinate ascent approach. We update \( F_u \) for each node \( u \) by fixing both \( W \) and the community membership \( F_v \) of all other nodes \( v \). After updating \( F_u \) for all nodes, we then update \( W \) while fixing community memberships \( F \). This way, we can decompose the non-convex optimization problem of Eq. (10.4) into a set of convex subproblems. We describe our solution to each of these subproblems next.

Updating community memberships. To update community memberships, we build on the optimization procedure used in BigCLAM (Chapter 7). However, we modify the procedure to consider node attributes (which BigCLAM ignores). We update the membership \( F_u \) of an individual node \( u \) while fixing all other parameters (the membership \( F_v \) of all other nodes, and logistic model parameters \( W \)).

We solve the following subproblem for each \( u \):

\[ \hat{F}_u = \arg\max_{F_{uc} \geq 0} \mathcal{L}_G(F_u) + \mathcal{L}_X(F_u), \]  

(10.5)

where \( \mathcal{L}_G(F_u) \) and \( \mathcal{L}_X(F_u) \) are the parts of \( \mathcal{L}_G, \mathcal{L}_X \) involving \( F_u \), i.e.,

\[
\mathcal{L}_G(F_u) = \sum_{v \in \mathcal{N}(u)} \log(1 - \exp(-F_u F_v^T)) - \sum_{v \notin \mathcal{N}(u)} F_u F_v^T \\
\mathcal{L}_X(F_u) = \sum_k (X_{uk} \log Q_{uk} + (1 - X_{uk}) \log(1 - Q_{uk}))
\]

where \( \mathcal{N}(u) \) is a set of neighbors of \( u \). Note that this problem is convex: \( \mathcal{L}_G(F_u) \) is a concave function of \( F_u \) [110 156] and \( \mathcal{L}_X(F_u) \) is a logistic function of \( F_{uc} \) when \( W \) is fixed.

To solve this convex problem, we use projected gradient ascent. The gradient can be
computed straightforwardly:
\[
\frac{\partial \mathcal{L}_G(F_u)}{\partial F_u} = \sum_{v \in \mathcal{N}(u)} F_{vc} \frac{\exp(-F_u F_v^T)}{1 - \exp(-F_u F_v^T)} - \sum_{v \notin \mathcal{N}(u)} F_{vc}
\]
\[
\frac{\partial \mathcal{L}_X(F_u)}{\partial F_u} = \sum_k (X_{uk} - Q_{uk}) W_{kc}.
\]

We then update each \(F_{uc}\) by gradient ascent and then project onto a space of non-negative real numbers \([0, \infty)\):

\[
F_{uc}^{\text{new}} = \max(0, F_{uc}^{\text{old}} + \alpha \left( \frac{\partial \mathcal{L}_G(F_u)}{\partial F_u} + \frac{\partial \mathcal{L}_X(F_u)}{\partial F_u} \right))
\]  
(10.6)

where \(\alpha\) is a learning rate which we set using backtracking line search [19].

**Updating logistic parameters.** We update parameters \(W\) of the logistic model by keeping community memberships \(F\) fixed. To compute this, we first notice that we can ignore \(\mathcal{L}_G\) in Eq. [10.4] as \(G\) does not depend on \(W\). Next, we also include \(l_1\)-regularization on \(W\), as we aim to learn sparse relationships between community memberships and node attributes:

\[
\hat{W} = \arg\max_W \sum_{u,k} \log P(X_{uk} | F, W) - \lambda |W|_1.
\]

Furthermore, as we employ an independent logistic model for each attribute, we only need to consider the \(k\)-th attribute when updating the weight vector \(W_k\):

\[
\arg\max_{W_k} \sum_u \log P(X_{uk} | F, W_k) - \lambda |W_k|_1.
\]  
(10.7)

Note that this is \(l_1\)-regularized logistic regression with input features \(F\) and output variable \(X\). Again, we simply apply a gradient ascent method:

\[
\frac{\partial \log P(X_{uk} | F, W_k)}{\partial W_{kc}} = (X_{uk} - Q_{uk}) F_{uc},
\]

\[
W_{kc}^{\text{new}} = W_{kc}^{\text{old}} + \alpha \left( \sum_u \frac{\partial \log P(X_{uk} | F, W_k)}{\partial W_{kc}} - \lambda \cdot \text{Sign}(W_{kc}) \right),
\]
where $\alpha$ is a step size as in Eq. 10.6.

Now, we iteratively update $F_u$ for each $u$ and then update $W_k$ for each attribute $k$. We stop iterating once the likelihood does not increase (by at least 0.001%) after a full iteration over all $F_u$ and all $W_k$.

**Determining community memberships.** After learning real-valued community affiliations $\hat{F}$, we need to determine whether node $u$ belongs to community $c$. To do so, we employ the same strategy as we did in Section 7.3. We regard $u$ as belonging to $c$ only if the corresponding $F_{uc}$ is above the threshold $\delta = \sqrt{-\log(1 - 1/N)}$.

**Computational complexity of CESNA.** We next analyze the computational complexity of CESNA. In particular, we show that a full iteration of CESNA takes time linear in the number of edges and attributes.

For simplicity, let us assume a single community $C = 1$. By using the same technique as in Eq. 7.4, we can compute $\frac{\partial L_G(F_u)}{\partial F_u}$ in $O(|\mathcal{N}(u)|)$.

$$
\sum_{v \in \mathcal{N}(u)} F_{ve} = \left( \sum_{v} F_{ve} - F_{ue} - \sum_{v \in \mathcal{N}(u)} F_{ve} \right).
$$

Therefore, updating $F_u$ for all nodes $u$ takes $O(|E| + NK)$ operations. Because updating $W_k$ takes just $O(N)$ for each $k$, a full iteration of CESNA takes $O(|E| + NK)$ operations, which is linear in the number of edges, nodes and the number of attributes.

Similar to BigCLAM and CoDA, CESNA also nicely lends itself to parallelization. In particular, updating $W_k$ naturally allows for parallelization, as we can update $W_k$ for multiple attributes $k$ simultaneously. Because $F$ is fixed, the problems in Eq. 10.7 are independent for different attributes $k$. We also update $F_u$ for multiple nodes $u$ in parallel, like what we did with BigCLAM. As we show in the next section, parallelization on a single shared memory machine boosts the speed of CESNA by a factor of 20 (the number of threads).


**CESNA hyperparameter settings.** In CESNA, the overall model likelihood is a combination of the network likelihood $L_G$ and the likelihood of node attributes $L_X$ (Eq. 10.4). As
the two likelihoods can have vastly different ranges we scale them using the parameter \( \alpha \). In particular, we introduce a hyperparameter \( \alpha \) that controls the scaling between the two likelihoods:

\[
\arg\max_{F \geq 0, W} (1 - \alpha) \mathcal{L}_G + \alpha \mathcal{L}_X - \lambda |W|_1.
\]

We choose values of hyperparameters \( \alpha \) and \( \lambda \) among \( \alpha \in \{0.25, 0.5, 0.75\}, \lambda \in \{0.1, 1.0\} \) based on the held-out data likelihood (i.e., by cross-validation). We note that the performance of CESNA does not change much with the values of hyperparameters. Setting \( \alpha = 0.5 \) (i.e., the unscaled version of Eq. 10.4) and \( \lambda = 1 \) gives reliable performances in most cases.

For other settings, we follow the procedure in BigCLAM (Section 7.3). To automatically find the number of communities \( T \), we use cross-validation. To initialize \( F \), we use locally minimal neighborhoods [54]. We use a zero-valued matrix to initialize \( W \). Refer to Section 7.3 for the further details.

### 10.4 Experimental Evaluations

We quantify the performance of CESNA by comparing it to state-of-the-art community detection methods in various social and information networks. We evaluate the performance of the methods by evaluating the accuracy of the detected communities when compared to the gold-standard, ground-truth communities. We also evaluate the scalability by measuring the running time as the network size grows.

**Dataset description.** For our evaluation, we consider five datasets where we have network information as well as node attributes. In addition to networks and attributes, we also have access to explicit *ground-truth* community labels. The availability of such ground-truth allows us to evaluate community detection methods by quantifying the degree of agreement between the detect and the ground-truth communities [130]. Table 10.2 lists the networks and their properties.

The networks come from 3 different domains: information network among Wikipedia articles (philosophers) [4], content-sharing network (Flickr) [130], and the three ego-networks that we considered in Chapter 9 (Facebook, Google+, and Twitter) [102]. We next describe
each of these networks in further detail.

The philosophers network \( [4] \) (Section 6.5) consists of Wikipedia articles about famous philosophers. Nodes represent Wikipedia articles about philosophers, and undirected edges indicate whether one article links to another. For the attributes of each node \( u \), we use a binary indicator vector of out-links from node \( u \) to other non-philosopher Wikipedia articles. For example, we regard a link to a Wikipedia article “Edinburgh” as a binary attribute “Edinburgh.” We consider 5,770 attributes, to which at least five philosophers have a link. Moreover, Wikipedia also provides categories (e.g., “Muslim philosophers”, or “Early modern philosophers”) for each article. We regard each category with more than five philosophers as a ground-truth community.

The Flickr image sharing network \( [130] \) consists of nodes which represent Flickr users, and edges indicate follow relations between users. We use tags of images uploaded by a given user as her attributes. In this network, the ground-truth communities are defined as user-created interest-based groups that have more than five members.

The last three networks (Facebook, Google+, and Twitter) are ego-networks that we considered in Chapter 9. We use the same way to define ground-truth communities as we did in Chapter 9. However, the key difference here is that we consider the node attributes as well as the underlying social network, whereas we consider only the network in Chapter 9. In Facebook and Google+, node attributes come from user profiles, such as gender, job titles, institutions, and so on. In Twitter, node attributes are defined by hashtags used by the user in her tweets. To reduce the dimensionality of the node attributes, we discard any attribute which the owner of the ego-network does not possess.

Table 10.2: Dataset statistics. \( N \): number of nodes, \( E \): number of edges, \( T \): number of communities, \( K \): Number of node attributes, \( S \): average community size, \( R \): community memberships per node.
**Baselines for comparison.** We consider the three classes of baseline community detection methods: (1) methods that use only the network structure, (2) methods that use only node attributes, and (3) methods that combine both.

The first class of baselines considers only the network, ignoring node attributes altogether: *Demon* [34] and *BigCLAM* [156] (Chapter 7) are state-of-the-art overlapping community detection methods.

Second is a class of baselines that focuses on node attributes without considering the network structure. Here, we use *Multi Assignment Clustering (MAC)* [52], which detects overlapping communities based on node attributes alone.

The third class of baselines we consider combines the network structure with node attributes. For this class, we choose three state-of-the-art methods. Based on Table 10.1 we select one algorithm from each model type: *Block-LDA* [11] represents soft-membership approaches, while the *CODICIL* [130] represents heuristics for non-overlapping communities, and the *EDCAR* [62] represents heuristics for finding dense subgraphs. Finally, we consider the *ML '12* [102] method, which represents overlapping hard-membership approaches.

For all baselines, we use implementations provided by the authors. All baselines except CODICIL require a user to specify the number of communities to detect. We set this parameter so that each model detects the same number of communities as CESNA. CODICIL and EDCAR also has other input parameters, for which we used default values provided by the authors.

**Evaluation metrics.** We quantify the performance in terms of the agreement between the ground-truth communities and the detected communities. To compare a set of ground-truth communities \( C^* \) to a set of detected communities \( C \), we adopt an evaluation procedure previously used in Section 7.4 and Section 9.4

\[
\frac{1}{2|C^*|} \sum_{C^*_i \in C^*} \max_{C_j \in C} \delta(C^*_i, C_j) + \frac{1}{2|C|} \sum_{C_j \in C} \max_{C^*_i \in C^*} \delta(C^*_i, C_j),
\]

where \( \delta(C^*_i, C_j) \) is some similarity measure between the communities \( C^*_i \) and \( C_j \). We consider two standard metrics \( \delta(\cdot, \cdot) \) for quantifying the similarity between a pair of sets, namely the F1 score and the Jaccard similarity. Thus, for each method, we obtain a score
CHAPTER 10. COMMUNITY DETECTION WITH NODE ATTRIBUTES

Table 10.3: **Performance of methods on five datasets.** *Info* indicates the information used by a given method (Network, Attributes, or Both). Best performing models are bolded. Symbol * indicates that CESNA outperforms a given baseline by 95% statistical confidence. Overall, CESNA statistically significantly outperforms all considered methods.

Experiments on recovering ground-truth communities. We evaluate the performance of CESNA and baselines on our five datasets. Table [10.3] shows the results where “N/A” means that the method cannot scale to a given network. We make several observations.

Comparing CESNA to methods without the node attributes (Demon and BigCLAM), we notice that CESNA achieves better performance, as it combines the information from the node attributes as well as the network. Similarly, CESNA also outperforms MAC, which only focuses on node attributes. In particular, CESNA never performs worse than state-of-the-art methods that use only a single source of data. The strong performance of CESNA is not obvious, as it would be entirely possible that combining two sources of data would confuse the algorithm and degrade the overall performance (in fact, notice that BigCLAM, which uses only the network structure, indeed outperforms most of the methods that use both sources of information). Thus, we believe that the strong performance of CESNA as an indication that CESNA combines the best ingredients from both worlds.

When comparing the performance of CESNA to methods that consider both the network structure and node attributes (CODICIL, Block-LDA, and ML ’12), we again observe the strong performance of CESNA. On average, CESNA gives 47% relative improvement in the accuracy of detected communities over methods that consider both sources of information.

We also note that CESNA shows a bigger margin in performance against the baselines in an information network such as the philosophers dataset, or a content-sharing network like Flickr than in social networks. In the philosophers network, for example, CESNA achieves a 14% relative gain in the F1 score and 15% in the Jaccard similarity compared...
to the best baseline. A possible explanation for this phenomenon is that in content-sharing and information networks, the properties/content of the nodes plays a much bigger role in link formation.

Overall, we note that across all datasets and evaluation metrics, CESNA yields the best performance in 8 out of 10 cases. In terms of average performance, CESNA outperforms Demon by 20%, BigCLAM by 6%, MAC by 112%, Block-LDA by 58%, CODICIL by 29%, EDCAR by 57%, and ML ’12 by 54%.

Last, we also measure the statistical significance of performance differences of CESNA and the baselines. For each baseline’s performance on each data set, we compute the statistical significance of CESNA outperforming the baseline using a one-sided $Z$-test. We use the symbol $^*$ in Table 10.3 to indicate a 95% statistical significance level. On the philosophers, Flickr, and Twitter datasets, CESNA outperforms every baseline at a 95% significance level. On Facebook, CESNA outperforms all baselines, at a 95% significance level in all but one case. On Google+, CESNA performs the second best compared to ML ’12.

**Experiments on partially observed networks.** Combining network and attribute information into a single method should, in principle, lead to the development of a more robust community detection algorithm. In particular, when networks may be incomplete or partially observed, the performance of CESNA should degrade gently, as it should be able to rely on the node attribute information; this way, it should compensate for the noise in the network structure.

To investigate the robustness of performance under an unreliable network structure, we next explore the problem of detecting communities from partially observed networks where some fraction of edges are missing while the node attributes are fully available. For the sake of evaluation, we remove a fraction $\gamma$ of edges in the network uniformly at random. Note that we regard a removed edge in the same way as an unobserved edge, because in practice we cannot distinguish between edges that do not exist (e.g., users who aren’t friends) and edges that are unobserved (e.g., users who haven’t gotten around to declaring their friendship yet).
Rather than examining performance of all 6 baselines, we focus on making a comparison over the three top baselines that use either the network or the node attributes: BigCLAM, which considers the network only and is the best baseline in our experiments; MAC, which only considers the node attributes; and CODICIL, which is the best performing baseline that considers both the network and the attributes. For each baseline, we measure the relative performance that CESNA achieves over the baseline:

\[
\frac{F_1^\gamma(CESNA) - F_1^\gamma(Baseline)}{F_1^\gamma(Baseline)}
\]

where \( F_1^\gamma \) is the F1 score in Eq. 10.8 for the network with \( \gamma \) fraction of edges removed.

In Figure 10.3, we display experimental results (with standard deviation) as we vary from \( \gamma = 0 \) to \( \gamma = 0.8 \). We consider all datasets except philosophers (for which, results are too noisy due to the small network size). For Flickr, we omit performance of MAC, as the algorithm was not able to process it due to too high time and space complexity.

In all cases, we note similar behavior (Figure 10.3). As the network becomes more unreliable, the improvement of CESNA over BigClam increases. On the other hand, for methods that use node attributes (and the network structure), we note that in Google+, the performance improvement of CESNA remains constant, while in Facebook and Twitter, the performance improvement of CESNA slowly shrinks as more and more of the network structure gets removed.

The results are intuitive: Even though the network contains many missing edges, CESNA still outperforms other methods by better leveraging the information present in the node attributes. The results with MAC and CODICIL, which are decreasing functions of \( \gamma \), nicely shows that the performance gain from the network structure diminishes as we remove more edges.

Last, we also briefly note that similar results are observed with the relative improvement in Jaccard similarity, and that CESNA consistently outperforms the other four baselines not shown in Figure 10.3 for every value of \( \gamma \).

**Evaluating scalability.** We evaluate the scalability of community detection methods by measuring each method’s running time on synthetic networks as we increase the network size. Using the Forest Fire model [94], we generate synthetic networks with the forward and
CHAPTER 10. COMMUNITY DETECTION WITH NODE ATTRIBUTES

Figure 10.3: **Robustness to edge deletion.** Relative gain in F1 over the best method with network information only (BigCLAM) and with node attributes only (MAC) when edges are randomly removed.

backward probabilities set to 0.36 and 0.32, respectively. For attributes, we generate $K = 10$ attributes for each node with independent Bernoulli random variables with probability 0.5.

Figure 10.4 shows the running time of methods versus the network size. Among the four baselines that consider both network and the node attributes (i.e., Block-LDA, CODICIL, EDCAR, ML ’12), we show CODICIL since it is the fastest among the four. We also consider a parallelized version of CESNA (CESNA (24 threads)).

Overall, we notice that CESNA is the second-fastest method overall, next to BigCLAM. However, we note that BigCLAM is expected to be faster than CESNA, as it uses a similar optimization procedure as CESNA yet without considering node attributes. MAC is the slowest, and CODICIL is the second-slowest method. DEMON is faster than CESNA for
Figure 10.4: **Algorithm runtime comparison.** Block-LDA and ML ’12 are omitted as they took more time than 10,000 seconds for networks larger than 1% of the X-axis (3,000 nodes).

small networks (up to 100,000 nodes), though CESNA is faster when the network becomes larger.

We obtain even further speedup by considering a parallel implementation of CESNA. Using 24 threads on a single machine, CESNA takes just 10 minutes to process a 300,000 node network.

Last, we also note that all the baselines shown in Fig. 10.4 solve “simpler” problems than CESNA. For example, CODICIL detects non-overlapping communities, which is simpler than detecting overlapping communities. Demon and BigCLAM consider only network information, ignoring node attributes. Nevertheless, CESNA is faster than CODICIL and Demon, and it takes about 30% more time than BigCLAM. Comparing CESNA to methods that achieve the same goal — that is, overlapping community detection with node attributes (i.e., Block-LDA, EDCAR, and ML ’12) — CESNA has a considerable advantage in scalability, as it is about an order of magnitude faster.
10.5 Qualitative Analysis on Communities and Node Attributes

Incorporating node attributes into community detection gives two direct advantages. The first advantage is the improved accuracy in community detection, which we observed in the previous section. The second advantage is that the node attributes provide cues for interpreting detected communities. For example, a community in a Facebook ego-network might consist of a set of high-school friends, and the homogeneity of a particular attribute in a given community might help us to interpret and explain its existence. Such interpretations are an important part of community detection [4, 5, 118], yet finding them is very time-consuming and may require domain knowledge, as in traditional settings, one has to infer the meaning of a given community based only on the identities of its members. By incorporating node attributes, however, CESNA allows us to characterize a community by examining the attributes associated with high logistic weights in the model.

In this section, we qualitatively analyze our results in the Facebook network and the philosophers network to provide insights as to how CESNA brings the two advantages (better interpretability and higher accuracy). In both networks, we find that CESNA is able to find the attributes that are naturally related to the communities. On philosophers data, we also show how CESNA can improve the accuracy of detected communities by incorporating node attributes.

Analysis of Facebook communities. CESNA learns the logistic model weight $W_{kc}$ for each attribute $k$ and community $c$. Highly positive values of $W_{kc}$ mean that members of community $c$ are likely to have attribute $k$, and a highly negative value means the opposite (members are likely not to have the attribute). Not every attribute will be associated with community memberships, as some attributes may be irrelevant for a given community. To characterize the level of association between communities and attributes $k$, we measure the $l_2$ norm $\|W_k\|$ of its logistic weight $W_k = \{W_{kc}\}$.

To examine which attributes are related to communities (either positively or negatively), we examine detected communities in Facebook ego-networks. We find that the top attributes are related to schools, including the schools attended, the types of education that users received, and the major. On the other hand, the bottom five attributes include work
Figure 10.5: Communities of philosophers found by CESNA (left) and equivalent communities detected by BigCLAM (right). Top: Community of Islamic philosophers, Bottom: Community of theologians. BigCLAM regards some notable philosophers in red letters as belonging to the communities, even though these philosophers have little to do with theology / Islam. CESNA does not make such mistakes, as CESNA jointly learns attributes associated with the community. (Attributes are in Fig. [10.6].)

Analysis of Philosophers communities. To analyze the member nodes of communities along with their related attributes, we examine the communities in the Philosophers network.
First, using CESNA, we identify communities, and then for each community we identify the top ten positively related attributes. In Figures 10.5a, 10.5c we show two of the detected communities. The figure displays the titles of the corresponding Wikipedia articles. Moreover, we also show the attributes associated with the two communities in Figure 10.6. In this figure, word sizes are proportional to the value of the logistic weight $W_{kc}$, i.e., more relevant attributes are larger. Note that node attributes in this network represent Wikipedia articles other than philosophers to which the node links, e.g., the attributes include famous non-philosophical figures, abstract concepts, historic events, places, and so on.

First, based on the names of important attributes, e.g., “Early Islamic Philosophy,” we observe that the community in Figure 10.6a represents Islamic philosophers, even without querying for the names of the philosophers in Figure 10.5a. These attributes also include some non-philosophical people related to Islam (e.g., René Guénon).

Similarly, Figures 10.5c and 10.6b show the members of the second community detected by CESNA and the top ten related node attributes. Again, “Catechism of the Catholic Church” tells us that this community consists of theologians. The node attributes also include many priests (e.g., Lawrence of Brindisi, Bede, Hilary of Poitiers, Petrus Canisius, and Francis de Sales).

We also compare these communities to those detected by the BigCLAM. For each community detected by CESNA in Figs. 10.5a and 10.5c, we identify the most similar BigCLAM community based on the $F_1$ score. Figures 10.5b and 10.5d show these communities as detected by BigCLAM.

Interestingly, we note that the communities detected by BigCLAM contain some philosophers (in red) who are not Islamic philosophers/theologians. The reason is that these philosophers (in red) are so influential that they are very well connected to other members of the community. For example, Aristotle is connected to 229 philosophers (about one fifth of all the nodes); thus, he appears in both BigCLAM communities in Figure 10.5. However, by leveraging node attributes, CESNA does not make this mistake and finds that Aristotle does not share the same attributes as any Islamic philosophers or theologians, which, thus, excludes him.
Figure 10.6: The node attributes which CESNA learns to be associated with the communities. Left: For the community of Islamic philosophers, Right: For the community of theologians.

10.6 Related Work

We summarize the related work in Table 10.1 and group it along two dimensions. First, we consider how the methods model statistical dependency between communities, node attributes, and the underlying network (column $D$ of Table 10.1). Figure 10.1 shows the two paradigms that are typically used. In Figure 10.1a, community memberships $F$ generate both the graph $G$ and attributes $X$, while in Figure 10.1b, $F$ and $X$ are given independently, and then the graph $G$ is generated by the interaction between $F$ and $X$. Second, we focus on how the methods model the community memberships of individual nodes (columns $O$ and $H$). Soft-membership models associate a probability distribution with the node’s membership to communities, which means the more communities a node belongs to, the less it belongs to each individual community (simply because probabilities have to sum to one). On the other hand, hard-membership models associate an independent binary variable for each node and community pair and, thus, do not suffer from the assumptions made by soft-membership models.

As shown in Table 10.1, heuristic single-assignment clustering methods for networks with node attributes [41, 130, 165] detect hard node-community memberships, however, because each node can belong to exactly one community, these methods cannot detect overlapping communities.

LDA-based methods [11, 29, 100] aim to find sets of nodes that have similar “topics”
of attributes and link among each other. These topic models are based on the paradigm in Figure 10.1a where community memberships nodes generate links and node attributes. However, these methods assume soft community memberships, which leads to unrealistic assumptions about the structure of community overlaps [158]. We note that recently developed methods [138, 151] also assume soft-membership and the paradigm in Fig. 10.1a.

10.7 Conclusion

In this paper, we developed CESNA, a scalable method for overlapping community detection in networks with node attributes. Its comparison to the state-of-the-art baselines reveals that CESNA exhibits improved performance both in terms of the accuracy of the detected communities as well as in scalability. CESNA has a linear runtime in the network size and is able to process networks an order of magnitude larger than comparable approaches. Moreover, CESNA also helps with the interpretation of detected communities by finding relevant node attributes for each community.

There are many possible directions for future work. One direction is to extend CESNA to handle more general types of attributes. Similarly, extending the method to cluster the attributes into “topics,” while also identifying communities would likely lead to even easier interpretation of detected communities. Finally, incorporating other sources of information than node attributes, such as information diffusion [13] or edge attributes [11], would also be possible.
Chapter 11

Conclusions

11.1 Summary of Contributions

As stated in the introduction (Chapter 1), the goal of this thesis was to:

Develop a family of accurate and scalable methods to detect network communities, and study the community structure of large networks.

We began by defining a reliable notion of ground-truth communities in several real-world networks. From the ground-truth communities, we discovered a new paradigm that communities form dense overlap, which was in stark contrast to today’s common wisdom.

Based on this new paradigm, we developed novel community detection methods (AGM and BigCLAM), which advance the state-of-the-art both in scalability and accuracy. We also presented two extensions of the methods: CoDA for detecting 2-mode communities as well as cohesive communities and CESNA for networks with node attributes. The successful community detection by our methods led us to discover that densely overlapping communities give rise to the core-periphery structure in real-world networks.

11.1.1 Summary of Our Methods

In this thesis, we developed four different community detection methods: AGM (Chapter 6), BigCLAM (Chapter 7), CoDA (Chapter 9), and CESNA (Chapter 10). Table 11.1 gives
Table 11.1: Summary of the proposed community detection methods. Parallel?: Whethe- hter the method is parallelizable or not. Max. network: the maximum number of nodes in the network that a method can handle in ten hours. For the distinction between cohesive communities and 2-mode communities, refer to Chapter 9.

11.2 Future Work

In this section, we discuss open research directions to which we extend our work in this thesis. We begin by describing medium-term (e.g., within a few years) research agenda and then move to long-term (e.g., within a decade) directions.

11.2.1 Medium-Term Research Agenda

Although we developed two extensions of our methods in Chapter 9 and 10, there remain more opportunities for us to extend our AGM and BigCLAM.

Extension for weighted networks and signed networks. Throughout this thesis, we assume that our input network is unweighted, meaning that a pair of nodes is either connected
or not. However, in many applications, the network edges are naturally associated with weights [38, 85]. Moreover, in some cases the network edges have signs [93], denoting whether the relationships are positive or negative. When an weighted (or signed) network is given, most community detection methods work in an ad-hoc fashion (i.e., by treating weighted (or signed) edges as if they are binary). However, such ad-hoc adaptation may lead to undesirable results on community detection, because ignoring edge weights or edge signs will lose very important information.

By modifying the generative process of AGM and BigCLAM, we can develop an extension of our methods for weighted (or signed) networks. One way to achieve this goal is to follow our steps in this thesis for these types of networks. That is, we first collect the ground-truth communities in weighted or signed network, study how the weighted or signed edges reflect community affiliations, and finally develop generative models and fitting procedures.

**Local methods.** In some applications, we may focus on finding communities locally, i.e., discovering communities around a specific seed node rather than a whole network [7]. In such settings, local methods [7, 87], which refers to a specific part of the input network, can be more efficient than general community detection methods which consider the entire input network. Starting form a given seed node, local methods iteratively add adjacent nodes to its output (i.e., a local community around the seed node). While adding the nodes, the methods also keep track of a given quality function which specifies how good the output community is. If the quality function reaches its maximum value, the methods output the community and finish. As we can see, local methods detect only one community for a given seed node.

We can envision designing a local community detection method based on our AGM by following way. The main benefit of using AGM for local community detection is that we can detect overlapping communities around the given seed node. The possible challenges would include designing a procedure to add adjacent nodes around the seed node and developing a reasonable quality function.

Once we develop an AGM-based local community detection method, we may use it to further scale up our community detection methods. Given an input network, we first detect overlapping communities from multiple seed nodes using local AGM, and then we
merge these local detection results to recover the communities in the entire input network. This approach has been applied to scale up other overlapping community detection methods [34].

**Extension for distributed frameworks.** Another direction to extend our methods is to exploit distributed systems such as MapReduce [37]. Currently, our methods assume that an input network fits in the memory of a single machine. In this sense, the parallelization of our methods is multi-thread computing rather than distributed computing. To handle the networks too large to fit in a single machine such as the Facebook network [141], we can extend BigCLAM so that it works with distributed systems. One way to achieve this goal is to apply a local version of our methods (that is mentioned in the previous section) to each cluster and then merge the results.

### 11.2.2 Long-Term Research Agenda

We now discuss research ideas that require combining community detection and other disciplines of computer science. Building and optimizing a novel systems based on these ideas would lead to a host of additional research questions.

**Semi-supervised community detection.** Communities may exhibit different structural properties. For example, Grabowicz et al. [56] observed that communities based common identity are distinguished from communities based common bond, and Abrahao et al. [1] discovered that communities detected by different methods are well separable by their structural properties. Given such diversity in communities, we may envision “semi-supervised” community detection where the goal is to detect communities with a certain structural signatures (e.g., communities with a lot of triangles inside). To build such framework, we should pose the following questions: How communities are characterized / distinguished structurally? Given a structural property, what is a principled way to find the communities with the given structural property? How we can evaluate the quality of detected communities under this framework?

**Modeling the evolution of communities.** Networks can provide snapshots of time-evolving complex systems. By developing models on how networks evolve over time [94], one can
understand the evolution of complex systems. We can envision a similar approach on communities, *i.e.*, developing models for the evolution of communities. This direction requires us to follow a series of steps that we follow in this thesis. That is, we gather ground-truth on time-evolving communities, and then we set up a generative model and evaluate.

**Interactive navigation of networks with communities.** Visualizing networks becomes extremely challenging when the number of nodes is higher than 1,000, because the network looks like a gigantic “hairball” [34]. Identifying community structures can provide a great help in visualizing large networks. For example, we can visualize a given network at the level of communities. The end user can browse how communities are connected and overlap in the network, and then choose a community that the user wants to further explore. This direction will give rise to many interesting questions such as how to visualize communities effectively.

**11.2.3 Outlook**

The study of large scale networks has drawn a great deal of interests in the last decade [27], and is expected to grow even more in the next decade because networks are a natural form of representing many large scale data. Currently, most organizations that deal with large scale data make every endeavor to analyze and extract insights from the networks derived from their data. As community detection is a fundamental tool to understand and analyze large networks, the techniques and discoveries presented in this thesis could prove to be beneficial to a wide range of organizations that study large networks and big data.
Bibliography


Appendix A

Appendix

A.1 Table of Symbols

Tables A.1, A.2, A.3, and A.4 describe the symbols used in the thesis.
<table>
<thead>
<tr>
<th>Relevant chapter</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entire thesis</td>
<td>$V$</td>
<td>Node set</td>
</tr>
<tr>
<td></td>
<td>$E$</td>
<td>Edge set</td>
</tr>
<tr>
<td></td>
<td>$C$</td>
<td>Set of communities</td>
</tr>
<tr>
<td></td>
<td>$M$</td>
<td>Set of community affiliations</td>
</tr>
<tr>
<td></td>
<td>$G(V, E)$</td>
<td>Network with node set $V$ and edge set $E$</td>
</tr>
<tr>
<td></td>
<td>$B(V, C, M)$</td>
<td>Bipartite affiliation graph where $V$ is a set of nodes in $G(V, E)$, $C$ is a set of communities, and $M$ is a set of community affiliations</td>
</tr>
<tr>
<td></td>
<td>$A$</td>
<td>Adjacency matrix</td>
</tr>
<tr>
<td></td>
<td>$N$</td>
<td>Number of nodes</td>
</tr>
<tr>
<td></td>
<td>$u, v$</td>
<td>Nodes</td>
</tr>
<tr>
<td></td>
<td>$c$</td>
<td>Community</td>
</tr>
<tr>
<td></td>
<td>$\varepsilon$</td>
<td>Background edge probability for the nodes which do not share any community</td>
</tr>
<tr>
<td></td>
<td>$p(u, v)$</td>
<td>Edge probability between nodes $u$ and $v$</td>
</tr>
<tr>
<td></td>
<td>$p_c$</td>
<td>Edge probability for the nodes in community $c$</td>
</tr>
<tr>
<td></td>
<td>$C_{uv}$</td>
<td>A set of communities that nodes $u$ and $v$ share</td>
</tr>
<tr>
<td></td>
<td>$l$</td>
<td>log-likelihood</td>
</tr>
<tr>
<td></td>
<td>$\mathcal{N}(u)$</td>
<td>Neighbors of node $u$</td>
</tr>
<tr>
<td>Chapter 3</td>
<td>$</td>
<td>E</td>
</tr>
<tr>
<td></td>
<td>$\langle C \rangle$</td>
<td>Average clustering coefficient</td>
</tr>
<tr>
<td></td>
<td>$\langle D \rangle$</td>
<td>Average shortest path length</td>
</tr>
<tr>
<td></td>
<td>$\langle k \rangle$</td>
<td>Average node degree</td>
</tr>
<tr>
<td></td>
<td>$K$</td>
<td>Number of ground-truth communities</td>
</tr>
<tr>
<td></td>
<td>$S$</td>
<td>Average community size (the number of nodes)</td>
</tr>
<tr>
<td></td>
<td>$F$</td>
<td>Average community memberships per node</td>
</tr>
<tr>
<td></td>
<td>$s$</td>
<td>Community size</td>
</tr>
<tr>
<td></td>
<td>$m$</td>
<td>Community membership size of a node</td>
</tr>
<tr>
<td></td>
<td>$o$</td>
<td>The size of overlaps between a pair of communities</td>
</tr>
<tr>
<td></td>
<td>$f$</td>
<td>Relative overlap size</td>
</tr>
<tr>
<td></td>
<td>$F_c(x)$</td>
<td>Complementary cumulative distribution function of $x$</td>
</tr>
<tr>
<td></td>
<td>$p(f)$</td>
<td>Histogram of $f$</td>
</tr>
</tbody>
</table>

Table A.1: Table of symbols
<table>
<thead>
<tr>
<th>Chapter 4</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(k)$</td>
<td>Edge probability of two nodes given that the nodes share $k$ communities</td>
<td></td>
</tr>
<tr>
<td>$k$</td>
<td>The size of cliques that Clique Percolation Method searches for</td>
<td></td>
</tr>
<tr>
<td>$n(i)$</td>
<td>Set of neighbors of node $i$</td>
<td></td>
</tr>
<tr>
<td>$JAC(x, y)$</td>
<td>Jaccard similarity between two sets $x, y$</td>
<td></td>
</tr>
<tr>
<td>$I, J$</td>
<td>Two overlapping communities</td>
<td></td>
</tr>
<tr>
<td>$O$</td>
<td>The set of overlapping nodes that belong to both $I$ and $J$</td>
<td></td>
</tr>
<tr>
<td>$X$</td>
<td>The number of the nodes in $O$</td>
<td></td>
</tr>
<tr>
<td>$Y$</td>
<td>The number of the nodes in $I \setminus O$</td>
<td></td>
</tr>
<tr>
<td>$p$</td>
<td>Probability of an edge between the nodes in $I \setminus O$ and $J \setminus O$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 5</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_k$</td>
<td>Parameter for the edge probability $p_k$ of community $k$. $1 - p_k = e^{-x_k}$</td>
<td></td>
</tr>
<tr>
<td>$KS(f, g)$</td>
<td>Kolmogorov-Smirnov statistic between two functions $f, g$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 6</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$A, B, C$</td>
<td>Communities</td>
<td></td>
</tr>
<tr>
<td>$\lambda$</td>
<td>$l_1$-regularization parameter for finding the number of communities</td>
<td></td>
</tr>
<tr>
<td>$B_0(V, C_0, M_0)$</td>
<td>Bipartite community affiliation graph with a very large number of communities</td>
<td></td>
</tr>
<tr>
<td>${\hat{p}_c(\lambda)}$</td>
<td>Optimal edge probabilities with $l_1$-regularization with $\lambda$</td>
<td></td>
</tr>
<tr>
<td>$B(\lambda)(V, C(\lambda))$</td>
<td>Community affiliation graph with communities with non-zero $\hat{p}_c(\lambda)$</td>
<td></td>
</tr>
<tr>
<td>$L(B(\lambda))$</td>
<td>Likelihood of $B(\lambda)$</td>
<td></td>
</tr>
<tr>
<td>$K(\lambda)$</td>
<td>The number of communities in $B(\lambda)$</td>
<td></td>
</tr>
<tr>
<td>$\sigma(\lambda)$</td>
<td>Sigmoid function $\sigma(\lambda) = \frac{1}{1 + e^{\alpha \lambda + \beta}}$</td>
<td></td>
</tr>
<tr>
<td>$\delta$</td>
<td>A threshold for determining a significant drop in $\sigma(\lambda)$</td>
<td></td>
</tr>
<tr>
<td>$K_m$</td>
<td>Clique of $m$ nodes</td>
<td></td>
</tr>
<tr>
<td>$V_u$</td>
<td>Set of nodes that are less than 2-hop away from node $u$</td>
<td></td>
</tr>
<tr>
<td>$k$</td>
<td>Input parameter for Clique Percolation Method</td>
<td></td>
</tr>
<tr>
<td>$K$</td>
<td>The number of communities to detect</td>
<td></td>
</tr>
<tr>
<td>$C_i$</td>
<td>Ground-truth community</td>
<td></td>
</tr>
<tr>
<td>$\hat{C}_j$</td>
<td>Detected community</td>
<td></td>
</tr>
<tr>
<td>$F_g(C_i)$</td>
<td>F1 score (harmonic mean of precision and recall) between ground-truth community $C_i$ and the best-matching detected community</td>
<td></td>
</tr>
<tr>
<td>$F_d(C_j)$</td>
<td>F1 score (harmonic mean of precision and recall) between detected community $\hat{C}_j$ and the best-matching ground-truth community</td>
<td></td>
</tr>
<tr>
<td>$H(X</td>
<td>Y)$</td>
<td>Extended entropy for sets of sets $X, Y$ [86]</td>
</tr>
<tr>
<td>$p_v(C_i)$</td>
<td>$p$-value of detected protein community $C_i$ determined by the GO term finder</td>
<td></td>
</tr>
<tr>
<td>$\bar{p}$</td>
<td>Average $p$-value of detected communities</td>
<td></td>
</tr>
</tbody>
</table>

Table A.2: Table of symbols
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_{uc}$</td>
<td>Indicator variable whether node $u$ belongs to community $c$</td>
</tr>
<tr>
<td>$\alpha_c$</td>
<td>Parameter for $p_c$ of community $c$</td>
</tr>
<tr>
<td>$M_{uc}$</td>
<td>$M_{uc} = \sqrt{\alpha_c M_{uc}}$</td>
</tr>
<tr>
<td>$F_{uc}$</td>
<td>Non-negative membership factor between node $u$ and community $c$</td>
</tr>
<tr>
<td>$F$</td>
<td>Community membership factor matrix consisting of $F_{uc}$</td>
</tr>
<tr>
<td>A, B</td>
<td>Communities</td>
</tr>
<tr>
<td>$X_{uv}$</td>
<td>Interaction between nodes $u, v$</td>
</tr>
<tr>
<td>$f$</td>
<td>Link function</td>
</tr>
<tr>
<td>$D$</td>
<td>Loss function</td>
</tr>
<tr>
<td>$\delta$</td>
<td>Threshold to determine whether a node belongs to a community or not</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>$l_1$-regularization parameter</td>
</tr>
<tr>
<td>$g(i)$</td>
<td>The index of the best-matching detected community for the ground-truth community $C_i$</td>
</tr>
<tr>
<td>$Re(X,Y)$</td>
<td>Recall of $Y$ reconstructing $X$</td>
</tr>
<tr>
<td>$m$</td>
<td>The number of community memberships of a node</td>
</tr>
<tr>
<td>$d$</td>
<td>Average distance from a given node to all the other nodes</td>
</tr>
<tr>
<td>$d(u,v)$</td>
<td>Shortest path length between nodes $u, v$</td>
</tr>
<tr>
<td>$c(l)$</td>
<td>Fraction of the largest connected component in the induced subgraph of the nodes who belong to at least $l$ communities</td>
</tr>
<tr>
<td>$o_c$</td>
<td>Fraction of community $cs$ members in the largest overlap with any other community</td>
</tr>
<tr>
<td>$CS(i)$</td>
<td>Core score of node $i$</td>
</tr>
</tbody>
</table>

Table A.3: Table of symbols
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A, B, C, D</td>
<td>Communities</td>
</tr>
<tr>
<td>$M_{uc}, L_{uc}$</td>
<td>Indicator variables whether node $u$ has outgoing, or incoming membership to community $c$ respectively</td>
</tr>
<tr>
<td>$\alpha_c$</td>
<td>Parameter for $p_c$ of community $c$</td>
</tr>
<tr>
<td>$M_{uc}, L_{uc}$</td>
<td>$M_{uc} = \sqrt{\alpha_c M_{uc}}, L_{uc} = \sqrt{\alpha_c L_{uc}}$</td>
</tr>
<tr>
<td>$F_{uc}, H_{uc}$</td>
<td>Non-negative membership factor for outgoing or incoming membership (respectively) between node $u$ and community $c$</td>
</tr>
<tr>
<td>$F, H$</td>
<td>Community membership factor matrix consisting of $F_{uc}$ or $H_{uc}$</td>
</tr>
<tr>
<td>$\delta$</td>
<td>Threshold to determine whether a node belongs to a community or not</td>
</tr>
<tr>
<td>$T$</td>
<td>Number of ground-truth communities</td>
</tr>
<tr>
<td>$S$</td>
<td>Average community size (the number of nodes)</td>
</tr>
<tr>
<td>$R$</td>
<td>Average community memberships per node</td>
</tr>
<tr>
<td>$\delta(X,Y)$</td>
<td>Similarity measure between two communities $X,Y$</td>
</tr>
<tr>
<td>$O(c), I(c)$</td>
<td>Set of nodes with outgoing or incoming memberships for community $c$</td>
</tr>
<tr>
<td>$J(c)$</td>
<td>Jaccard similarity between $O(c)$ and $I(c)$, $J(c) = \frac{</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Threshold to determine whether a given community is 2-mode or cohesive</td>
</tr>
<tr>
<td>$P_{uv}$</td>
<td>Edge probability between nodes $u,v$</td>
</tr>
<tr>
<td>$F_{uc}$</td>
<td>Non-negative membership factor between node $u$ and community $c$</td>
</tr>
<tr>
<td>$F$</td>
<td>Community membership factor matrix consisting of $F_{uc}$</td>
</tr>
<tr>
<td>$A, B$</td>
<td>Communities</td>
</tr>
<tr>
<td>$X_{uk}$</td>
<td>$k$-th attribute of node $u$</td>
</tr>
<tr>
<td>$Q_{uk}$</td>
<td>Probability that $X_{uk} = 1$</td>
</tr>
<tr>
<td>$W_{kc}$</td>
<td>Logistic parameter for community $c$ to the $k$-th attribute</td>
</tr>
<tr>
<td>$L_G$</td>
<td>Likelihood from network</td>
</tr>
<tr>
<td>$L_X$</td>
<td>Likelihood from node attributes</td>
</tr>
<tr>
<td>$\delta$</td>
<td>Threshold to determine whether a node belongs to a community or not</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>$l_1$-regularization parameter</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Weight between the likelihoods from network and the node attributes</td>
</tr>
<tr>
<td>$T$</td>
<td>Number of ground-truth communities</td>
</tr>
<tr>
<td>$S$</td>
<td>Average community size (the number of nodes)</td>
</tr>
<tr>
<td>$R$</td>
<td>Average community memberships per node</td>
</tr>
<tr>
<td>$\delta(X,Y)$</td>
<td>Similarity measure between two communities $X,Y$</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Fraction of edges to be deleted</td>
</tr>
</tbody>
</table>

Table A.4: Table of symbols