Statistical Inference of Large-scale Structure in Networks

by

Austin Polanco

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Applied Physics)
in the University of Michigan
2025

Doctoral Committee:

Professor Mark Newman, Chair
Associate Professor Marisa C. Eisenberg
Associate Professor Jordan M. Horowitz
Assistant Professor Abigail Z. Jacobs
Professor Xiaoming Mao

Austin Polanco
polancoa@umich.edu
ORCID iD: 0000-0003-2158-1249

© Austin Polanco 2025

Dedication

To my late grandfather,
for teaching me it's fun to learn.

Acknowledgements

This thesis has been in the making for longer than my tenure at the University of Michigan. It is wholly the product of three decades of support, encouragement, and motivation from countless people who have been part of my life during this time. It is with great pleasure that I acknowledge both them and their contributions as I complete this work.

First and foremost, I would like to thank my thesis committee for dedicating their time to review my work. In particular, I am deeply grateful to my advisor, Mark Newman, for taking me on as a student despite being in over my head and having bitten off more than I could chew. His guidance pushed me to think deeply, be thorough and meticulous, and, most importantly, take *ownership* of my work. I am especially thankful for his patience, empathy, and understanding during the more challenging phases of my Ph.D. My time with him has provided irreplaceable experience that I will carry forward in my career.

I also owe a debt of gratitude to the faculty and staff of the applied physics program, who made my transition to graduate school seamless. Cagliyan Kurdak and Cynthia D'Agostino McNabb treated me like family and were always there to ensure that I stayed on track and never had to worry about funding.

Reaching this point in my academic career would not have been possible without the support of many professors and teachers along the way. I am particularly grateful to Ben Blum-Smith for the many hours he spent with me after school when I was eleven and again at sixteen, dedicating time from *his own* Ph.D. studies for our "crazy math" sessions. His encouragement kindled a curiosity and way of thinking that unquestionably influenced my decision to pursue graduate studies. Twelve years ago, he told me that he was as invested in my education as I was, and he has truly lived up to those words, providing insightful and invaluable mentorship that has fostered both my mathematical and personal growth, enabling me to be where I am today.

Furthermore, I am thankful for the friendships I formed in graduate school, especially Joseph Essman for indulging my frequent rants about food and urban planning. I also thank my roommates Sophia Eakins and Amelia Grant-Alfieri, for many enjoyable cooking nights. Additionally, I am grateful to Nathanael Gutierrez, Gur-Eyal Sela, and Justin Wang, whose companionship made graduate school a much less lonely experience.

I would also like to thank my family, who have been there every step of the way. They have supported me both financially and emotionally, accepting my decision to spend all of my twenties in school rather than entering the workforce, and for that, I owe them immense gratitude. Finally, my heartfelt thanks go to my loving wife, Meritxell Feliu Ribas, for her unwavering support and compassion; without her, the writing of this thesis would have been done with much less spirit.

Although I have named only a few, numerous others—too many to enumerate here—have offered invaluable support along this journey. Their contributions are greatly appreciated and have been integral to the completion of this thesis. To say that *it takes a village* is not just an empty platitude; this thesis stands as a testament to this truth, as it is because of the collective efforts of everyone who has played a role in my life that it came to be.

Table of Contents

Dedication . . . ii

Acknowledgements . . . iii

List of Figures . . . vii

List of Tables . . . x

Abstract . . . xi

Chapter 1. Introduction . . . **1**

1.1 Network Preliminaries . . . 1
1.1.1 Dyadic Networks . . . 2
1.1.2 Higher-order Networks . . . 4
1.1.3 Networks in Science . . . 6
1.2 Probing Network Structure . . . 6
1.2.1 Degree and Degree Distribution . . . 7
1.2.2 Network Centrality . . . 8
1.2.3 Network Similarity . . . 10
1.2.4 Assortativity . . . 11
1.3 Random Graph Models . . . 12
1.3.1 The Erdős-Rényi Random Graph Model . . . 12
1.3.2 The Configuration Model . . . 13
1.3.3 Random Graphs as Null Models . . . 14
1.3.4 Community Detection . . . 16
1.4 Link Prediction . . . 19
1.4.1 Topological Methods . . . 21
1.4.2 Probabilistic Methods . . . 22
1.4.3 Network Embedding Methods . . . 22
1.4.4 Evaluating Performance . . . 24
1.5 Contributions . . . 28

Chapter 2. Drug-disease Networks and Drug Repurposing . . . **30**

2.1 Introduction . . . 30
2.2 Drug-disease Network . . . 33
2.3 Link Prediction Revisited . . . 34
2.3.1 Elementary Algorithms . . . 35

2.3.2 Machine Learning . . . 36
2.3.3 Probabilistic Network Models . . . 38
2.4 Results . . . 39
2.4.1 Running Time . . . 43
2.4.2 Bounds on the Number of Possible Discoveries, False Positives, and Precision . . . 44
2.5 Conclusions . . . 45

Chapter 3. Hierarchical Core-periphery Structure . . . 47

3.1 Introduction . . . 47
3.2 A Hierarchical Model of Network Structure . . . 49
3.2.1 Prior on Group Assignments . . . 52
3.2.2 Prior on Number of Groups . . . 52
3.3 Sampling from the Posterior Distribution . . . 53
3.3.1 Monte Carlo Algorithm for the Case of Fixed Group Number . . . 53
3.3.2 Algorithm for Varying Group Number . . . 56
3.3.3 Implementation . . . 59
3.4 Example Applications . . . 60
3.4.1 Traditional Two-group Core-periphery Structure . . . 60
3.4.2 Structure with an Arbitrary Number of Groups . . . 63
3.5 Conclusions . . . 65

Chapter 4. Bayesian Inference of the Microcanonical Hypergraph Block Model . 67

4.1 Introduction . . . 67
4.2 Microcanonical Hypergraph Block Model . . . 68
4.2.1 Prior Distributions on Model Parameters . . . 70
4.2.2 Prior on Hyperedge Counts . . . 71
4.2.3 Prior on Group Assignments . . . 71
4.3 Sampling from the Posterior Distribution . . . 72
4.4 Example Applications . . . 75
4.5 Conclusions . . . 78

Chapter 5. Conclusion . . . 79

Appendices . . . 82

Bibliography . . . 91

List of Figures

1.1 Euler's Seven Bridges 1

1.2 Examples of different network types and their corresponding adjacency matrices. In (a) we have a simple, unweighted, undirected network. In (c) we show an example network with multiedges and self-edges. Panel (e) is a weighted network, with the weighted edge illustrated in bold. Lastly, (g) is a directed network. Panels (b), (d), (f), and (h) are the adjacency matrices for the networks that precede them. 3

1.3 Equivalent representations of a network as a hypergraph and a bipartite network. Panel (a) is the incidence matrix that both representations share, the rows represent the hyperedges and the columns the nodes they join. In (b) we illustrate the hypergraph representation, where hyperedges (depicted as regular polygons) can join an arbitrary number of nodes. In (c) we show the same network as a bipartite network, hyperedges are shown as *nodes* in yellow. In blue are nodes the hyperedges join. 5

1.4 The sociograms of Jennings and Moreno. Boys are triangle nodes, girls are circle nodes, and lines join nodes if they interacted. 6

1.5 NYC subway commuting network and its degree distribution. In (a), nodes represent subway stations; node size being proportional to degree—highlighting major hubs in midtown and lower Manhattan. In (b), the degree distribution, illustrating the heavy-tailed nature of the network where a few stations connect to the majority of others. 9

1.6 Contact network of French high school students [42]. In (a) Nodes are colored by class: red, yellow, or blue for different student classes, and teal for teachers. In (b) Distribution of modularity scores for 10,000 networks with the same degree distribution and node labels. The vertical line indicates the modularity of the observed network. 15

1.7 If we assume a network is incompletely measured, the goal is to predict which edges are missing. The measured edges are illustrated as solid lines while the edge to be predicted is dashed. 20

1.8 Two networks, observed and actual. In (a) we have an example observed network. In (b) we show the true network, predicted edges depicted in red. . . 21

1.9 Visualization of embedding a network into some latent space. The embedding function f maps nodes in the network to a k-dimensional space, where f is learned by optimizing Equation (1.27). 23

1.10 Estimated latent node positions for the network in Figure 1.8a using the first two singular values of its adjacency matrix. 24
1.11 (a) Contingency table for a binary classifier. In (b) we give a visualization of classifier predictions. The ellipse contains the positive *predictions* of a classifier, surrounded by negative predictions. The circles are the data with color distinguishing their ground truth class; blue being positive and yellow negative. 25

2.1 A bipartite network of drugs and diseases of the type considered here. Nodes in the network are of two types—drugs and diseases—and edges connect only nodes of unlike types to indicate which drugs are indicated for treatment of which diseases. The network is assumed to be incomplete, so some edges that should be present are missing from the data (dashed lines). Our goal is to identify these. 31
2.2 Visualization of the network of diseases. In this figure, we show a subset of the diseases in our network, those falling in the ten most common disease categories, as labeled. Connections between diseases represent disease pairs that are treated by one or more of the same drugs and only the strongest connections are shown (disease pairs treated by the largest numbers of common drugs). 34
2.3 (a) Receiver operating characteristic (ROC) curves for seven of the best performing algorithms. The dashed diagonal line represents the expected performance of a no-skill (random) classifier. Inset: Area under the curve (AUROC) for the microcanonical SBM for various fractions of edges removed. (b) Precision/recall curves for the same selection of algorithms. Colors are the same as in (a). Inset: Top-100 precision and area under the prediction/recall curve (AUPR) for each algorithm. 41

3.1 The model proposed here is capable of capturing a range of different types of structure, including (a) traditional nested groups, (b) tree-like branching structures, and (c) complex structures of overlapping groups. 48
3.2 Two types of two-group core-periphery structure distinguished by the hierarchical model. (a) is a network of airline routes among European airports [22]. In (b) we show a A network of associations among terrorists involved in the 2004 Madrid train bombing [51]. (c) is a network representation of the Internet in November 1997 at the autonomous system level [74]. While (d) is a network of hyperlinks among a set of US political blogs [2]. Panels (a) and (b) have a core that is densely connected within itself but only sparsely connected to the periphery. This is the traditional definition of core-periphery structure. Panels (c) and (d) on the other hand show a kind of "inside-out" structure in which the core is strongly connected within itself and strongly connected to the periphery. 61
3.3 Political books network with a periphery and two cores corresponding to left and right leaning books. 63
3.4 American Football network where each of the cliques are connected to each other via the periphery. 64

3.5 Structure found in the network of windsurfers. (a) Traditional community structure. (b) Core-periphery structure. 65

4.1 Illustration of the generative process for Equation (4.6). (a) First the nodes are labeled by their group. (b) Node stubs are then chosen (bolded stubs) to fill the different slots for the varying hyperedge types. (c) Next, slots come together to form the different hyperedges 4.1c. (d) Lastly, the edges form the hypergraph. 69

4.2 For each data set we show the highest probability state found from our Monte Carlo simulation. (a) Political books network. (b) Plant-herbivore hypergraph. Herbivores are the edges and plants the nodes. (c) French high school contact hypergraph. (d) French primary school contact hypergraph. Panels (e)-(h) are contingency tables recording the classification accuracy of our model. 76

List of Tables

1.1 Cosine Similarity Scores for unconnected vertex pairs 21

2.1 Definitions of similarity measures $\sigma(u, v)$ between network nodes used in Eq. (2.1). In these expressions, u and v are nodes, d_u and d_v are their degrees (the number of edges connected to them), and n_{uv} is the number of common network neighbors between u and v. For nodes of degree zero, all of these measures (except the common neighbors measure) give $0/0$, in which case the similarity is defined to be zero. 37

2.2 Performance measures for link prediction algorithms, estimated from 50 repetitions of cross-validation with 10% of edges removed from the network. We divide our algorithms into three categories: "Elementary" denotes algorithms based on node degrees or simple similarity measures such as counts of common neighbors between node pairs; "Machine learning" denotes methods such as matrix factorization, Bayesian, and deep-learning methods; "SBM" denotes network-based methods that make use of the stochastic block model. The performance measures we use are the area under the ROC curve (AUROC), the area under the precision/recall curve (AUPR), area under the precision/recall curve normalized by prevalence, and precision over the top 100 predictions. Numbers in parentheses indicate standard errors on the trailing digits. Numbers in bold indicate the best performers. Running time is for a single run of each algorithm. 42

4.1 Summary of test hypergraphs. n is the number of nodes, m the number of hyperedges, $\mathrm{med}(l)$ is the median hyperedge size, $\langle l \rangle$ is the average hyperedge size, σ_l is the standard deviation of hyperedge sizes, $l_{\max}$ is the size of the largest hyperedge, and k is the number of ground-truth communities. 75

A.1 Categories of drugs removed from the data set, along with the number in each category. Note that the total number of drugs removed is less than the sum of the entries in the right-hand column because some drugs belong to more than one category. 83

Abstract

Networks are flexible representations of systems governed by many interacting components. This flexibility has led to their application across a broad range of disciplines, modeling trade between countries, cascading failures in power grids, the structure of the World Wide Web, neuronal connections in microscopic organisms like C. elegans, and even human and animal social dynamics. Crucially, networks capture the patterns of interactions between the components of a system. The structured nature of networks can be used to better understand the underlying dynamics of the system. In this thesis, we develop theoretical models to reveal large-scale structures in networks and explore how network structures can be used to make predictions.

We begin with a basic overview of how networks and network tools have been used to study scientific systems. Then, we motivate the use of networks by constructing a novel network of drugs and the diseases they treat. To this network, we apply network models for link prediction to address questions related to drug repurposing. Link prediction aims to estimate those edges that may be missing from the network, the prediction of which could motivate the study of potential candidates for drug repurposing. Previous work on network-based drug repurposing has focused primarily on networks of indirect connections between drugs and diseases, such as drug-protein or drug-gene networks. Some work has been done on drug-disease interactions, but using networks much smaller in scope. We do extensive cross-validation tests and discuss bounds on how well these link prediction methods can do on our dataset.

We then turn our focus to the modeling of large-scale structures in networks. First, we consider core-periphery structure, a structure commonly found in many types of network. Often times, this structure is considered with only two groups: a core and a periphery. This partition of the network is related to the importance or centrality of nodes, with more central nodes being in the core. There has been substantial research on models for networks characterized by this structure, and we extend these works by proposing one that can reveal more flexible divisions network. Crucially, we allow for multiple cores and peripheries, but also more general hierarchical structures. We propose a generative model that allows for any number of groups along with a model-fitting approach to determine

the number of groups directly from the network data.

Second, we study community structure in higher-order networks. With the growing popularity of higher-order networks, there is increasing interest in extending established methods for treating dyadic networks to these more complex ones. Detecting communities in higher-order networks could potentially reveal more sophisticated social structures in social networks or nontrivial functions of components in neuronal networks. We propose a generative model for community structure in higher-order networks and evaluate its performance on a representative selection of empirically measured networks with ground truth community structure.

These works collectively use network structure to provide insight into real-world network data and contribute additional tools for researchers to study their network systems.

Chapter 1

Introduction

1.1 Network Preliminaries

Understanding complex systems, especially those characterized by many interacting parts, poses a significant challenge in many disciplines. Frequently, we approach this problem by employing strategies for simplification to make the task more manageable. When appropriate, *networks* offer a powerful method for such simplification, serving as an insightful modeling tool. In their most basic form, networks are a set of dots connected by lines. In the language of the field, the dots are often called *nodes* (or *vertices*) and the lines *edges*.

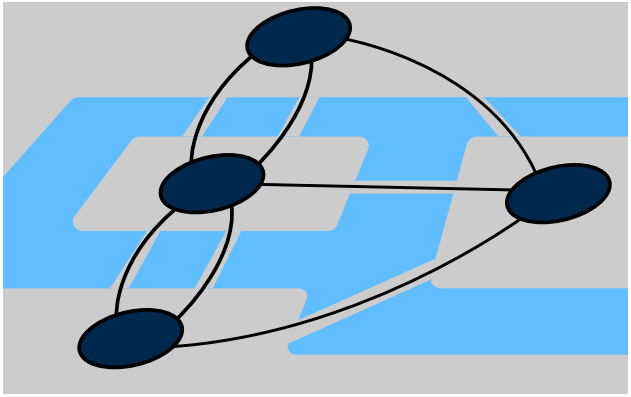


Figure 1.1: Euler's Seven Bridges

Some of the earliest work on networks came from mathematics—where networks are often called *graphs*—and began with the *Seven Bridges of Königsberg* problem. In this problem, seven bridges connect different parts of the Prussian city of Königsberg[1]. The challenge is to find a path that crosses each bridge exactly once. The mathematician

[1]Today known as Kaliningrad, this city is the administrative center of a province bearing the same name, notable for being an exclave of the Russian Federation wedged between Poland and Lithuania.

Leonard Euler realized that the specific shapes and distances between the land areas were not relevant; rather it was *how* the bridges connected them that mattered. He proved that crossing all seven bridges exactly once was impossible [37], making a seminal contribution to what we now call *graph theory*.

1.1.1 Dyadic Networks

But how exactly does one use a collection of dots and lines to represent scientific systems and solve problems? Fundamentally, the language of networks is expressed through mathematics, and as discussed previously, the dots are called nodes, and the lines edges. Typically, the edges join only pairs of nodes; however, later we will see how to generalize this to allow for more complex connections. In such cases, the term *dyadic network* can be used specifically to distinguish networks containing only pairwise connections from their higher-order counter parts. More formally, we denote $G = (V, E)$ to define the network G with a set of nodes V and a set of edges joining them E, with $n = |V|$ and $m = |E|$ being the number of nodes and edges, respectively. There are a number of representations of G that can be used to do calculations. One of the most flexible of such representations is known as the *adjacency matrix*. Its most elementary form has a very intuitive definition: it is an $n \times n$ matrix, A, such that for any pair of nodes $u, v \in V$

$$a_{uv} = \begin{cases} 1 & \text{if edge } \{u, v\} \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{1.1}$$

Adjacency matrices of the form described in Equation (1.1) correspond to *simple networks*, where edges denote only that a pair of nodes interact in some manner. Should one need to be more specific about the nature of the interaction, there are a number of attributes that can be used to decorate edges, such as:

- Self-edges: edges that connect a node to *itself*.
- Multiedges: multiple edges connecting the *same* pair of nodes
- Weighted edges: edges represented with a real value denoting the *strength* or *weight* of the interaction
- Directed edges: edges representing the *direction* of the interaction; from one node to another.

Generalizing the definition of an adjacency matrix to include these attributes can be done straightforwardly. For multi- and weighted edges, the corresponding entry is generally replaced by the weight or multiplicity of edges between u and v. To denote direction, a_{uv} is nonzero if and only if node v points to node u. Self-edges, somewhat counterintuitively, are usually represented by a 2 in the undirected case, and 1 in the directed case. In Figure 1.2 we illustrate a variety of network types with their corresponding adjacency matrices.

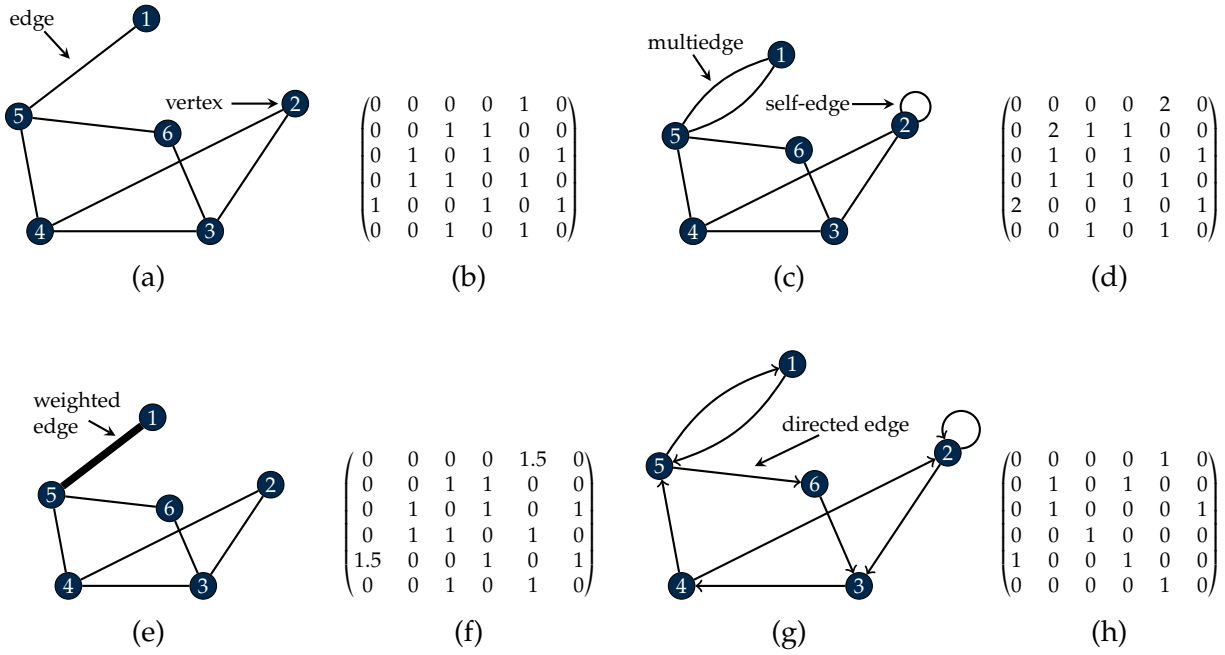


Figure 1.2: Examples of different network types and their corresponding adjacency matrices. In (a) we have a simple, unweighted, undirected network. In (c) we show an example network with multiedges and self-edges. Panel (e) is a weighted network, with the weighted edge illustrated in bold. Lastly, (g) is a directed network. Panels (b), (d), (f), and (h) are the adjacency matrices for the networks that precede them.

While adjacency matrices provide a convenient representation of networks, they are not the only one available. Another network representation that will be central to some of the topics discussed in this thesis is the *incidence matrix*. Whereas adjacency matrices directly represent connections between nodes, incidence matrices shift the focus to the relationship between nodes and the edges *incident* upon them. For a network with n nodes

and m edges, the incidence matrix, B, is an $m \times n$ matrix such that

$$b_{lu} = \begin{cases} 1 & \text{if node } u \text{ belongs to edge } l, \\ 0 & \text{otherwise.} \end{cases} \tag{1.2}$$

For the simple network shown in Figure 1.2a, its 7×6 incidence matrix is

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}. \tag{1.3}$$

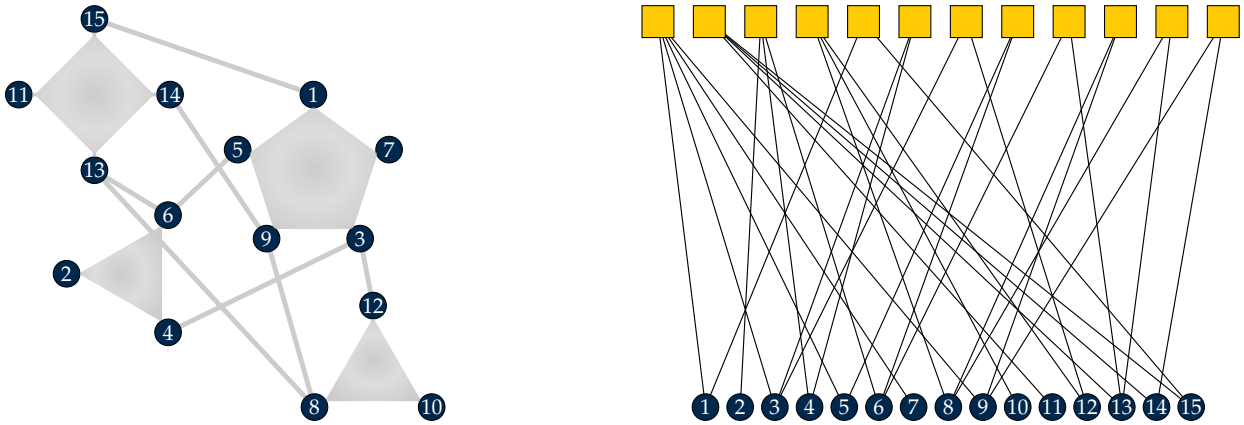
1.1.2 Higher-order Networks

So far, we have focused our discussion on networks where edges connect only two nodes. However, some networks can have a more complex structure with edges connecting any number of nodes. Such edges are generally referred to as *hyperedges* and any network with hyperedges is called a *hypergraph*, or sometimes a *higher-order* network. A hypergraph generalizes a traditional (or dyadic) network by representing relationships between more than just pairs of nodes. For example, consider a set of students forming various club committees, such as a chess club committee or a jazz club committee. Each committee has several members, and while we can create a network by placing pairwise edges between every committee member (on the same committee), committees typically interact as a whole rather than in pairs. Therefore, it might make more sense to join committee members using a single hyperedge to more accurately represent the nature of the social interactions.

Hypergraphs have another, arguably more convenient, interpretation: *bipartite networks*. A bipartite network, also known as a *two-mode network*, consists of two different types of nodes, with edges joining pairs of nodes only of different types. Networks with just one set of nodes, like those in Figure 1.2, are called *unipartite networks* when distinguishing them from bipartite ones. Formally, a bipartite network can be denoted as $\Gamma = (U, V, E)$, where U and V are *disjoint* sets of nodes, and edges in E connect a node $u \in U$ to a node $v \in V$, never linking nodes from the same set. Returning to our earlier example of students and committees; one set of nodes can represent the students, and the other set

$$\begin{pmatrix}
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0
\end{pmatrix}$$

(a)



(b) (c)

Figure 1.3: Equivalent representations of a network as a hypergraph and a bipartite network. Panel (a) is the incidence matrix that both representations share, the rows represent the hyperedges and the columns the nodes they join. In (b) we illustrate the hypergraph representation, where hyperedges (depicted as regular polygons) can join an arbitrary number of nodes. In (c) we show the same network as a bipartite network, hyperedges are shown as *nodes* in yellow. In blue are nodes the hyperedges join.

the committees. An edge between a student node and a committee node indicates that the student is a member of the committee.

Both hypergraphs and bipartite networks can be expressed completely using an incidence matrix. Whereas for hypergraphs, only the columns of the matrix represent nodes, in bipartite networks, both rows and columns are nodes, only that they represent nodes of different sets. In our student-committee example, you can equivalently interpret the committees as hyperedges or as a distinct set of nodes. In Figure 1.3 we show an example of a hypergraph, its equivalent bipartite network representation, and their corresponding

incidence matrix.

1.1.3 Networks in Science

The notation previously presented is largely shared between mathematicians studying graph theory and scientists using networks as an analytical tool. While graph theory remains an important and relevant area of study, its focus is primarily on the mathematical properties of idealized graphical structures. In contrast, the study of networks is based on empirically measured interactions and aims to gain insights into real-world phenomena.

While Euler was doing his work on graphs in the 18^{th} century, it was not until the 20^{th} century that the use of networks in science began to develop. During this period, social scientists began to explore methods for measuring social relationships. Significant contributions to the field came from Helen Hall Jennings and Jacob Levy Moreno, who were instrumental in establishing networks as a scientific area of study. They introduced some of the earliest examples of networks—which they called *sociograms*—to study social systems [82, 59]; a notable example being a network of schoolchildren interacting on the playground (reproduced in Figure 1.4). The introduction of sociograms provided



(a) (b)

Figure 1.4: The sociograms of Jennings and Moreno. Boys are triangle nodes, girls are circle nodes, and lines join nodes if they interacted.

researchers with a powerful tool to visualize and analyze social interactions. This advancement paved the way for further exploration into network structures, enabling researchers to extract meaningful insights about the underlying systems they represent.

1.2 Probing Network Structure

Once a network has been measured, the goal is to use it to better understand the system being modeled. Visualizations as in Figure 1.4 can indeed be enlightening, allowing

one to see structural patterns not immediately obvious from raw data. However, their utility is somewhat limited to qualitative descriptions, and as the size of the network grows, these visualizations become quite involved, with many of the structural features becoming obscured. In contrast, mathematical analyses provide a way to quantitatively summarize specific and relevant features of interest. Armed with the tools discussed in the previous section, we can begin a quantitative exploration of the structural properties of networks.

1.2.1 Degree and Degree Distribution

Networks exhibit structure at various scales, extending from the level of individual nodes to broader global network properties. At the smallest scale, we focus on nodes and their direct connections to others. The most fundamental property of a node at this level is its *degree*: the number of edges attached to the node. We can calculate the degree, d_u, of any node u directly from a network's adjacency matrix as follows:

$$d_u = \sum_v a_{uv}. \tag{1.4}$$

The degree is the number of *ends of edges* incident on a node, so if we count the degrees of all the nodes we will have equivalently counted every edge in the network twice, more concretely:

$$\sum_{uv} a_{uv} = \sum_u d_u = 2m, \tag{1.5}$$

where m is the number of edges. Note that this definition of degree makes it clearer why a self-edge is represented by a 2 in the adjacency matrix. If we have a network with just one node connected to itself, it has *two* ends of edges attached and Equation (1.5) gives the expected result. Similar expressions for degree can also be defined for other types of networks as well. For weighted networks, we can use Equation (1.4) directly, only that d_u will represent a *weighted* degree. In the directed case, we have two types of edges: in-going edges and out-going edges, so it makes sense to measure an *in-degree*, $d_u^{\text{in}} = \sum_v a_{vu}$ and an *out-degree*[2], $d_u^{\text{out}} = \sum_v a_{uv}$.

Once we know the degrees of all the nodes, we can study an illuminating global property of the network: the *degree distribution*. For a network of n nodes, let n_d denote

[2]Recall that for directed networks, the adjacency matrix is not symmetric by definition, so the order of the indices matters.

the number of nodes with degree d. The degree distribution is the fraction of nodes, $p_d = n_d/n$, with degree d. This distribution provides insight into the central tendency and variability of the connections in the network, shedding light on the large-scale organization of the nodes. For example, the average degree can be calculated as follows:

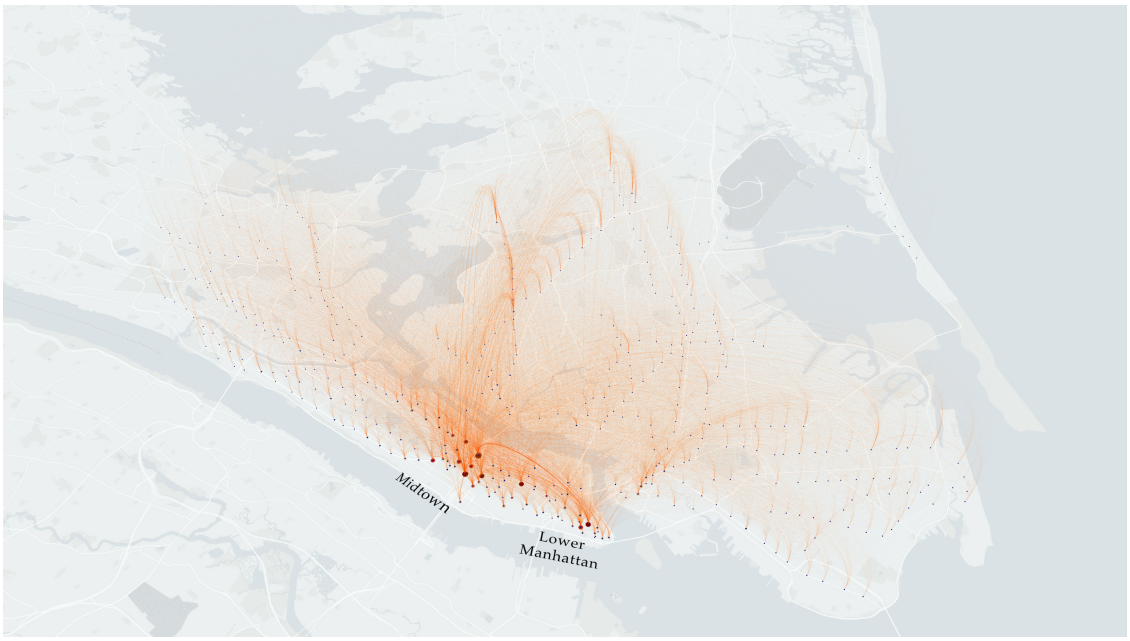
$$\langle d \rangle = \sum_{d'} d' p_{d'} = \frac{2m}{n}. \tag{1.6}$$

We may also be interested in the spread of the degrees in the network. Do many nodes have roughly the same degree or is there quite a bit of variation? In empirical networks, often the distribution is heavy-tailed [30, 20], meaning there are a handful of nodes with far higher degree than most others. It would, therefore, be natural to assume that the degree of a node tells us something about how *important* a node is to the structure of the network.

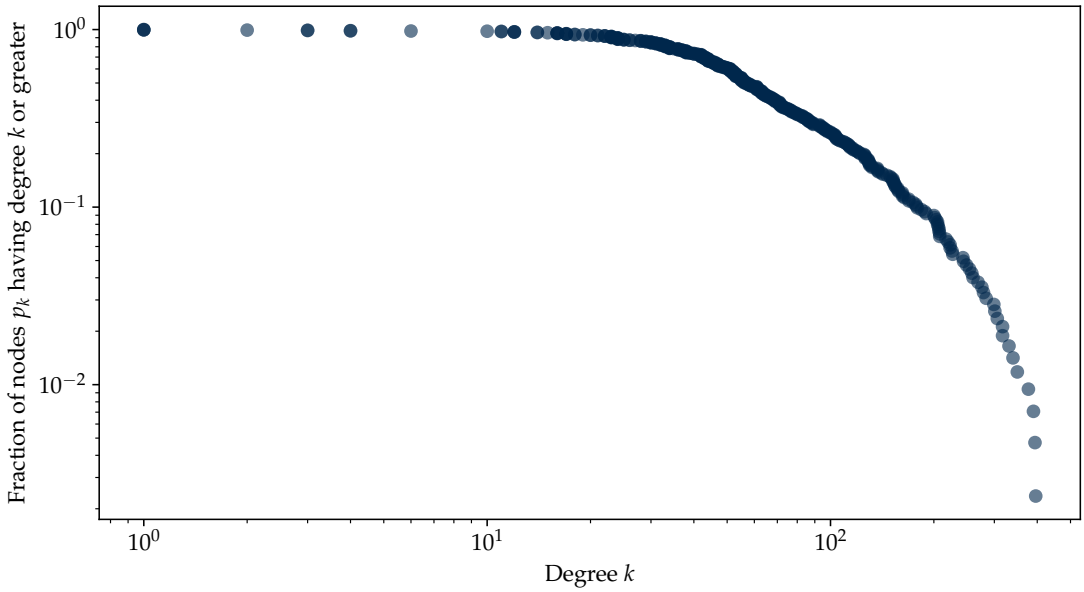
Consider the commuting network in Figure 1.5a. Here, nodes represent subway station complexes (of which there are 424) and edges represent the trips that occur between them in any direction. Most possible trips between stations are not frequently made; rather, a large fraction of trips are concentrated between a small set of stations located primarily in midtown and lower Manhattan. This is illustrated in Figure 1.5a, where the node radius is proportional to its degree. These stations function as major hubs, located in the economic centers of Manhattan, attracting a significant flow of passengers from the outer boroughs, where stations typically exhibit lower degrees. The resulting degree distribution is heavy-tailed, highlighting the connectivity of the network, where a few stations handle most of the commuting trips. Amongst these, Times Square-42 Street stands out as the station with the highest degree, connecting with 93 percent of all stations in the network. The hub stations are crucial; removing any one of them would significantly impact the network's structure, potentially disconnecting many stations. In contrast, removing a smaller node from the outer boroughs would leave the network largely intact. This underscores the concept of node importance in the network, which in this case, we have quantified by measuring its degree, a notion commonly known as *degree centrality*.

1.2.2 Network Centrality

There are, however, other *centrality* measures each aiming to capture different notions of importance. Degree centrality considers nodes with more neighbors as more important. However, we could argue that not all neighbors are equal; and that neighbors that are themselves important should contribute more to a node's centrality score than those that



(a)



(b)

Figure 1.5: NYC subway commuting network and its degree distribution. In (a), nodes represent subway stations; node size being proportional to degree—highlighting major hubs in midtown and lower Manhattan. In (b), the degree distribution, illustrating the heavy-tailed nature of the network where a few stations connect to the majority of others.

are less important. One such measure that aims to do this is the *eigenvector centrality* [18]. Eigenvector centrality extends degree centrality by contributing to a node's centrality in proportion to the centrality of its neighbors. We can therefore express the eigenvector centrality, c_u, of node u in terms of the adjacency matrix as

$$c_u = \kappa^{-1} \sum_v a_{uv} c_v. \tag{1.7}$$

Equation (1.7) can be expressed in matrix notation as $A\vec{c} = \kappa\vec{c}$, an eigenvalue equation suggesting its name. Assuming that we do not want negative centrality scores, then we can use the Perron-Frobenius theorem to pin down the single positive eigenvector of the adjacency matrix, $\vec{c}$, whose element c_u gives the centrality score of node u. There is no best centrality measure, and variants of eigenvector centrality such as Katz [64], PageRank [94] (used in early versions of Google search), and Hubs and Authority [66], have also been used to measure the importance of nodes.

1.2.3 Network Similarity

Another commonly used measure to investigate local network structure is node *similarity*. This measure aims to quantify the degree to which a pair of nodes is similar based on the local network structure. Like with centrality, there are different ways to quantify similarity, but most are variations of the same idea: the more neighbors a pair of nodes share, the more similar they are. For a simple network, we can count the number of common neighbors between nodes u and v as follows:

$$n_{uv} = \sum_w a_{uw} a_{wv}. \tag{1.8}$$

This alone, however, is not enough to determine if a pair of nodes is similar. If nodes u and v share four neighbors and nodes i and j share twelve neighbors, are i and j more similar simply because they share more neighbors? If the four neighbors shared by u and v constitute all of their neighbors, while the twelve shared by i and j are just a small fraction, then one might intuitively assume that u and v are more similar. To address this, most similarity measures are placed on a scale usually by means of some sort of normalization. One common normalization technique is to use the geometric mean of the degrees of the

node pair,

$$\sigma_{uv} = \frac{n_{uv}}{\sqrt{d_u}\sqrt{d_v}} = \frac{\sum_w a_{uw}a_{wv}}{\sqrt{\sum_w a_{uw}^2}\sqrt{\sum_w a_{vw}^2}}. \tag{1.9}$$

The second equality in Equation (1.9) is suggestive of a familiar operation from geometry: the cosine of the angle between two vectors. The numerator, $\sum_w a_{uw}a_{wv}$, is the dot product between the column vectors a_u and a_v of the adjacency matrix, and $\sqrt{\sum_w a_{uw}^2}$ is the norm. For this reason, Equation (1.9) is commonly known as *cosine similarity* or sometimes *Salton similarity* [114] after one of the first authors to propose this measure.

There are numerous other ways to normalize Equation (1.8) to derive different measures of similarity, each with its own merits. For example, Jaccard similarity [58] normalizes by the total number of distinct neighbors of both nodes, $d_u + d_v - n_{uv}$; effectively measuring the fraction of neighbors shared by both nodes. Alternatively, Hub-suppressed similarity [105, 77] normalizes using $\max\{d_u, d_v\}$ to reduce the influence of high-degree nodes.

1.2.4 Assortativity

Zooming out to examine the broader structural properties of networks, we find that connections between nodes often follow systematic patterns rather than occurring randomly. Networks frequently exhibit *assortative mixing*—or *homophily*—where significantly more edges join nodes that are correlated in some manner than nodes that are not. An example of this can be seen in Moreno's sociogram shown in Figure 1.4. The vast majority of edges are between students of the same gender—a highly improbable scenario if edges were distributed randomly. This pattern underlies the formation of *community structure* in networks, where nodes are grouped together based on shared characteristics. Community structure is a major area of research in network science, as understanding this structure provides insight into the network's behavior and organization. Returning to Moreno's sociogram, one can identify two communities by inspection: the boys and the girls, reflecting some of the associated gender dynamics of schoolyard play. There are a number of ways to measure the presence of community structure, *modularity* [84] being one of the most popular amongst researchers. Unlike the measures discussed previously, modularity is based on the use of *null models*, comparing how an observed network differs from one formed randomly. We will explore this measure further in the next section when we introduce random graphs models.

1.3 Random Graph Models

The measures introduced in the previous section can shed light on various aspects of network structure. However, they tell us little about *why* these structural features exist. Are they coincidences? Are these properties just noisy artifacts from the data collection process? To address these questions, researchers have developed theoretical network models to help distinguish random occurrences from meaningful patterns.

1.3.1 The Erdős-Rényi Random Graph Model

One of the most well studied and perhaps the simplest of random graph models is the $G\left(n,p\right)$ model. It was first studied by Solomonoff and Rapoport [117], but is most famously associated with the work of Paul Erdős and Alfréd Rényi, and is thus generally referred to as the Erdős-Rényi model [36]. This model is parameterized by the number of nodes n and a fixed edge probability p, defining an ensemble of simple networks. The probability of a particular network with adjacency matrix A is given by

$$P\left(A|n,p\right) = \prod_{u<v} p^{a_{uv}} \left(1-p\right)^{1-a_{uv}} \tag{1.10}$$

$$= p^m \left(1-p\right)^{\binom{n}{2}-m}. \tag{1.11}$$

The model does not fix the number of edges, just the probability, so networks drawn from $G(n,p)$ can differ in their number of edges. However, the expected number of edges is fixed and can be calculated. Equation (1.10) is a Bernoulli distribution with probability p, meaning that the expected value of an edge between a pair of nodes, $\langle a_{uv}\rangle$, is simply p itself. So,

$$\langle m\rangle = \left\langle \sum_{u<v} a_{uv} \right\rangle = \sum_{u<v} \langle a_{uv}\rangle = \binom{n}{2} p. \tag{1.12}$$

As discussed in Section 1.2.1, the degree of a node is a fundamental characteristic of local network structure. Indeed, under the Erdős-Rényi model, calculating the expected degree is straightforward. However, to better understand the broader behavior of Erdős-Rényi random graphs, it is more informative to examine the *distribution* of the degrees. For an arbitrary node u, the probability of connecting to any one of the other $n-1$ nodes is given by p. Thus, the probability that it is connected to one particular set of exactly k nodes is given by $p^k \left(1-p\right)^{n-1-k}$. If we want to know the probability of connecting to any arbitrary set of k nodes, we need to count the number of ways of choosing k nodes from

$n-1$, which is given by $\binom{n-1}{k}$. Thus, the total probability of having degree k is given by

$$P\left(d_u = k|\, n, p\right) = \binom{n-1}{k} p^k \left(1-p\right)^{n-1-k}, \tag{1.13}$$

meaning that the Erdős-Rényi model produces networks with binomially distributed degrees. In fact, one usually looks at networks where n is assumed to be large. If we write $p = c/(n-1)$, then a routine calculation shows that

$$\lim_{n\to\infty} P\left(d_u = k|\, n, p\right) = \frac{e^{-c}c^k}{k!}. \tag{1.14}$$

If we assume that the large networks we measure empirically can be well modeled by an Erdős-Rényi random graph, then we should expect that node degrees be Poisson distributed, sharply peaked around the mean $c = (n-1)p$. However, as mentioned in Section 1.2.1, many empirical networks actually exhibit heavy-tailed degree distributions and are therefore not adequately captured by a Poisson distribution. This suggests that the Erdős-Rényi model is not the most realistic model of real-world networks, limiting its utility as an analytical tool. This may seem obvious from the definition—one normally does not make friends by choosing the name of random person on earth out of hat—but now we have some theoretical justifications for why this may be.

1.3.2 The Configuration Model

Alternative models have also been studied that address some of the limitations of the Erdős-Rényi model. One example is the *configuration model* [17], which specifically tackles the issue of degree distribution. This model takes as input a sequence of node degrees, $\{d_u\}$, so that, as a hard constraint, all randomly sampled networks match the given degree sequence exactly. This in turn fixes the total number of edges in the network rather than just in expectation as in the Erdős-Rényi model.

In the configuration model, a node u has exactly d_u edge "stubs"—half of an edge attached to a node, but not yet to another. Therefore, there are $d_u d_v$ ways of pairing stubs between nodes u and v to form an edge. Since there are $2m$ stubs in the network, then in the limit of large m, the probability of an edge between nodes u and v is

$$p_{uv} = \begin{cases} \frac{d_u d_v}{2m} & \text{for } u \neq v, \\ \frac{d_u^2}{4m} & \text{for } u = v. \end{cases} \tag{1.15}$$

A closely related model that is slightly easier to treat analytically is the Chung-Lu model [28], where instead of specifying the degrees exactly, one specifies the expected degrees $\{\theta_u\}$ so that the probability of an edge is given by

$$p_{uv} = \begin{cases} \theta_u \theta_v & \text{for } u \neq v, \\ \frac{1}{2}\theta_u^2 & \text{for } u = v. \end{cases} \tag{1.16}$$

1.3.3 Random Graphs as Null Models

Neither the Erdős-Rényi nor configuration models capture many or most of the structural properties found in empirical networks. However, this is not necessarily a flaw; in fact, in many cases it is intentional. These models often serve as *null models*, providing a baseline for comparison. Null models effectively help answer the question:

Can some observed pattern be explained by random placement of edges alone?

In the case of the configuration model, one could be more specific and ask if it can be explained by a network's degree distribution. This approach enables us to test whether observed structural features emerge from the dynamics of the system being studied or if they simply arise from random chance.

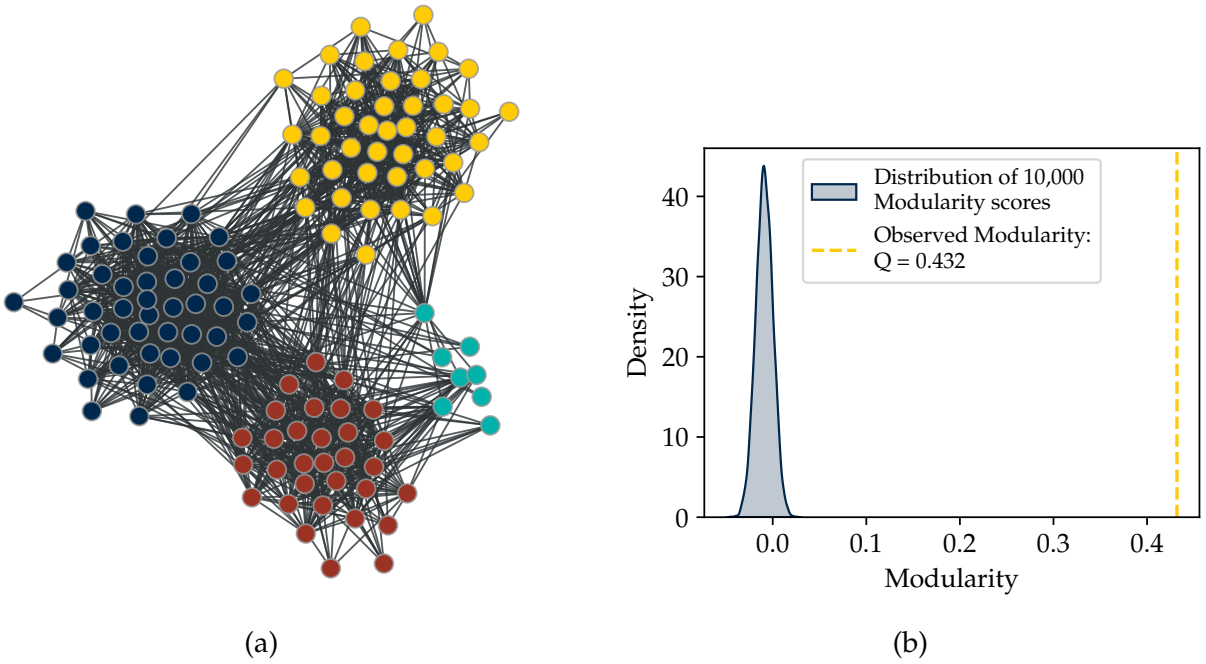
Perhaps one of the most popular applications of null models in network analysis is in the study community structure. In Section 1.2.4 we briefly mentioned that nodes in networks often form *groups* or *communities*. Loosely speaking, a community is a subnetwork within a larger network where nodes are more connected to each other than to those in other communities—more than would be *expected* by chance. Null models provide a precise definition for what is expected: the typical number of edges between nodes if the network were generated under the null model assumptions. Using the configuration model as a null model means that edges are placed randomly (while preserving node degrees), without considering attributes correlated with the nodes. Modularity, a widespread measure of community structure, employs the configuration model as its null model. This enables us to quantify the degree to which the number of edges within groups deviates from what would be expected in an equivalent network created by random edge placement. Its exact mathematical form is given by

$$Q = \frac{1}{2m} \sum_{uv} \left(a_{uv} - \frac{d_u d_v}{2m} \right) \delta_{g_u, g_v}. \tag{1.17}$$

Where g are the community assignments of the nodes, such that $g_u = r$ means that node u is in group r, and δ_{g_u, g_v} picks out the edges that fall between nodes of the same group.

Therefore, the first term in Equation (1.17) counts the number of within-group edges in the observed network. The second term is the expected number of edges between two nodes in the configuration model, again, the Kronecker-delta picking out only terms where nodes are in the same group. Ultimately, this quality function scores how well the edge placement aligns with a given partition of the nodes. The higher the score, the more *modular* the network appears, that is, the more it seems to naturally divide into groups.

Consider the contact network of French high school students [42] shown in Figure 1.6a. This network consists of 126 nodes and 1,710 edges, where nodes represent either high school students or teachers. The edges indicate that two individuals were in close proximity (between 1 and 1.5 meters) for at least 20 seconds. Each node is labeled to indicate



(a) (b)

Figure 1.6: Contact network of French high school students [42]. In (a) Nodes are colored by class: red, yellow, or blue for different student classes, and teal for teachers. In (b) Distribution of modularity scores for 10,000 networks with the same degree distribution and node labels. The vertical line indicates the modularity of the observed network.

whether it is a student or a teacher, and if a student, their class association. In the figure, different classes are denoted by the colors: red, yellow, and blue, while teachers are represented in teal. Using these categorical labels, we can calculate the modularity of the network, in this case $Q = 0.432$, illustrated by the vertical line in Figure 1.6b. Furthermore, observe the distribution of modularity scores for 10,000 randomly generated networks. These scores are concentrated near 0, while the modularity of the observed network is

much higher. This suggests that the degree distribution alone may not be sufficient to reproduce the modular structure of this network.

Assortative (or disassortative) structures are not the only properties that can be measured using null models. Another important property found in networks is *core-periphery* structure, which focuses more on grouping nodes based on their role or centrality in the network rather than categorical labels. A common definition of this structure is based on edge density: a core consists of nodes that are densely connected, whereas a periphery is made up of sparsely connected nodes, primarily linked to the core. This definition based on edge density aligns well with using the Erdős-Rényi random graph as a null model, as its parameter, p, controls the edge density in the network. Borgatti and Everett [19] formulated a quality function akin to modularity with the following form:

$$\rho = \frac{1}{2}\sum_{uv} a_{uv} g_u g_v - p\binom{n_p}{2}, \tag{1.18}$$

where n_p is the number of periphery nodes, and $g_u = 0$ if node u is in the core and $g_u = 1$ if its in the periphery. This function assumes an idealized core-periphery structure where all edges are between core nodes, which is represented in the first term of Equation (1.18). The second term is the expected number of edges in the periphery of a random network, analogous to how modularity's second term represents an expected number of edges of a random network, only assuming a different null model.

1.3.4 Community Detection

If we know the groups of the nodes in a network, then Equation (1.17) quantifies the extent to which those groups correlate with the connectivity pattern of the network. However, what if we did not know these groups beforehand, how could we determine if the network has community structure? This problem is called *community detection* and has been the subject of major research efforts in network science [40, 125, 41]. Optimizing modularity is one of the most widespread methods for detecting communities in networks. It involves finding a suitable division of the network that maximizes the modularity score, that is

$$\hat{g} = \operatorname*{argmax}_{g} Q(A, g). \tag{1.19}$$

Here, $\hat{g}$ are the group assignments that maximize the modularity score. In most applications, checking every possible partition g is impractical as there are k^n ways of dividing n nodes into k groups. As a result, a variety of optimization techniques have been de-

veloped to maximize Equation (1.17), including ones based on greedy heuristics, spectral clustering, and sampling amongst others [16, 83, 131, 113].

More recently, work on community detection has taken a different direction, employing generative models of community structure. In the same way that fitting a line to a set of data points helps researchers identify patterns, fitting a network to a generative model can reveal interesting aspects of network structure. One of the theoretical benefits of using such methods is their inherent explanatory power, as they require us to explicitly encode how network structure is generated. When studying community structure, this is done most often using the stochastic block model [55, 92] or one of its many variants [3, 63, 97].

In its most basic form, the stochastic block model is a direct generalization of the Erdős-Rényi random graph. One assumes that the network of n nodes divides into k groups. Edges are then placed randomly with probability p_{rs}, depending on the group memberships of the individual nodes in each pair—specifically, one node belonging to group r and the other to group s, more concretely:

$$P\left(A|p,g,k\right) = \prod_{u<v} p_{g_u,g_v}^{a_{uv}} \left(1-p_{g_u,g_v}\right)^{1-a_{uv}}. \tag{1.20}$$

If $p_{rs} = c$ is constant for all pairs of groups, then we recover exactly the Erdős-Rényi random graph model. As a generalization of the Erdős-Rényi model, one notable use case is in the identification of core-periphery structure. One assumes, like in the Borgatti and Everett quality function, that there are two labels a node u can take: $g_u = 0$ for the core and $g_u = 1$ for the periphery. If we constrain the probabilities in such a way that $p_{00} > p_{01} > p_{22}$ then this falls in line with our requirement that core nodes be more densely connected than periphery nodes. This method of studying core-periphery structure was proposed by Zhang *et al.* [130] and an extension allowing for multiple cores and peripheries is presented in Chapter 3.

In practice, Equation (1.20) can be challenging to treat analytically. As a result, researchers often adopt a slightly different formulation in which one places a Poisson distributed number of edges with mean ω_{g_u,g_v} between pairs of nodes u, v. This is a valid approach in the limit of large network size and is given by

$$P\left(A|\omega,g,k\right) = \prod_{u<v} \frac{\left(\omega_{g_u,g_v}\right)^{a_{uv}}}{a_{uv}!} e^{-\omega_{g_u,g_v}} \prod_{u} \frac{\left(\frac{1}{2}\omega_{g_u,g_u}\right)^{a_{uu}/2}}{(a_{uu}/2)!} e^{-\frac{1}{2}\omega_{g_u,g_u}}. \tag{1.21}$$

However, this variant of the model has similar shortcomings to the Erdős-Rényi random graph; in particular, it produces networks with Poisson degree distributions within each

group, making it unsuitable for applications to networks with heterogeneous degrees.

To address this limitation, the *degree-corrected* stochastic block model [63] was proposed, offering an approach that accommodates networks with broad degree distributions:

$$P\left(A|\omega,\theta,g,k\right)=\prod_{u<v}\frac{\left(\theta_u\theta_v\omega_{g_u,g_v}\right)^{a_{uv}}}{a_{uv}!}e^{-\theta_u\theta_v\omega_{g_u,g_v}}\prod_u\frac{\left(\frac{1}{2}\theta_u^2\omega_{g_u,g_u}\right)^{a_{uu}/2}}{(a_{uu}/2)!}e^{-\frac{1}{2}\theta_u^2\omega_{g_u,g_u}}. \tag{1.22}$$

This model incorporates the previously discussed Chung-Lu model by proposing that the mean number of edges between a pair of nodes be $\theta_u\theta_v\omega_{g_u,g_u}$, where the θ parameters control node degrees and the ω parameters control group connectivity. This variant allows for flexible inference, as one can model both disassortaive and assortative structures given by the appropriate constraints on the ω parameters while also preserving node degrees in expectation. For example, traditional assortative community structure emerges when ω_{rr} is greater than ω_{rs}, resulting in more within-group edges than between-group edges.

We have discussed how the stochastic block model generates networks with given community structure. But how can we achieve the reverse—how can we *discover* communities given an observed network? One approach is to use techniques from statistical inference. Assume that our network, represented by its adjacency matrix A, was generated using some statistical model dependent on group assignments g and other model parameters ϕ. Then, $P\left(A|g,\phi\right)$ is the model likelihood. Our goal, however, is to estimate these parameters *given* the network, as this is all the information we presumably have. Specifically, we aim to estimate $P\left(g,\phi|A\right)$, known as the posterior distribution, which is related to the likelihood by Bayes' rule

$$P\left(g,\phi|A\right)=\frac{P\left(A|g,\phi\right)P\left(g,\phi\right)}{P\left(A\right)}. \tag{1.23}$$

Here, $P\left(g,\phi\right)$ is called the *prior* distribution which which reflects our beliefs about the possible values that g and ϕ may take.

Equation (1.23) defines a probability distribution over all partitions of the network g. If our interests lie only in finding the partition that maximizes the likelihood of the network's generation by the model—akin to finding the line of best fit in linear regression—then it is often sufficient to, assuming $P\left(g,\phi\right)$ is constant, just maximize $P\left(A|g,\phi\right)$ directly. The degree-corrected stochastic block model requires several parameters other than the group partitions. When we focus solely on g, the parameters ω and θ are less central. To simplify the problem, we calculate their maximum likelihood estimates and then substitute them back into Equation (1.22) to obtain a profile likelihood that is only a function of g, our

parameter of interest. Karrer and Newman [63] showed that their maximum likelihood estimates are given by:

$$\hat{\theta}_u = \frac{d_u}{\Delta_{g_u}}, \qquad \hat{\omega}_{rs} = m_{rs}, \tag{1.24}$$

where $\Delta_r = \sum_u d_u \delta_{g_u,r}$ and $m_{rs} = \sum_{uv} a_{uv} \delta_{g_u,r} \delta_{g_v,s}$. These are just the fraction of degrees in the group of node u that belong to u and the number of edges running between groups r and s respectively. The resulting profile log-likelihood now only depends on g,

$$\mathcal{L} = \sum_{rs} m_{rs} \log \frac{m_{rs}}{\Delta_r \Delta_s}. \tag{1.25}$$

Equation (1.25) can then be maximized via your method of choice over all partitions g.

To make full use of the Bayesian framework implied by Equation (1.23), one can be more specific about the prior distributions on the model parameters and then calculate a *marginal likelihood* by integrating (or summing) them out. For example, an important characteristic of real-world networks is that they tend to be sparse. That is, while a network with n nodes has roughly n^2 places to put an edge, they usually have much fewer than this, often on the order of n edges. This suggests that using a prior that incorporates sparsity, instead of a constant prior—as in the maximum likelihood approach, where very few and very many edges are equally probable—could be beneficial. In Section A.3, we show in detail how to incorporate priors in the stochastic block model for inference.

Additionally, the Bayesian framework allows us to consider a distribution of highly probable parameter values, rather than just a single estimate. This flexibility provides a more complete picture of the likely parameter values while quantifying their associated uncertainty. This, however, introduces its own challenges, primarily in sampling from the posterior distribution. Various mathematical and computational techniques, such as Markov Chain Monte Carlo (MCMC) methods, have been developed to approximate the posterior, forming the basis of much of the more recent work on the inference of large-scale network structure [90, 107, 85].

1.4 Link Prediction

Until now, our discussion has focused on the structural properties that can be measured in networks and the random graphs used to model these features. We will now move on to a specific application of these ideas and leverage them to make predictions.

In many fields, data are often incomplete or subject to measurement errors, network

data being no exception. *Link prediction* is the process of identifying edges in an empirically measured network that are likely to exist but absent in the currently observed network. Link prediction methods aim to detect edges that are most likely to be missing, based on the patterns observed in the edges that have been measured. Suppose we have an observed

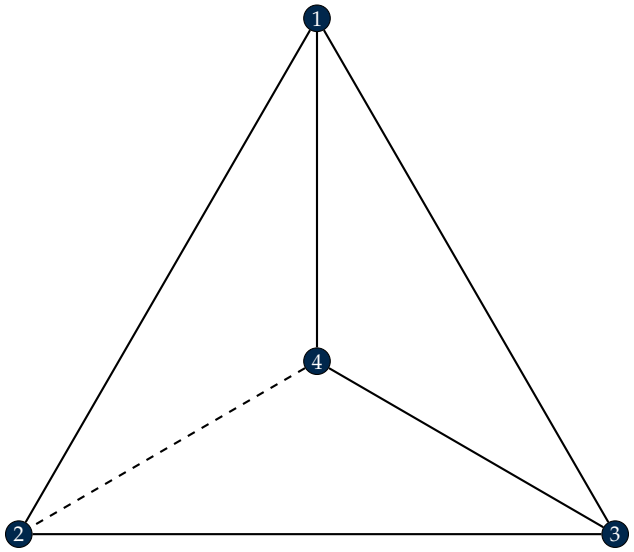


Figure 1.7: If we assume a network is incompletely measured, the goal is to predict which edges are missing. The measured edges are illustrated as solid lines while the edge to be predicted is dashed.

network of n nodes represented by its adjacency matrix A^o. We assume that this network is inherently incomplete and that the existence of missing edges can be predicted, better than random guessing, based on correlations within the network data. Generally, predicting missing network attributes, such as node communities, is an easier task than predicting edges, as we can often identify when such attributes are absent. However, predicting edges is more difficult because not every pair of unconnected nodes is necessarily missing. The challenge lies in determining which potential edges are genuinely missing.

All network-based link prediction methods adopt basically the same framework. One computes some prediction score for every pair of nodes, which indicates, often on an arbitrary scale, how likely those nodes are to be connected by an edge, with higher values denoting higher likelihood. Then, one sorts those values in decreasing order to create a list of potential missing edges from most likely to least. A good method will assign higher scores to "actual" missing edges than would be expected if pairs were scored randomly.

1.4.1 Topological Methods

Link prediction methods based on local network topology use the small-scale network structure to assign scores to node pairs. One common way to do this is to use some of the similarity measures discussed in Section 1.2.3 as well as methods based on node degrees and path lengths [75]. Consider the networks shown in Figure 1.8: suppose, for the sake of

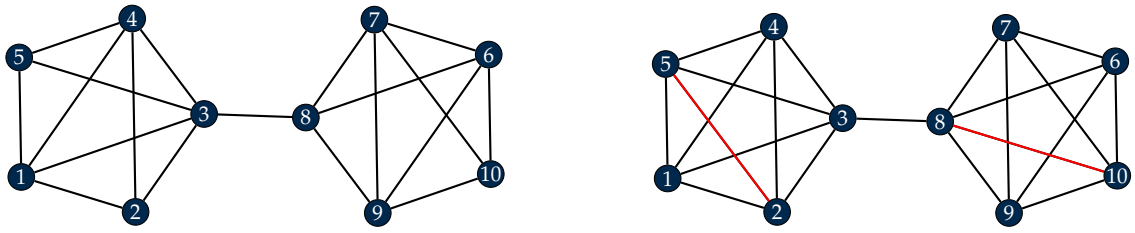


Figure 1.8: Two networks, observed and actual. In (a) we have an example observed network. In (b) we show the true network, predicted edges depicted in red.

example, we observe the network in Figure 1.8a, but we know that the true network is the one shown in Figure 1.8b. We can predict the missing edges by calculating the similarity score between every pair of unconnected nodes and taking the highest scoring pairs to be the ones most likely to be missing. In Table 1.1 we tabulate the top 10 scoring pairs; notice how our two missing edges have the highest score, while the pair $(1, 6)$ has a score of 0.0, since they do not share any neighbors. This is the basic idea behind the use of network topological features to predict missing edges.

Vertex Pair	Cosine Similarity
(2, 5)	1.00
(8, 10)	0.87
(2, 8)	0.29
(5, 8)	0.29
(1, 8)	0.25
(4, 8)	0.25
(3, 6)	0.22
(3, 7)	0.22
(3, 9)	0.22
(1, 6)	0.00

Table 1.1: Cosine Similarity Scores for unconnected vertex pairs

1.4.2 Probabilistic Methods

Another common approach to predicting missing edges involves using network models such as the stochastic block model discussed in Section 1.3.4. As mentioned previously, employing network models is analogous to fitting a line to data. Just as in curve fitting, where we are interested in making predictions about unobserved data beyond just estimating parameters like the slope and y-intercept, we can use network models to fit networks and predict unobserved edges. This basic framework was adopted by Clauset *et al.* [29] and was further applied specifically to the stochastic block model by Guimerà and Sales-Pardo [49]. The approach works as follows: our observed network is represented by the adjacency matrix A^o, which is assumed to be generated from a model with parameters Φ, given by $P(A^o|\Phi)$. In the degree-corrected stochastic block model, for example, $\Phi = (\omega, \theta, g, k)$. Our goal is to predict the network with the missing edges added, represented by the adjacency matrix A, *given* only our observed network A^o. More concretely, we aim to derive an expression for $P(A|A^o)$, which can be calculated directly from the definition of conditional probability[3]

$$P(A|A^o) = \int_\Phi P(A|A^o, \Phi)\, P(\Phi|A^o) = \int_\Phi \frac{P(A, A^o|\Phi)}{P(A^o|\Phi)} P(\Phi|A^o)\,. \tag{1.26}$$

The value of $P(A|A^o)$ then serves as a score indicating the likelihood that a pair of unconnected nodes should be connected. This approach is part of a broader process of network reconstruction, where model-driven predictions aid in filling gaps within the observed network structure [29, 87, 98, 120].

1.4.3 Network Embedding Methods

Alternative approaches to link prediction make use of a hitherto unexplored type of structure: latent-space structure. These methods aim to place nodes at positions in a Euclidean space such that nodes with similar positions are more likely to be connected by edges. For a network $G = (V, E)$, the goal is to find a mapping from the node set to a k-dimensional space, $f : V \to \mathbb{R}^k$. While there is flexibility in choosing f, in practice, one typically aims to preserve properties of the network in the embedding space such as node similarity, σ_{uv}, or even the adjacency matrix itself. To reconstruct the information obtained from the network, a second function, $\phi : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}$, is defined on the

[3]Here, Φ represents an arbitrary set of model parameters including ones that could be discrete. For notational clarity, we have chosen $\int_\Phi$ to indicate marginalization over these parameters, with the understanding that integration or summation should be applied as appropriate to the context.

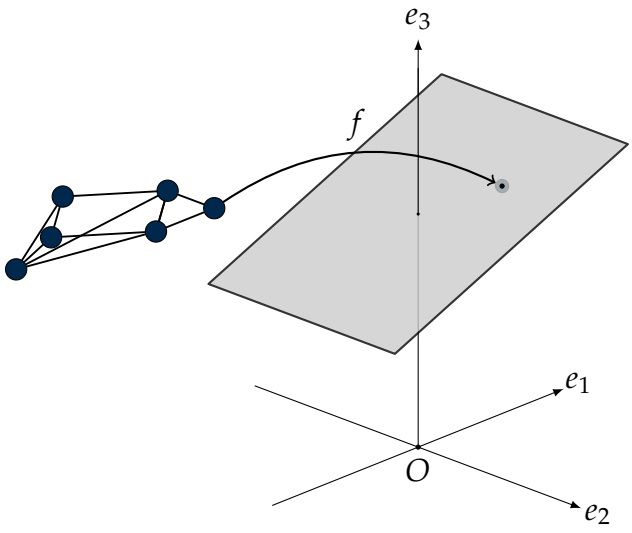


Figure 1.9: Visualization of embedding a network into some latent space. The embedding function f maps nodes in the network to a k-dimensional space, where f is learned by optimizing Equation (1.27).

embedding space. This function usually takes a simple form, such as the L_p norm of the difference between embedding vectors $\left\|f_u - f_v\right\|_p$ or some function of their inner product $h\left(f_u^{\mathrm{T}} f_v\right)$. Determining f can then be framed as the following optimization problem:

$$\underset{f}{\operatorname{argmin}} \sum_{uv} \ell\left(Q_{uv}, \phi(f_u, f_v)\right). \tag{1.27}$$

Here ℓ is an objective function characterizing the difference between some statistic Q measured on the network and its reconstruction ϕ in the embedding space. This framework is often called the *encoder-decoder model* in graph representation learning [50], where f acts as the encoder and ϕ the decoder.

A fairly standard application of this framework is the *singular value decomposition (SVD)*. This matrix-based method constructs a low-rank approximation of the adjacency matrix, A. It assumes the adjacency matrix can be factorized and written in the form: $A = U\Sigma V^{\mathrm{T}}$, where U and V are unitary matrices[4], and Σ is the diagonal matrix of singular values. By discarding (i.e., setting to zero) all but the k largest singular values, we form a modified diagonal matrix Σ', which we then use to compute the rank-k matrix $A' = U\Sigma' V^{\mathrm{T}}$. The encoder in this case is $f_u = \left(U\Sigma'^{1/2}\right)_u$ and the decoder is simply $\phi\left(f_u, f_v\right) = f_u f_v^{\mathrm{T}}$, which is precisely the element of the reconstructed adjacency matrix a'_{uv}. To perform link pre-

[4]For real symmetric matrices such as the adjacency matrix of an undirected network, $U = V$ in the singular value decomposition. It is equivalent, up to sign, to the eigenvalue decomposition.

diction, the value of $\phi\left(f_u, f_v\right)$ is used to predict the missing edges—larger (more positive) elements indicating higher prediction certainty. If we apply this method to the network shown in Figure 1.8a, we can visualize the estimated latent positions of the nodes by making a rank-2 approximation—using only the first two singular values—of its adjacency matrix. As depicted in Figure 1.10, the nodes cluster into two groups that correspond to the network's nearly complete cliques. This illustrates how nodes closer together in latent space are more likely to have an edge connecting them, including the pairs we aim to predict.

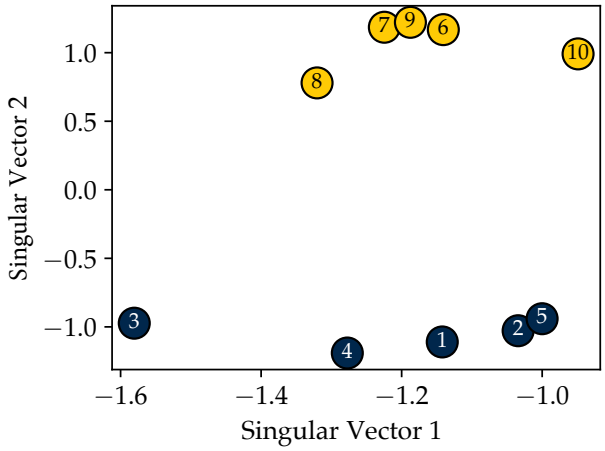


Figure 1.10: Estimated latent node positions for the network in Figure 1.8a using the first two singular values of its adjacency matrix.

There are, however, many ways to construct a network embedding. Some more sophisticated approaches are based on random walks of fixed length on networks. These random walks begin a certain node and move to a neighboring one at each step, forming sequences of visited nodes that capture local network structure. These sequences are then fed to a neural network to construct an embedding. In the language of the encoder-decoder framework, $Q_{uv} = P_t\left(v|u\right)$ is the probability of reaching node v starting at u on a walk of length t. The encoder is the trained neural network and the decoder is the inner product of the node embedding vectors. This is the underlying approach behind popular graph embedding algorithms such as DeepWalk and node2vec [101, 48].

1.4.4 Evaluating Performance

Having defined link prediction and provided an overview of common methods, we now shift our focus to the metrics used to evaluate their performance. It is convenient to first view the problem as a binary classification task, allowing us to adopt metrics commonly

used in supervised machine learning. The observed network serves as the training data, with edges as the positive class (P) and nonedges as the negative class (N). A standard technique for evaluating binary classifiers is cross-validation, where a random portion of the data is removed and later used to test the classifier's predictive accuracy. This provides us with a robust framework to assess the performance of all our link prediction algorithms and compare them to each other.

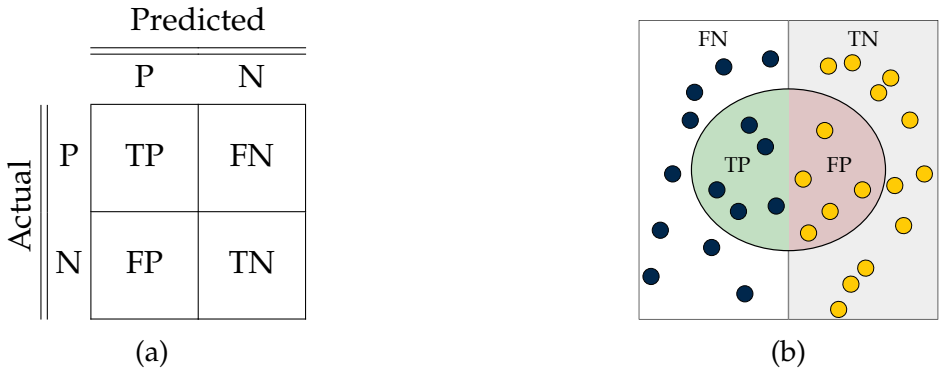


Figure 1.11: (a) Contingency table for a binary classifier. In (b) we give a visualization of classifier predictions. The ellipse contains the positive *predictions* of a classifier, surrounded by negative predictions. The circles are the data with color distinguishing their ground truth class; blue being positive and yellow negative.

As discussed in Section 1.4, link prediction algorithms output a score indicating the likelihood that a pair of nodes is joined by an edge. To determine the predicted class, a threshold, t, must be set such that pairs with scores above this threshold are considered positive (edges), and those below are considered negative (nonedges). With this threshold in place, predictions on the test set can be categorized into four different groups:

- True Positives (TP): The predicted class and the actual class are positive.
- False Positives (FP): The predicted class is positive, but the actual class is negative.
- True Negatives (TN): The predicted class and the actual class are negative.
- False Negatives (FN): The predicted class is negative, but the actual class is positive.

The relationship between the predicted and ground truth labels can be tabulated in a *contingency table*—also called a *confusion matrix*—illustrated in Figure 1.11. These fundamental elements help us to derive crucial metrics for evaluating the performance of binary classifiers.

Perhaps the simplest threshold-dependent metric is *accuracy*. It represents the fraction of all correct predictions made by the classifier and can be expressed in terms of the elements of the contingency table as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}. \tag{1.28}$$

In link prediction, we are primarily concerned with accurately predicting the positive class. Two metrics that capture the classifier's performance on this class are the *true positive rate* —often called *recall* or *sensitivity*—and the *false positive rate*. The true positive rate is the fraction of positives correctly predicted as positives, while the false positive rate is the fraction of negatives incorrectly predicted as positives. From the contingency table, they are expressed as:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{1.29}$$

$$\text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}}. \tag{1.30}$$

Both accuracy and recall can be misleading in imbalanced data sets like that of network data[5]. Accuracy might appear high simply by predicting the majority class (nonedges), while recall does not account for false positives. This can lead to an overly optimistic assessment of performance if many negative instances are incorrectly classified as positives.

An arguably better indicator of performance on the positive class is known as the *precision* or *positive predictive value*. This is the fraction of correct positive predictions, given by:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{1.31}$$

This metric is penalized by false positives meaning classifiers that predict many positive instances but also include a significant number of false positives will score low on precision. This is in contrast to recall, which considers only the actual positives identified.

The interplay between precision and recall highlights an important trade-off. Increasing recall by predicting more positives can result in lower precision if those predictions include many false positives. In contrast, setting a stringent threshold can lead to high precision, but many genuinely positive instances may be missed, lowering recall. Thus, improving one often comes at the cost of the other, and balancing these metrics typically

[5]Networks are sparse, there are disproportionately more places to put an edge than there are edges. This results in many more negative instances than positive ones.

depends on the specific application. For example, consider a network of drugs and diseases, where an edge indicates that a drug treats a disease. A pharmaceutical company seeking to repurpose drugs and optimizing for cost will prioritize high precision to ensure only the most promising drug-disease pairs are pursued. Conversely, a nonprofit research institution might prioritize high recall to explore a wide range of potential pairs, minimizing the risk that a viable pair might be overlooked. In short, precision is about the relevance of the results and recall about the completeness of the results, with the two often being at odds with each other.

The previously defined metrics are all dependent on some chosen threshold, t. In practice, selecting the optimal threshold is often challenging, and comparing different classifiers at a single threshold may not provide a complete picture of their relative performance. Threshold-free methods, such as the *receiver operating characteristic (ROC)* curve are often employed to address this. The ROC curve is constructed by plotting the TPR against the FPR; it is a parametric curve, parameterized by t, such that every point on the curve is given by $(\mathrm{FPR}(t), \mathrm{TPR}(t))$. A curve of similar construction is the *precision-recall (PR)* curve, where instead the precision is plotted as a function of the recall (TPR) for all thresholds t. These curves provide similar insights to recall and precision, but have the advantage of being threshold-free, allowing for more robust comparisons between classification models and a more comprehensive assessment of performance.

While the ROC and PR curves provide important insights into model performance, it is often useful to have a single number to summarize the overall performance of a classifier. This can be done by calculating the area under these curves, known as the *area under the curve*. For the ROC curve, this is the AUROC which can be thought of as a measure of how thorough or complete an algorithm's predictions are. If we consider a removed edge to be successfully predicted when an algorithm ranks it higher than the average nonedge, the AUROC score equals the fraction of edges successfully predicted. Values run from 0.5 to 1, with higher values being better, a value close to 1 indicates an algorithm that gives very complete results and misses few of the predictions it aims to make. However, similar to recall, AUROC is insensitive to false positives diluting the results; the presence of many incorrect predictions among the correct ones does not significantly affect the score. Similarly, the area under the PR curve, AUPR, averages precision over values of the recall (i.e., the true positive rate), automatically weighting the results towards the most promising predictions (those with highest prediction score), as recall tends to change most rapidly at the beginning of predictions.

These are just a few of the metrics available to evaluate link prediction. Rarely is any single one of them sufficient, and in practice, several metrics are used together to paint a

more complete picture of performance.

1.5 Contributions

This chapter reviews some of the basic definitions and methods used in network science that form the basis of the contributions presented in the remainder of this thesis. The following chapters will discuss work that leverages large-scale network structure to gain insights from empirical network data. We will explore a practical use case for link prediction along with novel models for large-scale network structures and their inference techniques.

In Chapter 2, we motivate the use of network-theoretic techniques with an application of link prediction to the problem of drug repurposing. Repurposing existing drugs to treat new diseases is a cost-effective alternative to *de novo* drug development, but there are millions of potential drug-disease combinations to be considered with only a small fraction being viable. In silico predictions of drug-disease associations can be invaluable for reducing the size of the search space. This work presents a novel network of drugs and the diseases they treat, compiled using a combination of existing machine-readable and textual databases, natural-language processing tools, and hand curation, and analyze it using a selection of network-based link prediction methods to identify potential drug-disease combinations. We measure the efficacy of these methods using cross-validation tests and find that several methods, particularly those based on graph embedding and network model fitting, achieve impressive prediction performance, with area under the ROC curve above 0.95 and average precision almost a thousand times better than chance.

In Chapter 3, we return to the study of large-scale network structures, focusing on core-periphery structure. We use inference methods based on a flexible network model that allows for traditional onion-like cores within cores, but also for hierarchical tree-like structures and more general non-nested types of structure. We propose an efficient Monte Carlo scheme for modeling observed networks and report results using a selection of real-world data sets. Additionally, we observe an empirical distinction between networks showing traditional core-periphery structure with a dense core weakly connected to a sparse periphery, and an alternative structure in which the core is strongly connected both within itself and to the periphery. Networks vary in whether they are better represented by one type of structure or the other. We also observe structures that are a hybrid between core-periphery structure and community structure, in which networks have a set of non-overlapping cores that correspond roughly to communities, surrounded by a single undifferentiated periphery.

In Chapter 4, we look at community structure in hypergraphs using flexible statistical inference methods that allows one to infer not only a partition of nodes into communities but also the number of communities. We formulate a Bayesian model from which we sample using a Markov chain Monte Carlo scheme with parallel tempering for adequate exploration of the probability landscape. We fit our model to empirical hypergraphs with a variety of hyperedge edge sizes and number of communities. Our results suggest that this model is competitive with other methods for community detection, being able to accurately recover the ground truth communities of the nodes.

Lastly, in Chapter 5 we provide our conclusions, reflecting on the work presented here and future directions for research.

Chapter 2

Drug-disease Networks and Drug Repurposing

In the previous chapter, we discussed various measures and metrics used to understand the structure and function of networks, along with theoretical approaches for capturing aspects of real-world network phenomena. In this chapter, we apply some of these techniques to a network data set, demonstrating a non-trivial use case of network analysis to explore questions of significant scientific importance, specifically drug repurposing.

2.1 Introduction

Drug repurposing, the practice of finding new uses for established medications, is a vital part of the pharmaceutical development landscape [93, 61, 104]. A fundamental part of the repurposing process is the identification of promising candidate drugs, and a significant amount of effort has been invested in the development of computational and statistical methods for performing this task [24, 123, 15, 76]. We approach the problem tying together many of the analytical methods discussed in Chapter 1, viewing it as a link prediction problem [75] on a network of drugs and the conditions they treat.

In this chapter we do two things: first, we discuss the compilation of an extensive network data set of drugs and the diseases they treat. The compilation uses a combination of existing data, both machine-readable and textual, computational natural language processing, and human curation and data cleaning. The resulting network represents a total of 2,620 drugs and 1,669 diseases. It differs from previous drug-disease data sets in being larger and more complete, and in being based solely on explicit therapeutic drug-disease indications, avoiding the use of associations inferred indirectly from, for example, drug function, targets, or structure.

Second, armed with our data set we apply network methods to predict potential new therapeutic drug-disease pairs. The data set takes the form of a *bipartite network*, having two types of nodes, representing drugs and diseases, and, as discussed in Section 1.1.3,

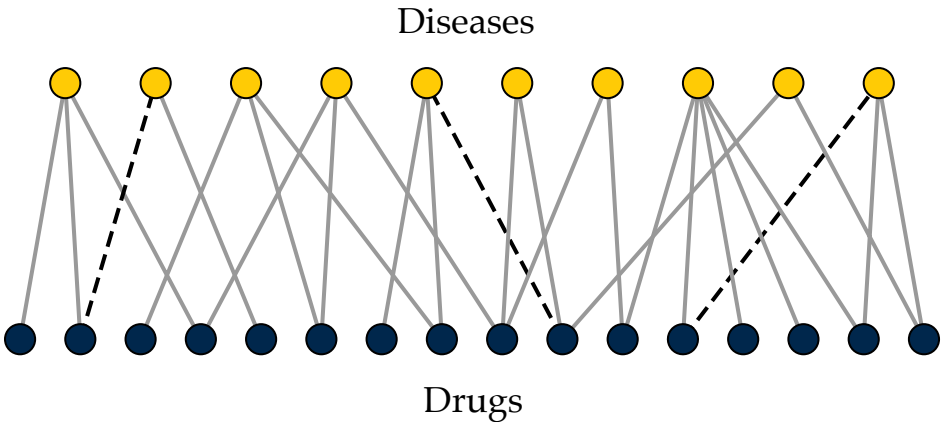


Figure 2.1: A bipartite network of drugs and diseases of the type considered here. Nodes in the network are of two types—drugs and diseases—and edges connect only nodes of unlike types to indicate which drugs are indicated for treatment of which diseases. The network is assumed to be incomplete, so some edges that should be present are missing from the data (dashed lines). Our goal is to identify these.

connections only between unlike kinds: a connection, or edge, between a drug node and a disease node means that the drug is indicated for treatment of the disease—see Figure 2.1. Using this network, we identify candidates for drug repurposing by applying *link prediction* methods.

Building on these works, in this chapter we take a selection of link prediction algorithms, extended where necessary to bipartite networks, and apply them to our network of drug-disease interactions. We run extensive cross-validation tests to quantify the performance of each algorithm, removing a small fraction of edges at random from the network and testing the algorithm's ability to tell which ones were removed.

Our primary finding is that a subset of the algorithms perform well on this task, showing an impressive ability to pinpoint the missing edges in the drug-disease network, as quantified by standard measures. The best of the methods we consider can achieve a measured area under the ROC curve in excess of 0.95 and average prediction precision almost a thousand times better than chance (although not in the same algorithm). Moreover, this performance is achieved using purely network-based methods—no pharmacological input, other than the network itself, is used. It is reasonable to suppose that a combination of network-based prediction and pharmacological insight could improve the performance still further and our methods could thus be used either alone or as part of a hybrid prediction strategy.

A number of previous authors have considered network methods for drug repurpos-

ing [47, 121, 57, 129, 1, 31], although they have taken somewhat different approaches to the one presented here. Gottlieb *et al.* [47] assembled a network similar to ours but smaller—about a quarter of the size—using some of the same resources but different methodology. They perform link prediction using similarity-based methods akin to the methods we describe in Section 2.3.1, and find moderately good performance, as we also do, though not competitive with the more sophisticated machine learning algorithms we consider. Wang *et al.* [121] took a somewhat similar approach with a small study based on an early version of the DrugBank database and employing a collaborative filtering algorithm that exploits network projections. Although they make only a small number of predictions, their work clearly shows the promise of these types of techniques. Huang *et al.* [57] offer a good example of indirect inference of drug-disease interactions, combining measures of disease similarity derived from text mining with measures of drug similarity from chemical and protein-protein interaction data, and predicting drug-disease combinations using a network label propagation algorithm. They obtain a number of medically relevant predictions, although overall measures of prediction performance are quite low. Zhang *et al.* [129] studied previously published data sets of known drug-disease associations along with pharmacological data such as structure, targets, and drug interactions, and proposed a new algorithm that combines these elements to predict drug-disease interactions. A limitation of this work is that the data include not only therapeutic interactions but also other drug-disease associations such as side effects, and hence the predictions also include non-therapeutic associations. Despite the incorporation of pharmacological information, the method of Zhang *et al.* performs less well by standard measures than some of the competing approaches. Abbas *et al.* [1] applied a range of link prediction methods, including some of the same ones we consider, to various pharmacological networks, including drug-target and drug-interaction networks. They were able to predict drug-disease associations indirectly by searching for drugs with relevant targets, but the methods they consider are not competitive with the best ones described here.

The remainder of this chapter is organized as follows. In Section 2.2 we introduce our data set and describe the methods used to assemble it. In Section 2.3 we describe the link prediction algorithms for bipartite networks that we employ, some of which are previously published while others are new but similar to previous methods for unipartite networks discussed in Section 1.4. Section 2.4 presents the results of our cross-validation tests and a comprehensive set of measurements of algorithm performance. We also report some formal results on link prediction performance that allow us to place bounds on the number of missing edges in the network and hence tell us about the quality of the data set and the number of potential opportunities for drug repurposing. In Section 2.5 we give

our conclusions.

2.2 Drug-disease Network

Our first task is the construction of the bipartite network of drugs and the diseases they treat, which involves a number of steps. As discussed in the introduction of this chapter, our network is based solely on known therapeutic drug-disease associations. The starting point is the DrugBank database (version 5.1.10, circa 2024) [122], an online index of over 15,000 drugs, with targets, chemical data, prescribing information, and other details. Many of these drugs are experimental or of dubious therapeutic value and we remove a significant number from the set, including drugs not approved for clinical use, drugs labeled as supplements, cosmetics, food or food additives, household products, allergens, or contrast agents, and drugs belonging to no known category. A full description of which drugs are removed is given in Appendix A.1.

Absent from the DrugBank database is a concise list of the conditions each drug is used to treat. To obtain such a list we use a combination of strategies. Some drugs in the database are accompanied by a Unique Ingredient Identifier or UNII code, an alphanumeric code that allows cross-listing with other databases. Using these identifiers we query another database, the NIH NCATS Inxight database [116], which includes machine-readable information on drug indications. DrugBank entries without a UNII code, or for which no uses are listed on Inxight, are further divided into two groups, those for which DrugBank contains human-readable indications, and those for which it does not. Drugs in the latter category are queried on the DailyMed database, another, smaller US federal database, which contains human-readable indications for some drugs. If no indications are found the drug is discarded.

This leaves us with a subset of drugs with human-readable indications, either from DrugBank or from DailyMed, typically in the form of a few sentences of English text. These indications are parsed using the OpenAI large language model (LLM) GPT-3.5 to return a machine-readable list of diseases and conditions for each drug. Some drug indications contain no disease information that the LLM is able to extract and these are discarded. Finally, every interaction in the network is checked by hand to confirm its correctness and to remove duplicate entries and other errors, for example: "Type II diabetes" and "Diabetes mellitus 2" appear as separate entries but refer to the same disease.

The final data set consists of 2,620 drugs and 1,669 diseases, with 8,946 edges connecting drugs to the diseases they are known to treat. This leaves over 4.3 million unconnected drug-disease pairs. It is our goal to predict which among these are the most promising

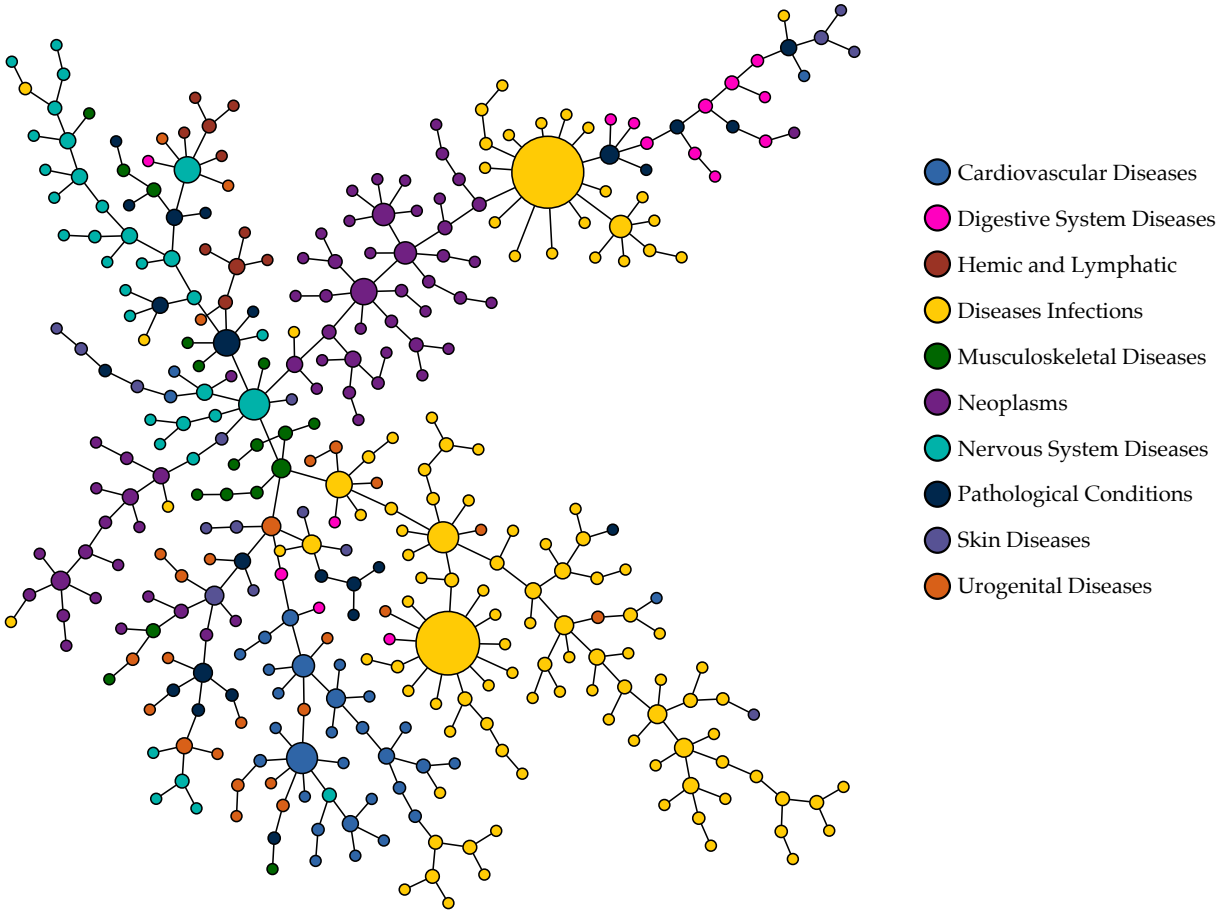


Figure 2.2: Visualization of the network of diseases. In this figure, we show a subset of the diseases in our network, those falling in the ten most common disease categories, as labeled. Connections between diseases represent disease pairs that are treated by one or more of the same drugs and only the strongest connections are shown (disease pairs treated by the largest numbers of common drugs).

candidates for drug repurposing. A visualization of a portion of the network, projected onto the diseases only, is shown in Figure 2.2.

2.3 Link Prediction Revisited

As discussed in Section 1.4, link prediction [75] is the task of estimating which edges are missing from an observed network. In the context of our drug-disease network this is equivalent to predicting specific drugs that can be used to treat specific diseases. Link prediction methods can in principle make use of a variety of ancillary information about network nodes and their characteristics. For instance, one can imagine a method that

employed chemical or structural information about drugs, clinical knowledge such as prescribing practice, or information about drug targets. A number of techniques that use such information for identifying drug repurposing candidates have been proposed in the past [24, 57, 129, 1, 13, 21]. Here, however, we take a purely network-driven approach that employs only the topological information contained in the network of drugs and diseases.

All network-based link prediction methods adopt basically the same framework. One computes some "prediction score" for every pair of nodes, which indicates, often on an arbitrary scale, how likely those nodes are to be connected by an edge, with higher values denoting higher likelihood. Then one sorts those values in decreasing order to create a list of potential missing edges from most to least likely. Typically, interest will focus on the initial entries in the list, which represent the most promising candidates, and a good link prediction algorithm is one for which a large number of these initial entries turn out to be correct predictions.

The heart of the algorithm is the definition and calculation of the prediction scores. Here we consider a variety of possible scores, drawing inspiration from various sources, including previously proposed algorithms for unipartite networks, representation learning methods, probabilistic models of network structure, and methods developed for the related problems of collaborative filtering and document classification. In detail the methods we consider are as follows.

2.3.1 Elementary Algorithms

In some contexts successful link prediction can be performed using surprisingly simple heuristics. Liben-Nowell and Kleinberg [75] have tested a range of such methods based on node degrees, path lengths, and network similarity measures. Perhaps the simplest of these methods is the degree-based heuristic they call "preferential attachment," after the well-known class of network growth processes by that name. Under this method, the prediction score for a drug-disease node pair u, v is simply equal to the product $d_u d_v$ of the degrees (i.e., the number of edges) at each of the nodes.

Another group of elementary algorithms are those based on node similarity discussed in Section 1.4.1. We have previously discussed similarity measures in Section 1.2.3, however, only in the context of unipartite networks. Generalizing this notion to bipartite networks involves some subtleties. We want to predict edges between nodes of unlike kinds (drugs and diseases in our application), but by definition such nodes do not share any neighbors: a disease node only has drug neighbors and a drug node only has disease neighbors. Instead, therefore, we define a prediction score between a drug node u and

a disease node v to be the sum of the similarities between v and other diseases that u is known to treat. In other words, our link prediction algorithm looks for diseases that are similar to those for which u is already used.

In mathematical terms, we define $\sigma(v, v')$ to be the similarity between disease nodes v and v', and $N(u)$ to be the set of diseases that u is known to treat. Then the prediction score for drug u to treat disease v is

$$x(u, v) = \sum_{v' \in N(u)} \sigma(v, v'). \tag{2.1}$$

One can also imagine defining prediction scores based on similarity between drugs: we could consider a particular disease v and look for drugs that are similar to ones currently used to treat it. This would give the alternate definition

$$x(u, v) = \sum_{u' \in N(v)} \sigma(u, u'). \tag{2.2}$$

We have experimented with both approaches, but find that Eq. (2.1) gives distinctly superior performance to Eq. (2.2), so we do not pursue (2.2) further here.

Equation (2.1) can be implemented with any of the many standard measures of node similarity in networks. We here consider the five with definitions given in Table 2.1: common neighbor count, cosine similarity, Jaccard coefficient, Dice (or Dice-Sørensen) coefficient, and the hub-suppressed index. All of these are based on the number of common neighbors between nodes, as described above, or equivalently the number of paths of length two. In addition, we consider one further measure, which we call the "Katz similarity" in recognition of its close similarity to the well-known Katz centrality [64], which counts paths of all lengths, but with less weight given to longer ones [73]. We have experimented with some other common similarities as well, but find that they give clearly inferior results, so we do not pursue them.

2.3.2 Machine Learning

The algorithms of the previous section are simple heuristics that do not attempt to make use of any deeper structure in the network, but a variety of other algorithms have been proposed that employ machine learning methods to extract more complex structural information by calculating network embeddings. Here we consider a representative set of algorithms from this class, including standard algorithms and some perhaps less familiar recent proposals.

Similarity measure	Formula
Common neighbors	n_{uv}
Cosine similarity	$\frac{n_{uv}}{\sqrt{d_u d_v}}$
Jaccard coefficient	$\frac{n_{uv}}{d_u + d_v - n_{uv}}$
Dice coefficient	$\frac{n_{uv}}{\frac{1}{2}(d_u + d_v)}$
Hub-suppressed index	$\frac{n_{uv}}{\max(d_u, d_v)}$

Table 2.1: Definitions of similarity measures $\sigma(u, v)$ between network nodes used in Eq. (2.1). In these expressions, u and v are nodes, d_u and d_v are their degrees (the number of edges connected to them), and n_{uv} is the number of common network neighbors between u and v. For nodes of degree zero, all of these measures (except the common neighbors measure) give $0/0$, in which case the similarity is defined to be zero.

Singular value decomposition (SVD): We discussed this method in Section 1.4.3 applied to the adjacency matrix of a network. However, applying this to a network's incidence matrix is exactly the same process. Recall that the incidence matrix is the rectangular matrix with elements $b_{uv} = 1$ if drug u is indicated to treat disease v and 0 otherwise. In this algorithm, one constructs a low-rank approximation to the incidence matrix B of the bipartite network. First, one computes the singular value decomposition $B = U\Sigma V^T$, then one zeros all but the k largest singular values, giving a modified diagonal matrix Σ', and then computes the rank-k matrix $B' = U\Sigma' V^T$. The elements of the resulting matrix are used to predict the missing edges, with larger values suggesting better predictions. We use the SVD implementation in the LAPACK linear algebra library. We have tested the algorithm for various values of k and find the best results around $k = 60$.

Probabilistic latent semantic analysis (PLSA): Another approach related to topic modeling is one in which we assign vectors $\vec{r}_u$ to nodes of one type and $\vec{s}_v$ to nodes of the other type, and the edge between u and v is a random variable with probability equal to the inner product $\vec{r}_u \cdot \vec{s}_v$. The dimension k of the vectors is a free parameter that can be tuned to give optimal results. This approach has been particularly employed in document classification, leading to the method known as probabilistic latent semantic analysis or PLSA [53]. Here we retask this method for our drug-disease network and use an expectation-maximization (EM) algorithm to fit the PLSA model, then use the probabilities $\vec{r}_u \cdot \vec{s}_v$ to predict the most likely missing edges. We find best results for vector dimension around $k = 90$.

Non-negative matrix factorization (NNMF): The embedding used for PLSA can be thought of as an approximate decomposition of the incidence matrix into a product of two non-negative matrices whose columns are the vectors $\vec{r}_u$ and $\vec{s}_v$. In addition to the EM algorithm above, a variety of other methods exist for finding such decompositions, based on various notions of approximation error, with the most common using a simple mean-squared error [70, 71]. Such approaches are known generically as non-negative matrix factorization algorithms. We use the implementation in the `scikit-learn` Python package, which employs a mean-squared error. We find best results for vector dimension around $k = 80$.

Node2vec: Node2vec is a graph embedding method developed by Grover and Leskovec [48] that uses biased random walks to train a machine learning model. The random walks generate sequences which are used as neighbor sets for nodes, from which the model then learns an embedding. The embedding dimension is a free parameter and we find best results for dimension $k = 256$. We use the implementation of node2vec in the `pytorch-geometric` Python package [39].

Generic bipartite network embedding (GEBE^p): Another embedding-based method is the GEBE^p method of Yang *et al.* [124], which finds embeddings by splitting the problem into two parts: first it computes a similarity between nodes of the same type, analogous to the measures of Section 2.3.1, then it computes weighted paths between nodes of different types. These are then combined into a single objective function and optimized to learn the embedding. Link prediction is performed by training a logistic regression classifier on the combined embedding vectors of node pairs and using the output of the classifier as the prediction score.

Bayesian personalized ranking (BPR): Bayesian personalized ranking [106] is a matrix factorization technique that works by maximizing the probability for each drug individually that a known interaction between the drug and a disease is ranked higher than an unknown one. We use the implementation in the `LightFM` Python package [69].

Intra-class connection triadic closure (ICTC): The ICTC method of Shin *et al.* [115] leverages artificial neural networks to perform link prediction. The method uses a linear graph autoencoder to learn implicit similarities between nodes of the same type, then uses these to predict the presence of edges in a manner similar to the methods of Section 2.3.1.

2.3.3 Probabilistic Network Models

An alternative approach to link prediction is a statistical one in which one fits a probabilistic model of network structure to the observed network. Various models have been used for this purpose. An early example is the work of Clauset *et al.* [29], who proposed a

hierarchical model that captures structure at multiple scales. More recent work has found success using various versions of the stochastic block model [55] discussed in Section 1.3.4. Guimerà and Sales-Pardo [49] were among the first to adopt this approach, employing the SBM in its original unmodified form, but better results are typically found using the variant "degree-corrected" SBM [63, 97].

We consider two variants of this approach. The first is a Bayesian version of the standard degree-corrected model in which group memberships are sampled using a single-node Markov chain Monte Carlo algorithm [107] and prediction scores are averaged over the resulting samples (see Section A.3). The second is a "microcanonical" version of the model in which the numbers of edges, rather than their probabilities, depend on group membership, and states are sampled using a non-local cluster Monte Carlo [97].

2.4 Results

We evaluate the performance of each of our link prediction algorithms on the drug-disease network using cross-validation. We randomly remove 10% of the edges in the network then measure the ability of an algorithm, when applied to the remaining network, to tell which ones were removed. The procedure is repeated 50 times for each algorithm and the results averaged. Success is measured by four standard metrics, some of which we presented in Section 1.4.4 with the others derived from those previously discussed. We provide a brief review of them below:

1. *Area under the Receiver Operating Characteristic curve (AUROC):* Perhaps the most common measure of binary classifier performance, this measure is equal to the area under the ROC curve, i.e., the curve of true positive rate as a function of false positive rate, from start to finish of a run of the algorithm, all the way from the most promising prediction to the least. Values run from 0.5 to 1, with higher values indicating better performance.

2. *Area under the precision/recall curve (AUPR):* In practical situations, we often do care about dilution of the predictions with false positives, in which case the precision—the fraction of predictions that are correct—can be a more useful measure of performance. AUPR provides a threshold-free measure of the predictive quality by average precision over the range of recall. We quote AUPR results as a percentage and they can be read, broadly speaking, as the average probability that a prediction the algorithm makes is correct.

3. *Normalized AUPR:* For sparse data sets we expect the raw precision numbers to be small—we are searching for a small number of needles in a very large haystack, so even an algorithm that does far better than chance will still have low precision. For this reason precision is often normalized relative to the baseline prevalence, i.e., the fraction of true positives in the entire test set. The normalized value measures how much more likely the algorithm is to return a true positive on the average guess than would a random, no-skill classifier.

4. *Top-k precision:* For large prediction problems such as ours it is often the top predictions that are of most interest. A drug developer cannot reasonably be expected to look through all four million drug-disease combinations in our data set, but a carefully curated selection of the most promising candidates could be very useful. The top-k precision is a measure of an algorithm's ability to generate a high-quality selection. It is equal to the fraction of correct predictions among the top k. Here we quote figures for the top 100.

The extent to which one cares about ROC curves versus precision/recall curves or top-k precision depends on one's goals. A clinical researcher (or a patient) with an interest in a particular condition or set of conditions might focus on high AUROC values, since they would want a thorough algorithm that is successful at finding all or most of the promising repurposing candidates, including those they care about, and misses very few. A drug developer, on the other hand, might focus on algorithms with high precision, since these would return the highest-quality predictions, and hence offer the best chance of finding drugs that can be usefully repurposed. In general, we find that higher AUROC figures correspond to lower precision, and vice versa, so one can score well on one or the other but not both.

Figure 2.3 shows average ROC and precision/recall curves for a selection of our algorithms and Table 2.2 summarizes the values of the performance metrics listed above. Inspecting the latter we note first that all algorithms give AUROC scores significantly above the baseline value of 0.5, indicating acceptable performance on the basic link prediction task. There is nonetheless some significant variation. The simple degree-based algorithm is the least competitive and not recommended for this application. The similarity-based algorithms on the other hand do surprisingly well: they all have similar performance with AUROC scores around 0.86, except for the Katz similarity, which fares less well. The machine learning algorithms PLSA, SVD, NNMF, node2vec, BPR, and GEBE^p are in the same vicinity but a little worse, but it is the final three algorithms that stand out for their impressive performance according to this metric—the ICTC deep learning algorithm and

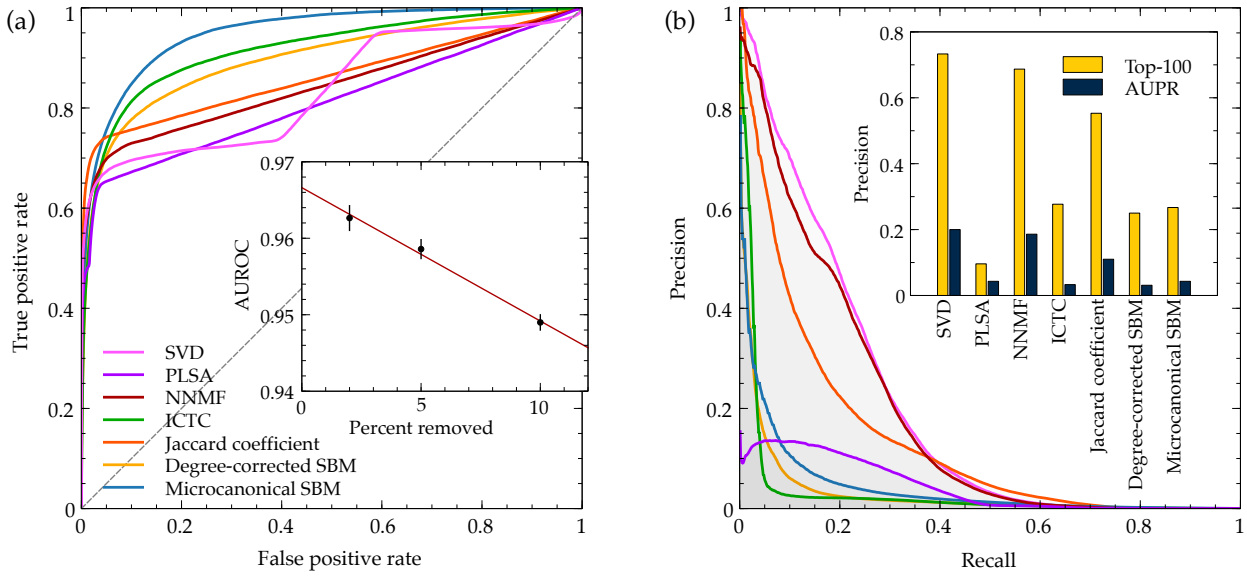


Figure 2.3: (a) Receiver operating characteristic (ROC) curves for seven of the best performing algorithms. The dashed diagonal line represents the expected performance of a no-skill (random) classifier. Inset: Area under the curve (AUROC) for the microcanonical SBM for various fractions of edges removed. (b) Precision/recall curves for the same selection of algorithms. Colors are the same as in (a). Inset: Top-100 precision and area under the prediction/recall curve (AUPR) for each algorithm.

the two versions of the SBM, with the microcanonical SBM giving the best performance of all, with an AUROC score close to 0.95.

Moreover, even this figure is an underestimate. Under normal conditions the performance of any algorithm can be expected to improve as we increase the number of edges in the training data set, which we can do by removing fewer edges for cross-validation. The inset to Figure 2.3 shows AUROC values for the microcanonical SBM for 10%, 5%, and 2% of edges removed. One cannot continue all the way zero—there have to be some edges to predict—but by extrapolating we estimate the AUROC at zero to be 0.967 ± 0.001. Presumably in real-world applications of the method to drug repurposing one would use the entire data set, in order to get the best results possible, so this figure may be the most realistic one for practical applications. Broadly speaking, it means that the microcanonical SBM algorithm is successful at identifying more than 96% of true drug-disease interactions. AUPR figures also vary substantially across algorithms, but they favor different algorithms from the AUROC scores. All of the similarity-based methods return respectable AUPR values in the vicinity of 8% or more. The best is the algorithm based on the Jaccard coefficient, with an AUPR score of 13.8% and a normalized value of 678,

	Algorithm	Area under curve: ROC	Area under curve: Precision/recall	Area under curve: PR normalized	Top-100 precision	Run time (sec)
Elementary	Degree	0.721(2)	0.35(2)%	17(1)	4.7(3)%	1
	Common neighbors	0.860(1)	7.9(1)%	384(5)	44.8(8)%	1
	Cosine similarity	0.860(1)	9.7(2)%	474(9)	45.6(9)%	1
	Jaccard coefficient	0.863(1)	13.8(2)%	678(9)	56.9(9)%	1
	Dice coefficient	0.862(1)	11.5(2)%	565(8)	49.4(9)%	1
	Hub-suppressed index	0.861(1)	12.9(2)%	630(9)	52.8(10)%	1
	Katz similarity [73]	0.789(2)	7.7(1)%	376(5)	44.4(6)%	4
Machine learning	PLSA [53, 54]	0.812(2)	4.3(1)%	209(4)	9.6(5)%	10
	SVD	0.835(1)	**20.0(2)%**	**970(10)**	**73.3(7)%**	40
	NNMF [70, 71]	0.844(1)	18.6(2)%	908(11)	68.7(7)%	57
	Node2vec [48]	0.821(2)	13.4(2)%	655(10)	51.0(9)%	3984
	GEBE^p [124]	0.750(1)	0.31(1)%	15(1)	5.9(1)%	20
	BPR [106]	0.860(1)	9.98(16)%	487(8)	46.3(8)%	17
	ICTC [115]	0.916(1)	3.3(1)%	161(3)	27.7(6)%	242
SBM	DCSBM [49, 63, 107]	0.898(1)	3.1(1)%	154(6)	25.0(8)%	367
	Microcanonical SBM [97]	**0.949(1)**	4.3(1)%	209(5)	26.7(8)%	4964

Table 2.2: Performance measures for link prediction algorithms, estimated from 50 repetitions of cross-validation with 10% of edges removed from the network. We divide our algorithms into three categories: "Elementary" denotes algorithms based on node degrees or simple similarity measures such as counts of common neighbors between node pairs; "Machine learning" denotes methods such as matrix factorization, Bayesian, and deep-learning methods; "SBM" denotes network-based methods that make use of the stochastic block model. The performance measures we use are the area under the ROC curve (AUROC), the area under the precision/recall curve (AUPR), area under the precision/recall curve normalized by prevalence, and precision over the top 100 predictions. Numbers in parentheses indicate standard errors on the trailing digits. Numbers in bold indicate the best performers. Running time is for a single run of each algorithm.

meaning that the algorithm is on average more than 600 times better at identifying drug-disease interactions than a no-skill random classifier. The two versions of the SBM fare less well in this test and in particular the microcanonical model, which is so impressive in terms of the AUROC measure, scores only 4.3%. Among the machine learning algorithms some, such as the sophisticated machine learning algorithm GEBE^p get very poor AUPR scores of under 1%.

The relatively simple singular value decomposition method is the standout on this test, with an AUPR score of 20.0% and performance almost a thousand times better than chance. Moreover, the leading predictions of this algorithm are substantially better even than this,

with a top-100 precision value of 73.3%, meaning that almost three-quarters of the first 100 predictions are correct. The NNMF algorithm also does well, with a top-100 precision of 68.7%. From the point of view of pharmaceutical development these two algorithms, SVD and NNMF, may well be the most promising. Some of the others also perform well on the precision measures—the Jaccard, Dice, and hub-suppressed similarities all score around 50% for instance—but are not competitive with SVD and NNMF. Overall, these results suggest that network-based link prediction algorithms can be a useful tool for identifying candidates for drug repurposing, substantially reducing the amount of work necessary to make each successful identification.

In passing, we also note an interesting coincidence in the dimension of the representations of the network found by several of our algorithms. For SVD we find that the ideal number of singular values to retain is about 60. For NNMF the equivalent number is 80, for PLSA it is 90, and the degree-corrected stochastic block model finds about 30 communities each of drugs and diseases, for a total of around 60 overall. (Exact numbers vary slightly during a run and from one run to another.) The fact that these disparate algorithms are all successful in their predictions and give similar dimensions for the data may be a hint that there are about 60 to 90 different classes of drugs/diseases in the data, an observation that could be of pharmacological interest. (An exception is the node2vec algorithm, for which the best results were obtained with a significantly higher embedding dimension of 256.)

2.4.1 Running Time

The algorithms we consider vary substantially in the amount of time they take to run. Approximate running times for a single run of each algorithm on all 4.3 million drug-disease combinations, measured in seconds of CPU time on conventional hardware circa 2024, are listed in Table 2.2. The elementary similarity-based methods, such as common neighbor counts and cosine similarity, are fastest, taking around one second of CPU time each. (The Katz similarity is slower, at 4 seconds, but still fast.) Although they are not the absolute winners in terms of prediction success, these algorithms do score well, particularly on precision, so they might be of use in cases where speed is important.

The machine learning algorithms are generally more computationally demanding, and in particular the singular value decomposition and non-negative matrix factorization methods are relatively slow, with SVD taking about 40 seconds of CPU time for a single run. Even this, however, is not a significant amount of time in a typical application where one only needs to run the algorithm once. Moreover, SVD and NNMF are both highly

parallelizable and on modern multicore processors the wallclock running time of standard multithreaded implementations is only a fraction of the CPU time—15 seconds or so is typical.

The three algorithms that perform best in terms of AUROC scores are also some of the most demanding—ICTC and the two versions of the stochastic block model. The ICTC algorithm, like many neural network methods, can be accelerated by the use of GPUs, but without such aids takes 8 minutes per run. And the microcanonical SBM, which has the best AUROC score overall, is the most demanding algorithm in our tests, with a running time of over an hour per run.

Balancing prediction success with speed, and assuming a preference for high precision rather than high AUROC scores, our overall pick for best algorithm is the singular value decomposition method, which runs in a few seconds and gives outstanding precision. Non-negative matrix factorization is also competitive. For applications requiring high speed, the similarity-based algorithms may be attractive, with the algorithm based on the Jaccard coefficient being the top performer in terms of prediction success.

2.4.2 Bounds on the Number of Possible Discoveries, False Positives, and Precision

In Appendix A.2 we derive a theoretical bound on the AUROC statistic. We show that the value A of the statistic satisfies $A \leq 1 - \frac{1}{2}\mu$, where μ is the fraction of node pairs in the network that are not observed to be connected by an edge but which are in truth—these are edges that are missing from the data set and they represent the potential successful predictions that we could make using our algorithms. Inverting the inequality, we see that

$$\mu \leq 2(1 - A), \tag{2.3}$$

which places a limit on the number of missing edges waiting to be discovered in the network. This inequality applies to the AUROC score for any algorithm and any fraction of edges removed for cross-validation, so we are at liberty to choose the algorithm that returns the highest value in order to achieve the best bound. In our case, the highest value is the extrapolated value of $A = 0.967$ obtained for the microcanonical SBM algorithm, and, substituting into Eq. (2.3), this implies that the fraction of possible drug-disease interactions remaining to be discovered in our network is at most $2 \times (1 - 0.967) = 0.066$, or 6.6% of the total. At first glance this seems like a small fraction, but it still translates to a substantial number of potential predictions because of the sheer size of the network. The number of node pairs unconnected by an observed edge is $2620 \times 1669 - 8946 = 4.36$

million and the maximum number of potential drug applications waiting to be discovered is 6.6% of this figure, or about 288 000, so there is plenty of room for exploration.

As further shown in Appendix A.2, we also have an upper bound $\nu \leq 2(1 - A)$ on the fraction ν of false *positives* in the data set, which is thus also limited to 6.6%. The sparsity of the data set, however, imposes a much sharper limit, since the number of observed interactions is relatively small, at just 8946. Taking 6.6% of this figure implies we have a maximum of just 590 possible false positives in our data set. This tells us something about the quality of the data: at most 590 of the recorded therapeutic drug-disease combinations are in error.

Finally, as shown in Appendix A.2, the precision is also affected by the presence of missing edges in the network—it is reduced by a factor of $1 - \mu$, as are measures proportional to precision such as AUPR and top-k precision. Given that $0 \leq \mu \leq 0.066$ in our case, we have $0.934 \leq 1 - \mu \leq 1$, which places relatively tight bounds on $1 - \mu$. In practice this means that the measured values of precision should be reasonably reliable and moreover that, to the extent they are modified in the presence of missing edges, they will be increased, not decreased, because the true precision is equal to the measured value divided by $1 - \mu$. Thus for instance the 20% figure we find for AUPR under the SVD algorithm could be a (slight) underestimate—the true value could lie anywhere between 20.0% and 20.0/0.934 = 21.4%.

2.5 Conclusions

In this chapter we have described the construction of a data set of 2620 drugs and 1669 diseases and conditions for which they are indicated, based on several preëxisting, publicly available databases, analyzed using a combination of machine learning methods and human data curation. The resulting data set describes 8946 known drug-disease interactions.

We have used this data set to test the performance of a basket of algorithms for network link prediction, with the goal of identifying potential candidates for drug repurposing. These methods regard the data set as a bipartite network of drugs and diseases, with edges in the network indicating which drug treats which disease, then attempt to identify potential missing edges, which would represent unknown drug-disease pairs. This is a formidable task—there are more than four million possible pairs to consider—but nonetheless several of the algorithms appear to perform well.

Performance of each algorithm is tested using cross-validation, in which a small fraction (in this case 10%) of edges are removed from the network and then the algorithm attempts

to predict the removed edges. We find good success with three algorithms in particular. An algorithm based on the probabilistic network model known as the microcanonical stochastic block model returns an area under the ROC curve of 0.949, which increases to 0.967 when extrapolated to the full network without edges removed for cross-validation, meaning roughly speaking that the algorithm successfully predicts over 96% of the missing edges in the network. These predictions however may be diluted with false positives strewn among them, an issue that is addressed by another performance measure, the precision. By this measure two other algorithms, based on singular value decomposition and non-negative matrix factorization, perform well. These algorithms give precision scores of 73% and 68% respectively on their top 100 predictions, indicating that over two-thirds of those predictions are correct. From the point of view of a drug developer hoping to find promising candidates, this may well be the most important statistic, and these the most promising algorithms.

One issue we have not tackled in this here, but which is of potential interest, is the identification of false positives in the data set. All of our link prediction algorithms give prediction scores for every pair of nodes in the network, both those currently unconnected, which are of interest for repurposing, and those currently connected. An unusually low score for one of the latter would indicate a drug-disease interaction that is indicated in the data set but which, in the eyes of the algorithm, appears suspicious: if we hadn't already been told of this interaction, we would have been unlikely to predict its existence. Such drug-disease pairs could be false positives—drugs that do not in fact treat the diseases they are claimed to. It is a straightforward calculation to find such pairs, but it is not the focus of the present work.

Chapter 3

Hierarchical Core-periphery Structure

In Chapter 2, we explored how network structure can be used to make predictions. We employed various link prediction methods that leverage different characteristics of network structure, ranging from small-scale to large-scale features. Now, we shift our focus to modeling network structure itself, particularly, *core-periphery* structure. This large-scale structure characterizes a commonly observed phenomenon related to the roles nodes play in the network. Core nodes tend to be densely connected to each other, suggesting they play a more significant or central role in the function of the network. In contrast, periphery nodes have fewer connections to other nodes, indicating a less important role in the network.

Most theoretical approaches to studying this structure assume that there is one core group and one periphery group. In this chapter, we relax this constraint and present a model that allows for any number of cores and peripheries. Moreover, we introduce a flexible model-fitting approach to detect these groups and determine their number directly from the data.

3.1 Introduction

In recent years, a significant amount of effort has been devoted to analyzing and understanding large-scale structure in networks, especially community structure [46, 40], but also nested [46, 29] and overlapping [95, 3] communities and stratification [52, 89], as well as related issues of embedding and graph representation learning [50]. In this chapter, we focus on a less well-studied form of large-scale structure, *core-periphery structure*, in which a network is divided into a densely connected core of nodes surrounded by a sparser periphery [19, 56, 109]. The observation of core-periphery structure communicates different information about a network from community structure. Where community structure deals with the identification of groups or types of nodes, core-periphery structure focuses

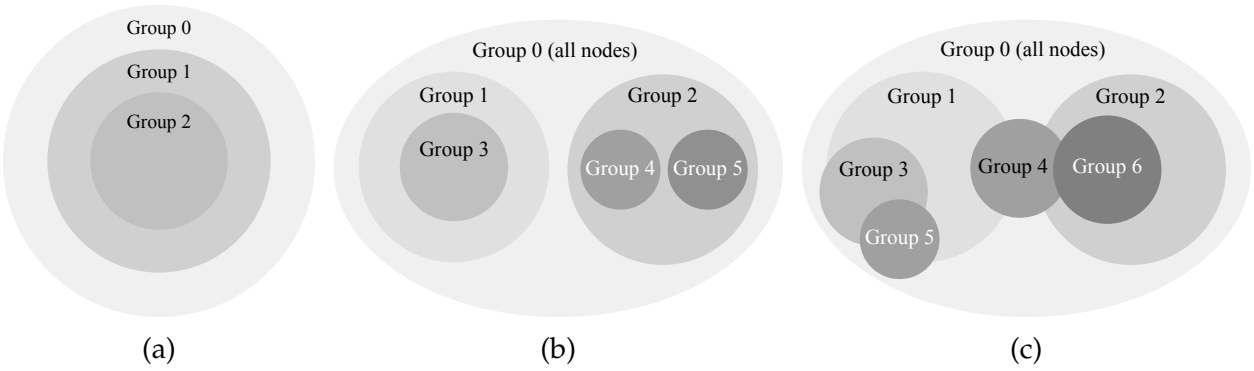


Figure 3.1: The model proposed here is capable of capturing a range of different types of structure, including (a) traditional nested groups, (b) tree-like branching structures, and (c) complex structures of overlapping groups.

on their roles and centrality. Core-periphery structure is integral to understanding the link between node position and function in networks, for instance in the Internet [23, 5], neuroscience [10], and economics [38].

A range of heuristic methods have been proposed in the past for detecting core-periphery structure in networks. In the simplest case the challenge is to take unlabeled network data and assign each node of the network in some automated fashion to either the core or the periphery. In more complex cases, one may attempt to infer an onion-like sequence of deeper and deeper cores within cores.

The problem, however, is not entirely well posed, since we have not precisely defined what "core" and "periphery" mean. Various researchers have chosen to define them in various ways, and there is, as a result, a corresponding spectrum of algorithmic approaches. Perhaps the oldest approach is the *k-core decomposition*, in which nodes are recursively removed from a network in order of increasing degree and the sequence in which they are removed defines a sliding scale from core to periphery [4]. This method essentially equates the core with high-degree nodes. Another well-established method is that of Borgatti and Everett [19], who defined a quality function akin to the well-known modularity function for community detection [88]. Borgatti and Everett's function takes as input a network and a putative division into core and periphery and returns a score that indicates whether the division is a "good" one in a certain sense. Then by maximizing the function over all possible divisions one can find the "best" core-periphery decomposition. Variants of the same idea have been explored by Rombach *et al.* [109], as well as by Kojaku and Masuda [67] who proposed a multigroup version. Cucuringu *et al.* [33] have explored several methods for finding core-periphery structure based on counting geodesic paths and on spectral network properties. Other methods use node-level properties of the

network such as the clustering coefficient [56] or centrality measures [72].

Stochastic block models, commonly used in community detection [55, 63], have also been applied to core-periphery structure. In these models, one assumes that nodes are divided into types (e.g., core and periphery) and that the probability ω_{rs} of an edge between a pair of nodes depends on the types r and s of the nodes. Zhang *et al.* [130] studied a two-group version of this model with $\omega_{11} > \omega_{12} > \omega_{22}$, which generates networks with classic core-periphery structure, then fitted the model to empirical network data to detect structure. Gallagher *et al.* [44] took a similar approach but with more than two groups and multiple nested cores such that $\omega_{rs} = f[\max(r, s)]$ for some function $f[r]$.

In this chapter, we also take a model-fitting approach to the study of core-periphery structure, but we formulate a more general model that includes the classic two-group structure but also allows for more flexible structures as well. In particular, we consider a hierarchical model with any number of groups, the number being determined by the network structure using Bayesian model selection, and we specifically allow for the possibility that the hierarchy may not be perfectly nested and moreover that there maybe multiple cores and multiple peripheries at any level in the hierarchy. These features allow our model to capture variants of core-periphery structure that occur in real-world networks but are not captured by traditional analyses. In a multilevel hierarchy, for example, an inner core may not be perfectly contained within an outer one. In a network with community structure, the individual communities may each have a separate core of their own, or there may be several cores within a single periphery. In this sense, the structures we consider are not directly comparable to previously studied forms of core-periphery structure and we do not expect the output of our algorithms to be directly comparable. But by fitting our proposed model to network data we provide a way to search for unexpected details of network structure, and in the process encounter some new formations that have not been previously observed.

3.2 A Hierarchical Model of Network Structure

One can think of conventional core-periphery structure as a hierarchy. In the simplest case, the hierarchy has just two levels: an outer periphery and an inner core. In more elaborate cases, there can be an onion-like succession of levels, each one nested inside the next, from the outermost to the innermost—see Figure 3.1a. This, however, is not the most general kind of nested structure. A more general structure can have any number of first-level cores within a single periphery, and any number of second-level cores within those first-level ones, and so forth (Figure 3.1b). A structure like this can be captured

with a tree or dendrogram, and some attempts have been made to fit dendrograms to networks [29]. The work presented here goes still further by allowing for the possibility that the hierarchy is not perfectly nested. In our model, we envisage a hierarchical series of cores, but we allow these cores to be placed in any way around the network, with no requirement that they be nested within one another (Figure 3.1c). In practice, if two cores do not overlap then they do not interact and hence their relative order in the hierarchy has no effect, but if they overlap then the higher ranked one takes precedence over the lower in a manner we will describe. The net result is a hierarchical model that generalizes both the conventional onion layers and the dendrogram and, as we will see, captures a wide range of possible structures. We then fit this model to network data to infer core-periphery structure in real-world networks. In detail, the procedure is as follows.

Consider an unweighted, undirected network of n nodes, which can belong to any of k groups labeled by $r = 0 \ldots k-1$. By contrast with traditional community structure models, we allow nodes to belong to any number of groups simultaneously, up to and including all of them. In addition, every node always belongs to group 0, which acts as a sort of default or base group. Thus, each node belongs to group 0 plus some selection of the other groups $1 \ldots k-1$. (We label the first group 0 rather than 1 to remind ourselves of its special status.)

We define a set of indicator variables g_u^r such that $g_u^r = 1$ if node u belongs to group r and 0 otherwise. Then we define a set of probabilities ω_r, one for each group including group 0, and place edges between node pairs independently with probability ω_r, where r is the *highest common group* that both nodes of the pair belong to, meaning the one with the highest number. For example, if one node belongs to groups 0, 1, 2 and another belongs to 0, 1, 3, then there is an edge between them with probability ω_1.

Figure 3.1c illustrates the behavior of the resulting network: the groups form "patches" that lie one on top of another and the topmost visible patch takes precedence for each node pair. For example, if all nodes belong to group 0 only, then every pair of nodes has equal probability ω_0 of being connected, which gives us a standard random graph. But if we assign some subset of the nodes to group 1 then for any pair of nodes that are both in group 1 the probability of an edge becomes ω_1 and overrides the previous ω_0. Similarly, any pair of nodes assigned to group 2, including nodes already in groups 0 and 1, have probability ω_2 of connection, overriding ω_0 and ω_1, and so forth. The end result is a model that starts out as a random graph but then adds variation and detail to the network wherever it is needed to capture local structural features. This approach has some similarities to those of Kojaku and Masuda [67] and Gallagher *et al.* [44], but it differs crucially in that it does not force the groups to be either strictly nested within each other

or nonoverlapping. Given the definition of the model, our goal is now to fit it to observed network data to infer the best choice of groups g_u^r for each node.

Suppose we have a network of n nodes, represented by its $n \times n$ adjacency matrix A with elements $a_{uv} = 1$ if there is an edge between nodes u and v and 0 otherwise. Then the probability of observing a particular network if it was generated from our model with given k and given group memberships g is

$$\begin{aligned} P(A|\omega, k, g) &= \prod_{u<v} \omega_{h(u,v)}^{a_{uv}} \left(1 - \omega_{h(u,v)}\right)^{1-a_{uv}} \\ &= \prod_{r=0}^{k-1} \omega_r^{m_r} \left(1 - \omega_r\right)^{t_r - m_r}, \end{aligned} \tag{3.1}$$

where $h(u, v)$ is the highest common group of nodes u and v, and

$$t_r = \sum_{u<v} \delta_{r,h(u,v)} \tag{3.2}$$

is the number of node pairs with highest common group r, and

$$m_r = \sum_{u<v} a_{uv} \delta_{r,h(u,v)} \tag{3.3}$$

is the number of such pairs that are connected by an edge.

Our primary interest in performing the fit is to determine the group memberships g_u^r. The values of the parameters ω_r are not of particular interest, so we eliminate them by marginalizing. We assume a uniform prior $P(\omega_r) = 1$ for all r and write

$$\begin{aligned} P(A|k, g) &= \int P(A, \omega|k, g)\, d\omega \\ &= \int P(A|\omega, k, g) P(\omega)\, d\omega \\ &= \prod_{r=0}^{k-1} \int_0^1 \omega_r^{m_r} (1 - \omega_r)^{t_r - m_r}\, d\omega_r \\ &= \prod_{r=0}^{k-1} \frac{m_r!(t_r - m_r)!}{(t_r + 1)!}. \end{aligned} \tag{3.4}$$

3.2.1 Prior on Group Assignments

We consider two different scenarios. In the first, the number k of groups in the network is fixed; in the second it is not. The number might be fixed if, for instance, our goal is to find traditional core-periphery structure, for which there are always two groups, the core and the periphery. In other cases, we may be interested in allowing the number of groups to vary and determining what number best fits the network we observe.

If the number of groups is fixed, we can write

$$P(g|A,k) = \frac{P(A|g,k)P(g|k)}{P(A|k)}, \tag{3.5}$$

and we can maximize this quantity to find the most probable values of g, or sample from the distribution it defines to generate plausible core-periphery structures in proportion to their probability. Here we do the latter using a Monte Carlo method.

To do this we need to make a choice for the prior $P(g|k)$. Naively we might assume a prior that is uniform over all assignments g, but this would be a mistake. Such a choice produces group sizes that are narrowly peaked about $\frac{1}{2}n$, which is highly unrealistic. Similar problems occur in traditional community detection [97, 107], where a good solution is instead to choose the *sizes* of the groups to be uniform rather than the assignments and we take an analogous approach here. For each group $r > 0$ we first choose a size n_r uniformly at random between 0 and n, each value thus having probability $1/(n+1)$. Then we choose uniformly at random one of the $\binom{n}{n_r}$ ways to assign n_r nodes to the group, each choice thus having probability $1/\binom{n}{n_r}$. Hence for all groups the total probability of an assignment is

$$P(g|k) = \prod_{r=1}^{k-1} \frac{1}{(n+1)\binom{n}{n_r}} = \prod_{r=1}^{k-1} \frac{n_r!(n-n_r)!}{(n+1)!}. \tag{3.6}$$

3.2.2 Prior on Number of Groups

In traditional core-periphery structure calculations one considers the existence of a single core and a single periphery, and this is the approach taken for instance in [130]. If our goal, however, is to find multiple groups of unknown number, including multiple or overlapping cores, then we need to allow k to vary, which means choosing a prior on k.

Here we adopt the approach taken in [80, 90] and use a Poisson distribution with mean 1:

$$P(k) = \frac{e^{-1}}{(k-1)!}. \tag{3.7}$$

Note that group 0 always exists, so the distribution of the number of groups is effectively a distribution over $k-1$, which is why we have $1/(k-1)!$ in the denominator.

With this choice, we can now write

$$\begin{aligned} P(g,k|A) = P(k)P(g|A,k) &= \frac{P(k)P(g|k)P(A|g,k)}{P(A)} \\ &\propto \frac{1}{(k-1)!}\prod_{r=1}^{k-1}\frac{n_r!(n-n_r)!}{(n+1)!}\prod_{r=0}^{k-1}\frac{m_r!(t_r-m_r)!}{(t_r+1)!}. \end{aligned} \tag{3.8}$$

Again, we can sample from this probability to generate a selection of values k and group assignments g that are representative of the network. In the following section we describe the algorithm we use to achieve this.

3.3 Sampling from the Posterior Distribution

Our goal is to sample from the posterior distribution (3.8), which we do using a Markov-chain Monte Carlo method. This procedure effectively generates a series of plausible core-periphery decompositions of the given network, not a single decomposition, although in practice one can simply examine the highest-probability state found as an indication of the structure present in the network. We describe the method first for the simpler case where the number of groups k is fixed, then for the more complicated case of varying k.

3.3.1 Monte Carlo Algorithm for the Case of Fixed Group Number

For the case of fixed k our Monte Carlo scheme is as follows.

1. We choose a group s uniformly at random from $1 \ldots k-1$.

2. With equal probability $\frac{1}{2}$ we propose to either remove a node from group s or add a node to it. If we are removing, the node to be removed is chosen uniformly at random from those currently in the group. If there are no nodes in the group, we do nothing and move on to the next Monte Carlo step. If we are adding, the node to be added is chosen uniformly at random from those currently not in the group. If

the group is full—all n nodes are already members—we do nothing and move on to the next step.

3. The proposed move is accepted with the Metropolis-Hastings style acceptance probability

$$\alpha(g \to g') = \min\left(1, \frac{P(A|g', k)}{P(A|g, k)}\right), \tag{3.9}$$

where g' represents the group assignments after the addition or removal. If the move is accepted, the chosen node is added or removed as proposed. If the move is not accepted, the group assignments g remain unchanged on this step.

4. Repeat from step 1.

In the limit where this algorithm tends to an equilibrium distribution of states, that distribution will be the one given in Eq. (3.5). To demonstrate this, it suffices to prove two results: first that the algorithm is ergodic and second that it satisfies detailed balance. Ergodicity requires that every state of the system be accessible from every other by a finite sequence of moves. This is trivially true in the present case, since the membership of any group can be set to anything we like in at most n moves by first removing any nodes we don't want and then adding in those we do.

Detailed balance is a little more complicated. Detailed balance requires that in equilibrium the average rate of moves $g \to g'$ equals the average rate $g' \to g$, which means

$$P(g|A, k)P(g \to g') = P(g'|A, k)P(g' \to g), \tag{3.10}$$

where $P(g \to g')$ is the probability of making the transition $g \to g'$. This probability can be written as

$$P(g \to g') = \pi(g \to g')\alpha(g \to g'), \tag{3.11}$$

where $\pi(g \to g')$ is the probability of proposing the move and $\alpha(g \to g')$ is the probability of accepting it as in Eq. (3.9). Then Eq. (3.10) can be written as

$$\frac{P(g'|A, k)}{P(g|A, k)} = \frac{\pi(g \to g')\,\alpha(g \to g')}{\pi(g' \to g)\,\alpha(g' \to g)}. \tag{3.12}$$

We can show that this condition is satisfied by the proposed Monte Carlo algorithm as follows.

From Eq. (3.5) we have

$$\frac{P(g'|A,k)}{P(g|A,k)} = \frac{P(A|g',k)P(g'|k)}{P(A|g,k)P(g|k)}, \tag{3.13}$$

while from Eq. (3.9) the ratio of the two acceptance probabilities is

$$\frac{\alpha(g \to g')}{\alpha(g' \to g)} = \frac{P(A|g',k)}{P(A|g,k)}. \tag{3.14}$$

Substituting (3.13) and (3.14) into (3.12), a factor of $P(A|g',k)/P(A|g,k)$ cancels and we are left with

$$\frac{P(g'|k)}{P(g|k)} = \frac{\pi(g \to g')}{\pi(g' \to g)}. \tag{3.15}$$

If our Monte Carlo algorithm satisfies this condition, then it satisfies detailed balance.

Using Eq. (3.6), the left-hand side can be written as

$$\frac{P(g'|k)}{P(g|k)} = \prod_{r=1}^{k-1} \frac{n_r'!(n-n_r')!}{n_r!(n-n_r)!}, \tag{3.16}$$

where n_r is the number of nodes in group r before the move and n_r' is the number afterwards. Suppose the particular move we are considering $g \to g'$ is one that adds a node to group s. Then $n_s' = n_s + 1$, while $n_r' = n_r$ for all other groups, so (3.16) simplifies to

$$\frac{P(g'|k)}{P(g|k)} = \frac{(n_s+1)!(n-n_s-1)!}{n_s!(n-n_s)!} = \frac{n_s+1}{n-n_s}. \tag{3.17}$$

For the right-hand side of Eq. (3.15), for the same move that adds a node to group s, the proposal probability is

$$\pi(g \to g') = \frac{1}{k-1} \times \frac{1}{2} \times \frac{1}{n-n_s} = \frac{1}{2(k-1)(n-n_s)}. \tag{3.18}$$

Here the factor $1/(k-1)$ is the probability of choosing the particular group r out of all $k-1$ possibilities, the factor $\frac{1}{2}$ is the probability of choosing to add a node, and the factor $1/(n-n_s)$ is the probability of choosing the particular node to be added from the $n-n_s$ possibilities. Meanwhile, for the reverse move $g' \to g$, which involves removing the same

node from group s again, the proposal probability is

$$\pi(g' \to g) = \frac{1}{k-1} \times \frac{1}{2} \times \frac{1}{n_s+1} = \frac{1}{2(k-1)(n_s+1)}, \tag{3.19}$$

since there are now $n_s + 1$ nodes in the group. The ratio of the two proposal probabilities is thus

$$\frac{\pi(g \to g')}{\pi(g' \to g)} = \frac{1/2(k-1)(n-n_s)}{1/2(k-1)(n_s+1)} = \frac{n_s+1}{n-n_s}, \tag{3.20}$$

which agrees with Eq. (3.17) and hence Eq. (3.15) is satisfied and detailed balance is obeyed in this instance. The proof for the case where we remove a node from group s follows the same lines and leads to the same conclusion: the algorithm satisfies detailed balance and hence samples correctly from the target distribution $P(g|A, k)$ in equilibrium.

3.3.2 Algorithm for Varying Group Number

When k is allowed to vary the algorithm is more complex, involving two types of moves that each take us from a combined state (g, k) to a state (g', k'), as follows.

Type 1: In a move of type 1 we choose a group s uniformly at random from $1 \ldots k-1$. With probability $\frac{1}{2}$ we add a new node to the group chosen uniformly from the set of nodes that do not currently belong; otherwise, we remove an existing node from the group, chosen uniformly from those in the group. If we choose to add a node but group s is already full then we do nothing and move on to the next Monte Carlo step. If we choose to remove a node and group s is already empty then the entire group is deleted and the number of groups k decreases by one, with the labels of all groups above s also decreasing by one so that they still run to a maximum of $k-1$.

Type 2: In a move of type 2 we choose a group index s uniformly at random from $1 \ldots k$. We increase by one the labels of all groups s and greater (if there are any), create a new empty group with label s, and increase the value of k by one.

With these definitions, the complete algorithm is now as follows:

1. With probability $1 - 1/2k(n+1)$ propose a move of type 1.

   1a) If $k = 1$ do nothing, since this implies all nodes are in group 0 only, so there are no moves to be made and there is no change of state on this Monte Carlo step.

   1b) Otherwise when $k > 1$ choose a random move of type 1.

2. Else, with probability $1/2k(n+1)$, choose a random move of type 2.

3. Accept the proposed move with probability

$$\alpha(g, k \to g, k) = \min\left(1, \frac{P(A|g', k')}{P(A|g, k)}\right). \tag{3.21}$$

   Accepted moves are performed as proposed. If the move is not accepted the state of the system remains unchanged.

4. Repeat from step 1.

This algorithm again satisfies the condition of ergodicity trivially: we can reach any state with any number of groups in a finite number of moves by first removing all nodes from all groups except group 0, then removing the groups themselves, then adding back the appropriate number of groups and filling them with the desired nodes. The algorithm also satisfies the condition of detailed balance, which for this algorithm takes the form

$$\frac{P(g', k'|A)}{P(g, k|A)} = \frac{\pi(g, k \to g', k')\,\alpha(g, k \to g', k')}{\pi(g', k' \to g, k)\,\alpha(g', k' \to g, k)}. \tag{3.22}$$

The left-hand side can be written as

$$\frac{P(g', k'|A)}{P(g, k|A)} = \frac{P(g', k')P(A|g', k')}{P(g, k)P(A|g, k)}, \tag{3.23}$$

and the ratio of acceptance probabilities is

$$\frac{\alpha(g, k \to g', k')}{\alpha(g', k' \to g, k)} = \frac{P(A|g', k')}{P(A|g, k)}. \tag{3.24}$$

Substituting from (3.23) and (3.24) into (3.22), a factor of $P(A|g', k')/P(A|g, k)$ cancels and our detailed balance condition reduces to

$$\frac{P(g', k')}{P(g, k)} = \frac{\pi(g, k \to g', k')}{\pi(g', k' \to g, k)}. \tag{3.25}$$

From Eqs. (3.6) and (3.7) the left-hand side is

$$\frac{P(g', k')}{P(g, k)} = \frac{(k-1)!\prod_{r=1}^{k-1} n_r'!(n - n_r')!/(n+1)!}{(k'-1)!\prod_{r=1}^{k'-1} n_r!(n - n_r)!/(n+1)!}. \tag{3.26}$$

Consider first the case where we propose a move of type 1 that adds a node to group s.

Then $k' = k$ and $n'_s = n_s + 1$, and $n'_r = n_r$ for all other groups r, so Eq. (3.26) becomes

$$\frac{P(g', k')}{P(g, k)} = \frac{n_s + 1}{n - n_s}. \tag{3.27}$$

The probability of proposing such a move is

$$\begin{aligned}
&\pi(g, k \to g', k') \\
&\quad = \left(1 - \frac{1}{2k(n+1)}\right) \times \frac{1}{k-1} \times \frac{1}{2} \times \frac{1}{n - n_s} \\
&\quad = \left(1 - \frac{1}{2k(n+1)}\right) \frac{1}{2(k-1)(n - n_s)},
\end{aligned} \tag{3.28}$$

while the probability of proposing the reverse move is

$$\pi(g', k' \to g, k) = \left(1 - \frac{1}{2k(n+1)}\right) \frac{1}{2(k-1)(n_s + 1)}. \tag{3.29}$$

Thus the ratio of the two is

$$\frac{\pi(g, k \to g', k')}{\pi(g', k' \to g, k)} = \frac{n_s + 1}{n - n_s}. \tag{3.30}$$

Between Eqs. (3.27) and (3.30), our detailed balance condition (3.25) is now satisfied. By a similar argument we can show that detailed balance is also satisfied when a node is removed from a group.

Now consider a move of type 2, which creates a new empty group with a random label s. For such a move Eq. (3.26) becomes

$$\begin{aligned}
\frac{P(g', k')}{P(g, k)} &= \frac{(k-1)! \prod_{r=1}^{k} n'_r!(n - n'_r)!/(n+1)!}{k! \prod_{r=1}^{k-1} n_r!(n - n_r)!/(n+1)!} \\
&= \frac{1}{k(n+1)},
\end{aligned} \tag{3.31}$$

where all factors inside the products have canceled except for those pertaining to the new group, which gives us the factor of $1/(n+1)$.

The proposal probability for this move is equal to the probability that we decide to do a move of type 2 times the probability that we choose to add a new group with a particular label s out of the k possibilities, giving

$$\pi(g, k \to g', k') = \frac{1}{2k(n+1)} \times \frac{1}{k} = \frac{1}{2k^2(n+1)}. \tag{3.32}$$

The reverse move on the other hand occurs when we perform a move of type 1 and choose group s from the k possibilities, then attempt to remove a node only to discover that the group is already empty, causing us to delete the entire group. The proposal probability for this move is

$$\begin{aligned}\pi(g', k' \to g, k) &= \left(1 - \frac{1}{2(k+1)(n+1)}\right) \times \frac{1}{k} \times \frac{1}{2} \\ &= \frac{1}{2k}\left(1 - \frac{1}{2(k+1)(n+1)}\right).\end{aligned} \tag{3.33}$$

Now the ratio of the two probabilities is

$$\begin{aligned}\frac{\pi(g, k \to g', k')}{\pi(g', k' \to g, k)} &= \frac{4k(k+1)(n+1)/(2(k+1)(n+1)-1)}{2k^2(n+1)} \\ &= \frac{2(k+1)/k}{2(k+1)(n+1)-1} \\ &= \frac{1}{k(n+1)} + \mathrm{O}(1/n^2).\end{aligned} \tag{3.34}$$

Here we assume n is large and hence that terms of order $1/n^2$ can be neglected, making Eq. (3.34) equal to Eq. (3.31), and hence our detailed balance condition, Eq. (3.25), is satisfied.

This completes the proof of correctness of our algorithms. In the following sections we apply these algorithms to fit our model to a variety of networks in order to study core-periphery structure.

3.3.3 Implementation

The algorithms described above are reasonably efficient. The bottlenecks for run time are the evaluation of the quantities t_r, Eq. (3.2), and—for the algorithm with varying numbers of groups—the addition and removal of groups.

When a node u is added to or removed from group r, the highest common group between u and any of the other members of r may change, requiring updates to the variables t_r. In the present implementation we iterate individually over all nodes in r to check for changes. The highest common group between two nodes can be calculated in constant time, so this procedure takes time of order the average size of a group, which is n/k. For the version of the algorithm in which k is fixed, this gives a contribution of order n to the run time per Monte Carlo step.

For the version of the algorithm with varying number of groups, there is extra work

to be done when the number of groups changes. As described in Section 3.3.2, when we delete a group, the labels of all nodes in higher-numbered groups must be decreased by one, which takes worst-case time $O(n)$. On the other hand, moves that decrease the number of groups are relatively rare, occurring approximately k/n of the time, so that on average the relabeling of nodes contributes time $O(n) \times k/n = O(k)$ to the run time.

Overall, therefore, we expect the run time per step to be $O(n + k)$, or $O(n)$ if we assume that n increases faster than k, and numerical tests confirm this behavior. Assuming that the number of Monte Carlo steps for the entire calculation scales as the number of nodes, this implies the total run time will be $O(n^2)$, which is workable though not ideal: many community detection algorithms run in $O(n^2)$ time, but the fastest, such as the Louvain algorithm [16], run in time $O(n \log n)$. It is possible that one might be able to find a shortcut that would allow one to compute updates to the t_r faster than $O(n)$, but we do not at present know of a way to do this.

Our current implementation of the algorithm performs about 500 000 Monte Carlo steps per second for the smaller networks on standard hardware (circa 2023), dropping to 100 000 or less for the largest ones, with total running times ranging from a few minutes to about two hours for the example networks studied here. We do not give any direct comparisons of run time or performance of our algorithm against other algorithms for core-periphery structure, since the particular type of structure we consider is different from other types that go under the "core-periphery" name. As far as we are aware, there are no competing methods for finding hierarchical structure of the kind described in this chapter.

3.4 Example Applications

In this section we give example applications of our methods to a selection of real-world networks, revealing a range of behaviors and structures of interest in the core-periphery divisions of these systems.

3.4.1 Traditional Two-group Core-periphery Structure

For our first set of examples, we perform calculations in which the number of groups is fixed at $k = 2$, which corresponds to the traditional two-group core-periphery structure, as studied by many previous authors. Figure 3.2 shows examples of such structure found in four different networks. For each network the structure shown is the highest-probability structure found during a single run of our algorithm with 10^9 Monte Carlo steps and in

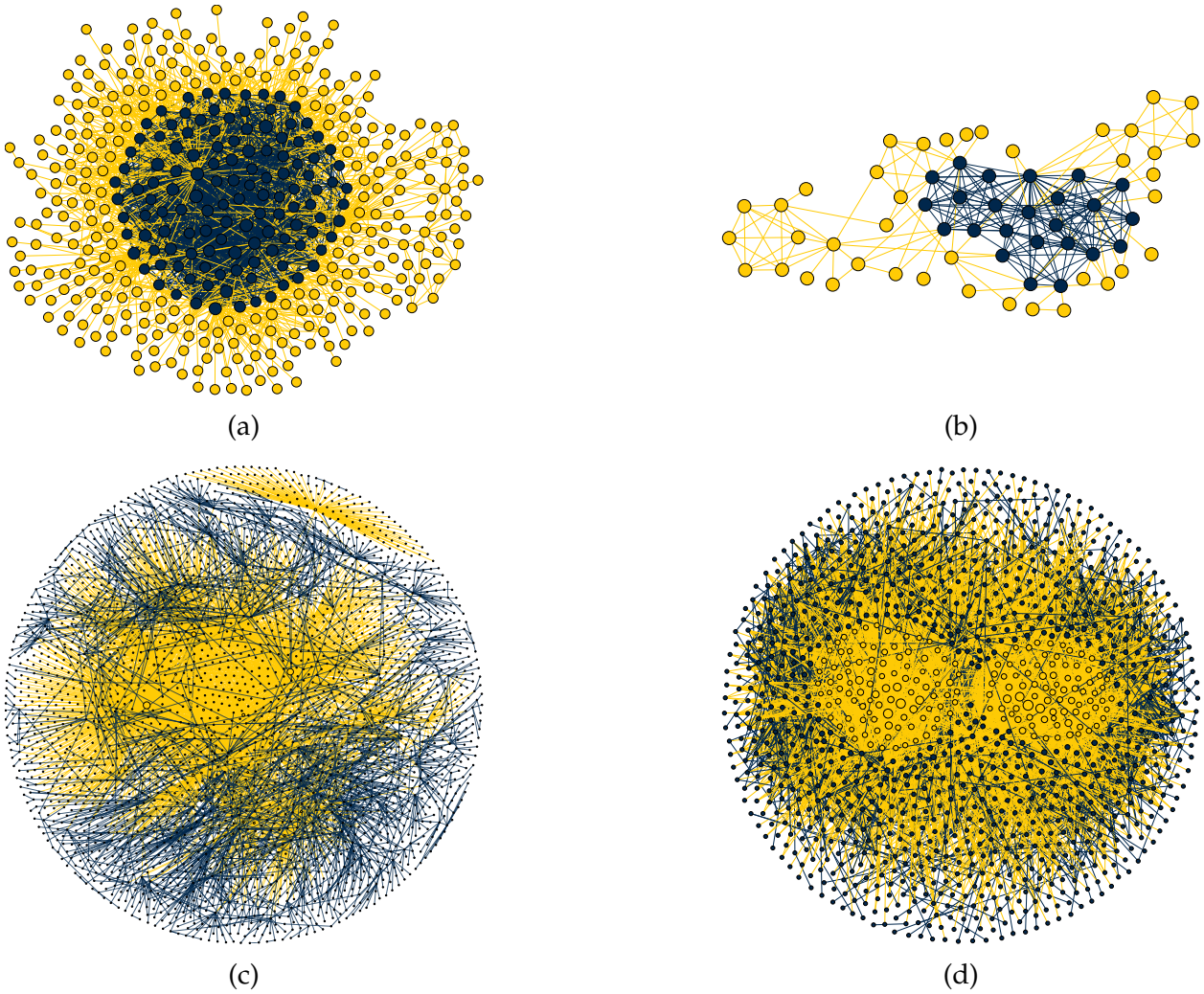


Figure 3.2: Two types of two-group core-periphery structure distinguished by the hierarchical model. (a) is a network of airline routes among European airports [22]. In (b) we show a A network of associations among terrorists involved in the 2004 Madrid train bombing [51]. (c) is a network representation of the Internet in November 1997 at the autonomous system level [74]. While (d) is a network of hyperlinks among a set of US political blogs [2]. Panels (a) and (b) have a core that is densely connected within itself but only sparsely connected to the periphery. This is the traditional definition of core-periphery structure. Panels (c) and (d) on the other hand show a kind of "inside-out" structure in which the core is strongly connected within itself and strongly connected to the periphery.

each case the algorithm finds clear core-periphery divisions, as highlighted by the colors.

A number of interesting features emerge in these examples. First, we note that, as our model is defined, it is arguably the *edges* that belong to groups, not the nodes. As we have said, a node can belong to any number of groups, but an edge only belongs to one: the

properties of each edge are determined solely by the highest-numbered group to which its two nodes both belong and in this sense the edge belongs to this group only. In Figure 3.2 we have colored the edges according to the group they belong to and this provides a clear and useful visualization. (The same trick will also be useful in Section 3.4.2 when we study divisions with larger numbers of groups.)

All of the illustrations in Figure 3.2 use the same color scheme: group 0 is in yellow and group 1 is in blue. The figure reveals that there are two distinctly different types of core-periphery structure, one where the core is group 0 and one where it is group 1. Recall that the probability ω_r of connection between two nodes depends on their highest common group r, meaning in this case that edges between nodes that are both in group 1 have probability ω_1 while all others have probability ω_0. With this in mind take a look at the figure.

Figures 3.2(a) and (b) show results for a network of airline routes [22] and a network of associations among a group of terrorists [51] respectively. In both of these cases the core found in the network is represented by group 1 (in blue) and the periphery by group 0 (in yellow), with $\omega_1 > \omega_0$. This implies that there is a high probability of edges within the core (blue edges) and a lower probability both in the periphery and also between the core and the periphery (yellow edges).

Conversely, in Figs. 3.2(c) and (d), which represent the Internet at the autonomous system level [74] and a network of political weblogs [2], the groups are reversed, with the core being group 0 and the periphery being group 1, and $\omega_1 < \omega_0$. In this "inside-out" type of structure there is a high probability of connections both within the core and between the core and periphery (yellow edges), and a lower probability in the periphery (blue edges).

These two types of core-periphery structure represent quite different circumstances. In the first, the core is isolated from the periphery in the sense that it is densely connected only within itself and sparsely connected to everything else. In the second, the core is strongly connected everywhere, both to itself and to others and dominates the connectivity of the network. The latter ("inside-out") structure is particularly interesting because it deviates from the traditional definition of core-periphery structure as formulated for instance by Borgatti and Everett [19], who assumed an isolated core. Our method naturally and automatically distinguishes between the two types of structure.

The two types make some sense in the present case. For the airline route network, for instance, the core broadly represents airline hubs and the periphery represents regional airports. One expects strong connections between hubs—almost all pairs of hubs have direct flights—but one expects only weak connections to the outlying airports, many of

which only fly to a single hub. Conversely, in the weblog network, for example, the core represents the most influential blogs, ones which most members of the community link to, so we expect connections to be strong not only within the core but also between the core and the periphery.

3.4.2 Structure with an Arbitrary Number of Groups

Now let us look at what happens when we allow the number of groups to vary, taking whatever value is necessary to best fit the structure of the network. Here again we find some interesting features. As a first example, Figure 3.3 shows a copurchasing network of books. The nodes in this network represent 105 popular books on US politics and the edges represent frequent copurchase on Amazon.com, i.e., purchase by the same buyers. This network, which has been studied previously by a number of authors [84, 100], is known to show clear community structure in which the network divides into communities of left- and right-leaning books. Our core-periphery analysis, as indicated by the colors in the figure, finds three groups: two cores and a single periphery. The two cores correspond to the innermost members of the left- and right-leaning communities while the periphery captures the remainder of the network. Thus, the algorithm has found the political divide between left and right but also finds a large group of peripheral books that, at least in this analysis, are well represented as a homogeneous mass, suggesting that they are not strongly connected to either side of the political aisle.

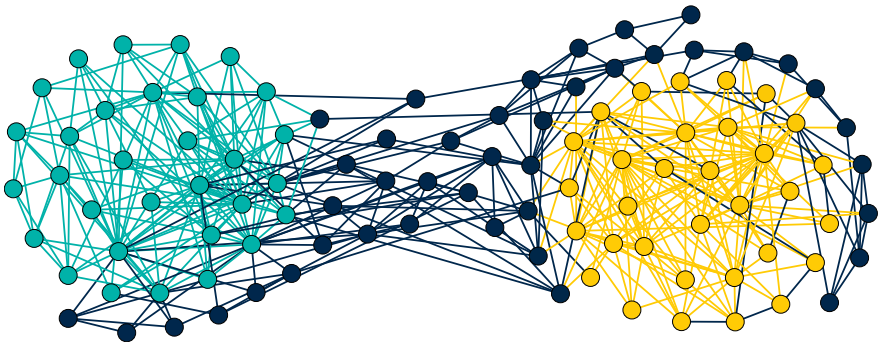


Figure 3.3: Political books network with a periphery and two cores corresponding to left and right leaning books.

Figure 3.4 shows a similar finding for another well studied example of community structure, a network of competition between US college teams in the sport of American football [46]. College football teams are divided into a number of groups or "conferences," and most games are played between teams in the same conference, so the network of games

played, as analyzed here, has strong community structure which can easily be discovered with a range of community detection algorithms. Again, however, our core-periphery analysis returns a more subtle picture, as shown in the figure. Our algorithm finds a separate core for each conference, accurately dividing most teams into the 11 conferences in the network. A small number of teams—many of them independents who belong to no conference—are not assigned to any core, and all inter-conference games are assigned to the periphery. This makes good sense: it tells us that the conferences constitute a clear set of separate groups in the network, while inter-conference play and non-conference teams constitute a single periphery. This is an accurate description of the network and a more economical one than the standard community structure division, as found for instance using the stochastic block model, which also assigns a separate community for each conference but in addition assigns a separate probability for inter-conference play between every single pair of conferences, rather than recognizing that a single periphery is an adequate and more parsimonious description.

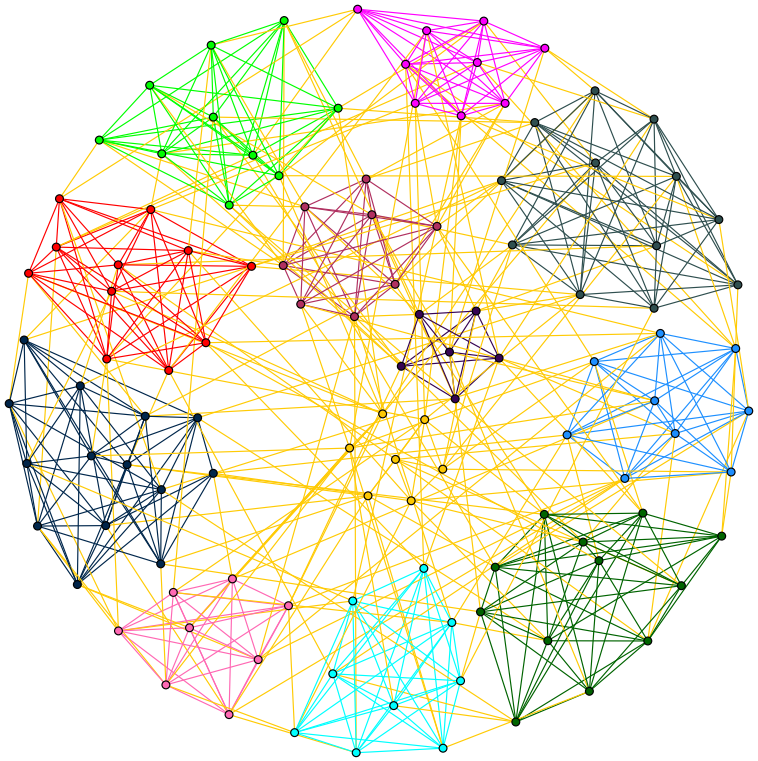
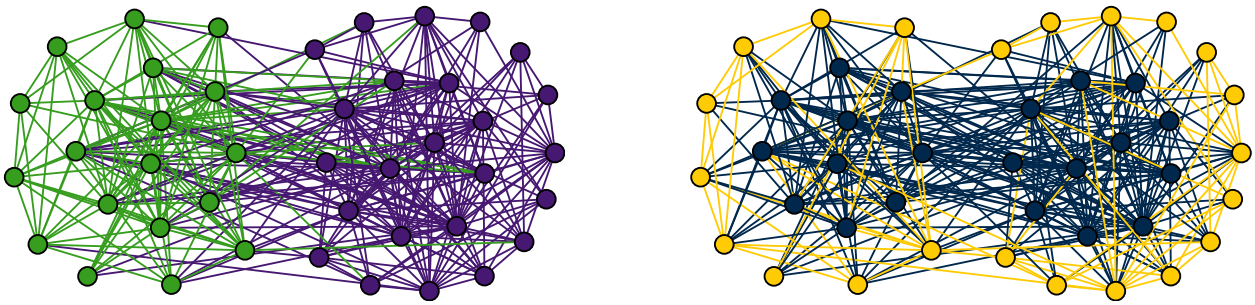


Figure 3.4: American Football network where each of the cliques are connected to each other via the periphery.

In these last two examples our algorithm has found a hybrid of core-periphery structure and community structure. While this is illuminating for these particular examples, it is important to realize that this is not inevitable, and that the algorithm will return other



(a) (b)

Figure 3.5: Structure found in the network of windsurfers. (a) Traditional community structure. (b) Core-periphery structure.

structures where appropriate. Figure 3.5 shows an example. The network in this figure is a famous one from the social networks literature, a network of interactions observed by Freeman [43] between a group of people windsurfing off the California coast in 1986. This network is known to have a clear two-group community structure which is easily found by community detection—see Figure 3.5a. When analyzed using the methods of this chapter, we also find two groups, but they are not the same: now we find core and periphery but no clear division between the communities, suggesting that connections within the core may be just as important as divisions between the two communities.

3.5 Conclusions

In this chapter, we have proposed a hierarchical model of core-periphery structure in networks and a Monte Carlo scheme for fitting it to observed network data. Applying these methods to a variety of real-world networks we find a number of interesting patterns. The method is able to capture traditional two-group core-periphery structure consisting of a dense core weakly connected to a sparse periphery. In some networks, however, we find that a better fit is given by a novel "inside-out" structure in which the core is connected strongly both within itself and to the periphery. Various networks are better represented by one or other of the two types of structure and the distinction between the two could offer a more nuanced view of structure and function in these networks.

We have also investigated cases where there are more than two groups in the network, generalizing the traditional core-periphery structure (as other authors have also done). For this we use a Monte Carlo scheme that allows the number of groups to vary freely, automatically choosing the number that best fits the network in question. In some cases,

we find a structure akin to a hybrid between core-periphery structure and community structure in which there is a separate core in each of several communities plus a single periphery surrounding all of them. In other cases, we find pure core-periphery structure without any communities.

There are a number of possible directions for further research using these methods. First, we have looked here at only the highest probability structures found by our algorithms but in principle the algorithms return a complete sample of high-probability structures drawn from the posterior distribution of the model and it would be interesting to study the range of structures within such a sample. Are they all closely similar, so that a single consensus structure can well represent them all, or is there significant variation between structures, and if so of what kind? Second, one could examine generalizations of the method to broader classes of networks, such as directed and weighted networks and multiplex networks. Another interesting question is whether there exists a natural "degree-corrected" version of the model akin to the degree-corrected stochastic block model of [63]. The model proposed here is not degree corrected, which could cause issues with networks that have a very broad degree distribution.

Chapter 4

Bayesian Inference of the Microcanonical Hypergraph Block Model

Much of the scientific work on large-scale network structure focuses on traditional dyadic networks, where nodes are connected only in pairs. In Chapter 1, we briefly discussed another type of network, hypergraphs, which are networks where edges join any number of nodes, edges referred to as *hyperedges*. In this chapter, we present a Bayesian model for community structure in hypergraphs, a Markov-chain Monte Carlo algorithm to sample from this model, and apply it to a selection of empirical hypergraphs.

4.1 Introduction

Traditionally, network models have focused on representing *pairwise* interactions. However, recent research has shown that some systems, such as social networks [6], ecological networks [7], and neural networks in the brain [128], can be better represented by higher-order interactions (involving two or more nodes). Consequently, there is a growing body of work dedicated to studying the structure and function of higher-order networks and generalizing the successful tools used for analyzing dyadic interactions.

Early work by Milo *et al.* [81] and later by Benson *et al.* [14] studied network motifs—-recurrent patterns of network connections—-suggesting that they may give rise to higher-order interactions in biological and ecological networks. Battiston *et al.* [11] discuss some of the theoretical challenges of studying higher-order interactions and how they differ from conventional dyadic networks. Additionally, random graph models, often the basis for more sophisticated probabilistic models of network structure, have also been extended to hypergraphs. These extensions include generalizing commonly studied models such as the Erodős-Rényi [36], Configuration [17], and Chung and Lu [27] models [34, 25, 9, 102]. Building upon this, statistical inference of hypergraphs has seen increasing use in the study of higher-order interactions.

Young *et al.* [127] developed a Bayesian model to reconstruct hypergraphs directly from pairwise network data. Models for detecting community structure via statistical inference have been studied by Chodrow *et al.* [26], Contisciani *et al.* [32], and more recently by Ruggeri *et al.* [112] and Kritschgau *et al.* [68] each defining generative models for hypergraphs with latent community structure similar to the widely studied Stochastic Block Models [55, 63]. Kamiński *et al.* [62] and Pister *et al.* [102] have also proposed methods to detect community structure in hypergraphs, however, Kamiński *et al.* propose a method based on a generalization of modularity [84] to hypergraphs, while Pister *et al.* propose a limited generalization of the Stochastic Block Model with only two probabilities representing within-group connections and between-group connections. An extensive discussion on extensions of network models to higher-order interaction networks is given by Battiston *et al.* in [12].

While some of the previously mentioned authors all formulate generative models for hypergraphs with community structure, their methods find structure via maximum likelihood estimation. We propose a fully Bayesian model and sample from the posterior, allowing us to have distributions of probable partitions rather than just a single optimal state. We build upon the work of Kritschgau *et al.* who propose a microcanonical likelihood-based model similar to the Microcanonical Stochastic Block Model [97]. However, the method we propose is more flexible, allowing one to infer not only a partition of the nodes amongst groups, but also the total number of partitions using Bayesian model selection.

The remainder of this chapter is organized as follows: in Section 4.2 we reformulate the likelihood of Kritschgau *et al.* [68] and define the prior distributions of the parameters to specify a fully Bayesian model, in Section 4.3 we discuss the Markov-chain Monte Carlo algorithm used to sample from the posterior, in Section 4.4 we apply our model to a variety of networks and discuss its performance, and lastly in Section 4.5 we give our conclusions.

4.2 Microcanonical Hypergraph Block Model

Suppose we have a hypergraph H with n nodes and m edges, represented by its incidence matrix B, where elements $b_{iu} = l$ if edge e_i is incident on node u with multiplicity l and 0 otherwise. Each node u belongs to one of k groups, with $g_u = r$ indicating that node u is in group r. For a hyperedge e_i, let $\rho_r = \sum_u b_{iu}\delta_{g_u,r}$, counting the number of nodes in group r joined by e_i. Each hyperedge e_i is therefore characterized by $\rho = (\rho_1, \dots, \rho_k) \in \mathbb{N}^k$. Moreover, let m_ρ denote the number of hyperedges of type ρ. Lastly, the number of

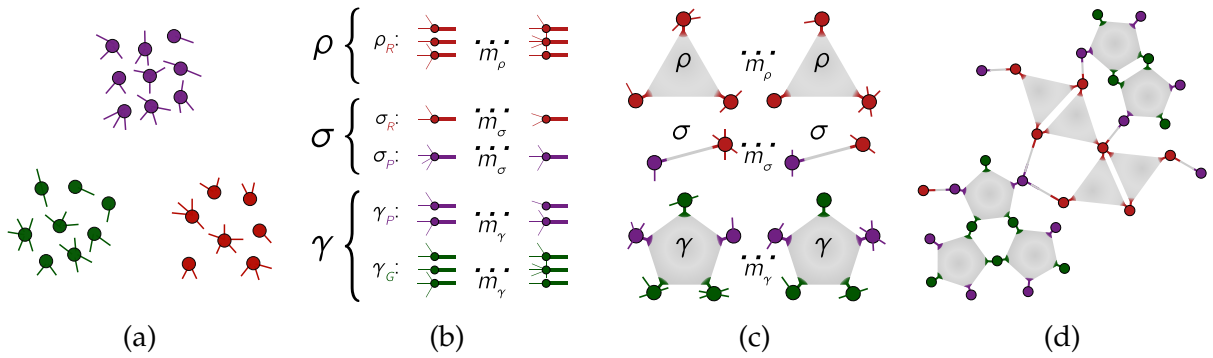


Figure 4.1: Illustration of the generative process for Equation (4.6). (a) First the nodes are labeled by their group. (b) Node stubs are then chosen (bolded stubs) to fill the different slots for the varying hyperedge types. (c) Next, slots come together to form the different hyperedges 4.1c. (d) Lastly, the edges form the hypergraph.

hyperedges incident on a node u, its degree, is given by $d_u = \sum_i b_{iu}$. Consequently, $\Delta_r = \sum_u \sum_i b_{iu} \delta_{g_u,r}$ is the sum of the degrees of all nodes in group r, equivalently expressed as

$$\Delta_r = \sum_{\text{all } \rho} m_\rho \rho_r. \tag{4.1}$$

Having defined the model parameters, we can begin to enumerate the number of hypergraphs compatible with the hard constraints they impose. If we count each stub of every node in group r, we have Δ_r stubs. The quantity $m_\rho \rho_r$ represents the number of slots where stubs in group r can be placed to form hyperedges of type ρ. Accordingly, the number of possible ways of distributing Δ_r stubs amongst the hyperedges is given by

$$Q_r = \frac{\Delta_r!}{\prod_{\text{all } \rho} (m_\rho \rho_r)!}. \tag{4.2}$$

By piecing together the different ρ_r slots in $(\rho_1, \ldots, \rho_k)$ we form a hyperedge of type ρ. The number of ways of forming these m_ρ is then given by

$$\Xi^\rho = \frac{1}{m_\rho!} \prod_r \frac{(\rho_r m_\rho)!}{(\rho_r!)^{m_\rho}}. \tag{4.3}$$

The factor $1/m_\rho!$ accounts for the overcounting of permutations of the m_ρ hyperedges, which we treat as indistinguishable from one another. Finally, we can count the number

of possible hypergraphs admitted by the constraints

$$\Omega\left(g, k, m_\rho\right) = \left(\prod_r Q_r\right)\left(\prod_{\text{all } \rho} \Xi^\rho\right) \tag{4.4}$$

$$= \frac{\prod_r \Delta_r!}{\prod_\rho m_\rho! \prod_\rho \prod_r (\rho_r!)^{m_\rho}}. \tag{4.5}$$

and the probability of choosing one of these hypergraphs is simply the inverse of Equation (4.5)

$$P(H|g, k, m_\rho) = \frac{\prod_\rho m_\rho! \prod_\rho \prod_r (\rho_r!)^{m_\rho}}{\prod_r \Delta_r!}. \tag{4.6}$$

Different stub arrangements correspond to the same hypergraph, in principal we can account for this by dividing Equation (4.5) by a factor of $\prod_u d_u! / \prod_i b_{iu}!$. However, this factor is a constant of the hypergraph and does not play a role in the inference of the community structure, so we drop these constants for the sake of notational clarity.

Equation (4.6) admits hypergraphs with *multi-hyperedges* which may not be representative of many empirical hypergraphs, however, as is done with models for dyadic networks [63, 97, 91], we make the assumption that the number of these multi-hyperedges does not scale with the size of the hypergraph. Similar assumptions are also made by the hypergraph models previously referenced in [26, 32, 112].

4.2.1 Prior Distributions on Model Parameters

With the model likelihood now specified, we continue by discussing parameter estimation, the goal being to find a partition of the nodes into groups that best explains the data under the model. Previous approaches to community detection in dyadic networks directly maximize the model likelihood as in [111, 63, 8] and current models of hypergraph community structure continue in this way including Kritschgau *et al.* [68] who maximize Equation (4.6) via simulated annealing. Here we take a different approach drawing from more recent work on community detection in dyadic networks [108, 107, 97, 126, 103] that make use of a Bayesian framework, offering the ability to directly incorporate uncertainty and regularization.

To this we note that the likelihood in Equation (4.6) can be expressed as

$$P\left(H|g, k, m_\rho\right) = \frac{P\left(g, k, m_\rho|H\right) P\left(m_\rho|k\right) P\left(g|k\right) P\left(k\right)}{P\left(H\right)}. \tag{4.7}$$

Our objective is to estimate the latent parameters g, k, and m_ρ from the data which are fully determined by the hypergraph H. To achieve this, we must sample from the posterior distribution given by

$$P\left(g, k, m_\rho | H\right) \propto P\left(H | g, k, m_\rho\right) P\left(m_\rho | k\right) P\left(g | k\right) P\left(k\right). \tag{4.8}$$

where we have omitted the term $P\left(H\right)$ since it is a constant.

4.2.2 Prior on Hyperedge Counts

We begin by introducing a prior on the hyperedge counts between groups, m_ρ. With the absence of specific information regarding the distribution of the number of hyperedges, we adopt a prior constrained by the total number of hyperedges in the network and their sizes. Consider a hyperedge of size $s \geq 2$. If there are k groups, the number of ways in which to color nodes in the hyperedge is given by

$$\beta_s = \binom{s + k - 1}{s}, \tag{4.9}$$

and the total number of ways of forming μ_s hyperedges of size s is therefore

$$B_s = \binom{\mu_s + \beta_s - 1}{\mu_s}, \tag{4.10}$$

where the total number of hyperedges is given by $m = \sum_s \mu_s$. Taking the product of Equation (4.10) over all hyperedge sizes present in the network gives us the total number of ways creating a set of m-hyperedges, and we chose one uniformly at random with probability

$$P\left(m_\rho | k\right) = \prod_s \binom{\mu_s + \beta_s - 1}{\mu_s}^{-1}. \tag{4.11}$$

The prior in this form is maximally uninformative and is not biased toward assortative or disassortative structures.

4.2.3 Prior on Group Assignments

To find optimal partitions of the network within a Bayesian framework, we must specify a prior on group assignments. Suppose for a moment that the number of groups k is fixed,

then we must make a choice for $P(g|k)$. Naïvely, one might choose a prior of the form $P(g|k) = k^{-n}$, i.e., all assignments g being equally likely. However, as has been discussed before in [90, 107, 97, 126, 103] such a choice produces group sizes that are narrowly peaked, in this case about n/k.

In accordance with previous work, we require instead that group *sizes* be uniformly distributed

$$P\left(g|k\right) = \binom{n-1}{k-1}^{-1} \frac{\prod_r n_r!}{n!}, \tag{4.12}$$

where groups are non-empty and n_r is the number of nodes in group r, such that $n = \sum_r n_r$. Moreover, we aim not only to find suitable partitions of the network, but also to find a representative *number* of partitions k. This requires us to also make a choice for a prior on k, which we do by setting $P(k) = 1/n$. Thus, the joint probability of generating an assignment g with k groups is

$$P\left(g,k\right) = \binom{n-1}{k-1}^{-1} \frac{\prod_r n_r!}{n!} \frac{1}{n}. \tag{4.13}$$

With these choices, we have now specified a fully Bayesian model from which we can sample,

$$P(g,k,m_\rho|H) \propto \frac{\prod_\rho m_\rho! \prod_\rho \prod_r (\rho_r!)^{m_\rho}}{\prod_r \Delta_r!} \prod_s \binom{\mu_s + \beta_s - 1}{\mu_s}^{-1} \binom{n-1}{k-1}^{-1} \frac{\prod_r n_r!}{n!} \frac{1}{n}. \tag{4.14}$$

4.3 Sampling from the Posterior Distribution

We sample states in proportion to their probability given by Equation (4.14) using a Markov-chain Monte Carlo (MCMC) routine. The algorithm explores the space of partitions by proposing changes to group memberships of the nodes one at a time; proposals being accepted with a Metropolis-Hastings acceptance probability. The sampling scheme we employ follows directly from [96, 99], however, generalized to work for hypergraphs.

The sampling scheme includes two types of proposals: moving a node from its current group to another existing group (which we call a type 1 move), and moving a node from an existing group to a newly created one (a type 2 move).

To facilitate the calculation of proposal probabilities, it is useful to transform the hypergraph into a dyadic network G with adjacency matrix A by considering its clique expansion. To construct this network, we take each hyperedge of size l and replace it with

$\binom{l}{2}$ dyadic edges, connecting every pair of nodes in the hyperedge. This allows one to calculate the familiar quantities from the Stochastic Block Model

$$m_{rs} = \sum_{uv} a_{uv} \delta_{g_u,r} \delta_{g_v,s}, \tag{4.15}$$

the number of dyadic edges running between groups r and s and

$$\kappa_r = \sum_{s} m_{rs}, \tag{4.16}$$

the sum of the degrees in G of nodes in group r.

With these quantities defined, we can now outline the specifics of the proposal mechanisms.

Type 1: In a Type 1 move, we propose moving node u from its current group $g_u = r$ to an existing group s. To propose s, we first inspect the group $g_w = t$ of a random neighbor w of node u. With probability $R_t = \epsilon k/(\kappa_t + \epsilon k)$ we choose s uniformly at random from $\{1, ..., k\}$; otherwise, s is chosen with probability in proportion to the number of edges connecting nodes in groups t and s, m_{ts}. The total probability of choosing s is therefore

$$p\left(r \to s|t\right) = \frac{1}{k} R_t + \frac{m_{ts}}{\kappa_t} \left(1 - R_t\right). \tag{4.17}$$

Thus the proposal probability for a move of this type is given by

$$p\left(r \to s\right) = \sum_{t} p_u^t \frac{m_{ts} + \epsilon}{\kappa_t + \epsilon k}, \tag{4.18}$$

where p_u^t is the fraction of neighbors of u that are in group t and ϵ is arbitrary constant greater than 0 that enforces ergodicity and serves to balance fully random moves and moves informed by the currently inferred state of the network, note that as $\epsilon \to \infty$, Equation (4.17) tends to $1/k$.

Type 2: In a Type 2 move, we propose moving node u to a newly created group, choosing $s = k + 1$ and thereby increasing the number of groups by one.

The probability of proposing a move from state (g, k) to a state (g', k') is then given by,

$$\pi\left(g, k \rightarrow g', k'\right) = \begin{cases} q & \text{if s is a new group,} \\ \left(1-q\right) p\left(r \rightarrow s\right) & \text{otherwise,} \end{cases} \tag{4.19}$$

where $0 \leq q \leq 1$ is a free parameter representing the probability with which to propose a new group.

Armed with these definitions, the complete algorithm is as follows:

1. With probability $1 - q$ propose a move of type 1 for node u.

   1a) Choose a random neighbor, w, and inspect its group $g_w = t$.

   1b) With probability $\epsilon k/(\kappa_t + \epsilon k)$ we choose s uniformly at random from $\{1, ..., k\}$

   1c) otherwise, we choose s by selecting a random node v with a neighbor in group t and setting $s = g_v$.

2. Else, with probability q, we make a move of type 2; selecting a newly created group $s = k + 1$.

3. Accept the proposed move with probability

$$\alpha(g, k \rightarrow g', k') = \min\left\{1, \frac{\pi\left(g', k' \rightarrow g, k\right)}{\pi\left(g, k \rightarrow g', k'\right)} e^{\beta \Delta \mathcal{L}}\right\}. \tag{4.20}$$

   Accepted moves are performed as proposed. If the move is not accepted, the state of the system remains unchanged.

4. Repeat from step 1.

Here, β is an inverse-temperature parameter and

$$\Delta \mathcal{L} = \log P(g', k', m'_\rho | H) - \log P(g, k, m_\rho | H)$$

In practice, we find that the above algorithm can have long mixing times and is susceptible to getting trapped in metastable states, also noted by Peixoto in [96]. To address this issue, Peixoto employs an agglomerative heuristic to determine an initial starting point for the Markov chain. Instead, our approach involves running multiple chains in parallel at different inverse temperature values. This is a commonly used technique in Statistical Physics and Molecular dynamics known as replica exchange or parallel tempering [78, 86, 119]. The implementation requires only a minor modification of the algorithm

where, every c sweeps of the hypergraph, after step 3, we propose a swap between chains i and j and accept with probability

$$\alpha_{ij} = \min\left\{1, e^{(\beta_i - \beta_j)(\mathcal{L}_i - \mathcal{L}_j)}\right\} \tag{4.21}$$

where $\mathcal{L}_i$ is the log-posterior of the chain at β_i. Moves of these types are reversible and therefore maintain detailed balance. We note that only the configurations sampled from the $\beta = 1$ chain are used.

4.4 Example Applications

network	n	m	med(l)	$\langle l \rangle$	σ_l	$l_{\max}$	k
political books	105	441	2.0	2.0	0.0	2	3
plant-herbivore	52	21	6.0	8.7	4.4	22	2
high school contact	327	7818	2.0	2.3	0.5	5	9
primary school contact	242	12704	2.0	2.4	0.5	5	10

Table 4.1: Summary of test hypergraphs. n is the number of nodes, m the number of hyperedges, med(l) is the median hyperedge size, $\langle l \rangle$ is the average hyperedge size, σ_l is the standard deviation of hyperedge sizes, $l_{\max}$ is the size of the largest hyperedge, and k is the number of ground-truth communities.

To illustrate the performance of this model, we apply it to a selection of empirical networks with reputed community structure, a summary of these data is given in Table 4.1. In each of these examples, our simulation ran for 100 million Monte Carlo steps. Each replica was initialized with a random number of groups and a random assignment of nodes to each of these groups. The number of replicas, the temperature spacing, and the spacing between exchanges were each tuned to have an exchange acceptance rate between 0.2 and 0.3 to allow for adequate exploration of the probability landscape [45, 110]. The first row of Figure 4.2 illustrates the highest probability states found for each of our examples; the colors representing the different inferred communities to which the nodes belong. In the second row of the figure, we present confusion matrices. The vertical axis represents the ground-truth labels, while the horizontal axis represents the inferred labels. Each cell contains the number of nodes that have the corresponding combination of inferred and ground-truth labels.

For our first example, we look at a traditional dyadic network: a widely studied co-purchasing network of political books depicted in Figure 4.2a. The nodes in this network

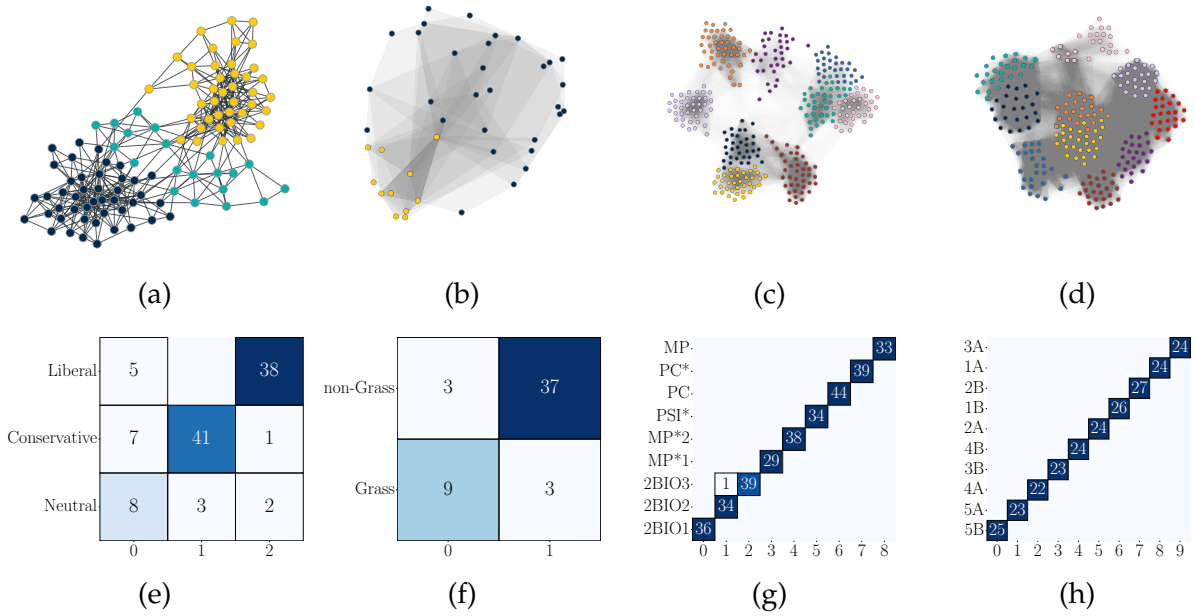


Figure 4.2: For each data set we show the highest probability state found from our Monte Carlo simulation. (a) Political books network. (b) Plant-herbivore hypergraph. Herbivores are the edges and plants the nodes. (c) French high school contact hypergraph. (d) French primary school contact hypergraph. Panels (e)-(h) are contingency tables recording the classification accuracy of our model.

are 105 popular books on US politics and the edges joining them are frequent copurchases (bought by the same buyers) on Amazon.com [84]. The ground-truth labels represent the political leanings of the books: Conservative, Neutral, Liberal. From the confusion matrix in Figure 4.2e, we can see the the model is able to find the community structure in the network and is consistent with previous results [84, 100].

Our second example is a bipartite network where the node types are herbivores (specifically grasshoppers) and plant species; a grasshopper node and a plant node are connected if the grasshopper feeds on the plant. This network was collected and first studied by Joern [60] in order to understand the feeding patterns and diets of grasshoppers in Altuda, Texas. Bipartite graphs can be interpreted as hypergraphs if we choose one of the node sets to be the hyperedges. Here, we have chosen the grasshoppers to be our hyperedges, and they join plant nodes if the particular grasshopper feeds on those set plants. One of the grasshopper species was dropped since they only feed on one plant and we do not permit hyperedges of size 1. This, however, did not reduce the number of plant nodes as the same plant was consumed by other grasshopper species. The metadata lists only the names of the grasshopper species and most plant species, but the community structure

found by our model correlates strongly with grass species and non-grass species, so we have chosen these as the ground-truth labeling of the plant nodes.

From the contingency table in Figure 4.2f we see that the model can very clearly distinguish between grasses and non-grasses based on the feeding habits of the grasshoppers. The three grass species it gets wrong (*Panicum obtusum, Sporobolus sp., Bouteloua curtipendula*) are consumed by grasshoppers that feed mostly on non-grass species while the non-grasses the model misclassifies exist in edges largely dominated by grass species.

Next, we look at an empirically measured hypergraph. This is a hypergraph of contacts in a French high school first studied as dyadic network by Mastrandrea *et al.* [79] and the particular hypergraph version of the data set used in this work was compiled by Chodrow *et al.* [26]. Nodes represent high school students and hyperedges join students that were in proximity to one another (between 1m and 1.5m) for at least 20s. The ground-truth labels represent a categorization particular to French Lyceés called "classes préparatoires" which are roughly subject specializations. From Fig 4.2g, its clear that our model is able to find the community structure of the hypergraph misclassifying only a single student in the highest probability state.

Lastly, we look at another contact hypergraph this, time nodes representing French primary school students and their teachers. This was also first studied as a dyadic network by Stehlé *et al.* [118] and we use the hypergraph version also compiled by Chodrow *et al.* [26] Here, hyperedges join both students and teachers that were in proximity to one another. The ground-truth labels are classrooms (10 in total), however, teachers are labeled separately. Following the work of Chodrow *et al.* [26] and Kritschgau *et al.* [68] we consider teachers to belong to the classroom they teach so that there are 10 communities total rather than the 11 distinguished by the metadata. The model was able to perfectly recover the community structure of the hypergraph assigning every student and teacher to their correct classroom.

In all of these examples, we find that the results are competitive with previously proposed methods for community detection. A key difference between our method and other hypergraph models is that we do not make strong assumptions about the nature of the data. Unlike the approaches in [26, 32, 112, 68], we did not need to specify the number of communities, nor did we have to assume assortativity, as done by Chodrow *et al.* [26] in their AON model and Contisciani *et al.* [32] in their Hypergraph-MT model. The structure was fully resolved by the data itself.

4.5 Conclusions

Here, we have extended the model of Kritschgau *et al.* [68] to form a fully Bayesian model that generates hypergraphs with community structure. Our experiments demonstrate that our method is not only able to detect a hypergraph's community structure, but is also able to infer, fully from the data, the number of communities. Previous work on generative hypergraph models of community structure rely on maximum likelihood estimations and user specification for the number of communities. Our method is more flexible and allows one to not make such strict assumptions about the nature of the data.

Our work leaves several avenues for future research. Our inference algorithm is computationally expensive, requiring many CPU cores for the parallel computations needed to perform the replica exchange Monte Carlo algorithm. Additionally, faster mixing times might be achievable with more effective initializations of the Markov chain. Currently, we use fully random initializations, but incorporating techniques such as the Fibonacci search algorithm to determine the number of groups, as suggested by Yen and Larremore [126], or the greedy optimization proposed by Peixoto [96, 99], could improve the computation time.

Furthermore, we currently examine only the highest probability states found by our algorithm. A key advantage of a fully Bayesian model that allows sampling is the ability to generate a distribution of high-probability structures. This distribution can be studied to find representative divisions, as did Kirkley and Newman [65] for dyadic networks. Leveraging the samples drawn from the Monte Carlo simulation could also facilitate extensions of this model for hypergraph link prediction, an area that remains underexplored.

Chapter 5

Conclusion

This thesis set out to explore large-scale network structures and use them to make meaningful predictions. We began with a discussion of the basic representations, measures, and models of network structure. Chapter 2 continued with an application of these tools to make predictions on a novel data set. We demonstrated how raw data can be transformed into a network of drugs and the diseases they treat. Then, by generalizing the measures and models for unipartite networks to bipartite networks, we leveraged different features of network structure to perform link prediction. Crucially, we achieved impressive performance based solely on the network, without relying on the pharmacological properties of our data. This is an obvious direction for future work: one might reasonably assume that incorporating information related to the scientific mechanisms behind why drugs treat the diseases they do, that link prediction methods could make more refined predictions for repurposing, rather than solely basing it on correlations in the connectivity patterns.

The most successful link prediction methods of Chapter 2 are based on models of large-scale network structure, a highly active area of research in network science. Methods such as modularity maximization or node2vec are largely descriptive, typically explaining what the structure *looks* like rather than how it might have emerged. In contrast, the models presented in Chapters 3 and 4 are fully Bayesian generative models of network structure. These models have the advantage of requiring explicitly stated assumptions of how structure is supposed to arise. This allows us to probe questions about why networks appear as they do, how they change with varying parameters, and provides insight into their function. Moreover, generative models facilitate the incorporation of model selection, addressing fundamental challenges often found in community detection such as determining how many groups there are. Our models utilize Bayesian model selection, enabling the number of latent features to be fully resolved from the data and eliminating the need to make these assumptions or having to do extra model comparisons. Furthermore, the Bayesian framework employs sampling methods that allow us to move beyond single-point parameter estimates. By sampling a distribution of high-probability

parameters, we can capture the structural properties we study and quantify uncertainty in these estimates. If the distribution is strongly peaked around a single parameter value or a small number of similar values, then we have a high degree of certainty about the structure of the network. If the distribution is broad and spread over many different values, then there is a lot of uncertainty.

Chapter 3 explored a relatively understudied type of network structure: core-periphery structure. Traditional methods for studying this structure often assume only two roles for nodes, either as core nodes or as periphery nodes. We presented a more flexible model that allows for the existence of multiple cores and peripheries. This model is able to make subtle distinctions between two types of traditional one-core-one-periphery structure as well, one that has not been previously found. It distinguishes between a core-periphery structure where core nodes connect densely only to other core nodes while being sparsely connected to periphery nodes, and one where core nodes are densely connected to both core and periphery nodes, offering deeper insights into the broader organization of networked systems. Importantly, we do not explicitly assume core-periphery structure in the generative process; rather, the model is flexible enough to find variations in the density of edges present in the network which might correlate strongly with some definitions of this type of structure. This leaves some openings for future work; the model as presently formulated essentially generalizes the Erdős-Rényi random graph model. However, in Chapter 1 we discussed that there is a notion of degree correction we can make that allows us to more faithfully mimic the degree distribution of real-world networks, which the current model cannot do, since "groups" generated from the model have Poisson degree distribution. Extending this model by allowing for the explicit control of node degrees could enable its use in community detection, allowing us to study overlapping community structure, which is currently a difficult phenomenon to model.

In Chapter 4, we continued studying large-scale network structure, this time in higher-order networks. Recent years have seen a growing body of work on higher-order networks, much of it focused on defining and generalizing the measures and metrics discussed in the beginning of Chapter 1 [12]. However, the development of theoretical models of hypergraph structure has been slower. The work presented in Chapter 4 aims to advance our understanding by proposing a generative model for community structure in higher-order networks, along with a Bayesian model inference approach. Although some work on community structure in hypergraphs exists, many of these models require strong assumptions like assortativity. Moreover, these models often make point estimates of the parameters, allowing only the assessment of the highest probability state instead of considering a more comprehensive range of parameter values. Our proposed method

extends previous work to mitigate these issues in dyadic networks to hypergraphs. There are a number of ways that the technique presented in this study could be extended. If we focus on the model likelihood, being able to explicitly control hyperedge sizes could allow for more precise study of how hyperedges impact the emergence of community structure in hypergraphs. Furthermore, we have seen in dyadic networks that there is a fundamental detectability limit for community structure [35]. Currently, no such limit has been formalized for higher-order networks, although some authors have alluded to their existence [26, 68] and such a limit could be further explored using our generative model. One of the main challenges, however, is the computational complexity of the model inference. Improvements to the MCMC algorithm used to sample from the model posterior could lead to faster mixing times and more efficient use of computing resources.

In this thesis, we have demonstrated that what may initially appear as a simplistic representation of scientific systems of interest is, in fact, rich with structure and complexity. The study of networks provides a balance between first-principles theoretical developments and messier data-driven approaches, all with the ultimate goal of better understanding the complex systems that underpin our world. Here, we have presented only modest contributions to the effort, highlighting that significant work remains to be done.

Appendix A

Supplementary Material for Chapter 2

A.1 Selection of Drugs Included in the Data Set

As described in Section 2.2, our data set is assembled from a combination of three pre-existing drug databases, DrugBank, NCATS Inxight, and DailyMed. We start with DrugBank, version 5.1.10, which lists a total of 15 236 distinct medications. Not all of these are of therapeutic interest however. The list also contains things such as cosmetics, foods, allergens, household products, and others, and a crucial step in the preparation of the data is the removal of unwanted entries, which we do in three stages.

First, the DrugBank database assigns each drug to one or more of a set of overlapping status types, including "approved," "investigational," "withdrawn," "illicit," and others. We restrict ourselves to drugs labeled as "approved," a category that denotes drugs approved for clinical use by the US Food and Drug Administration or one of several comparable agencies in other countries or regions. This removes 10 769 drugs from the data set, more than two-thirds of the initial list.

Second, DrugBank assigns to each drug some number of descriptive labels, including both functional and chemical labels, such as "amino acids," "anti-bacterial agents," and "analgesics." We use these labels to identify entries of minimal therapeutic value, eliminating all those in the categories listed in Table A.1. This removes an additional 866 entries. There are also some entries in the database that are not listed as belonging to any category. These "unknown" drugs, of which there are 422, we also remove.

In total we end up removing 12 057 of the starting set of database entries, leaving 3179 core approved drugs in our data set. A final round of hand cleaning of the data reduces this to the 2620 used for the calculations in the paper. During the hand cleaning phase, every interaction in the data set was checked manually to confirm its correctness and a range of other housekeeping operations were performed. Some diseases appeared multiple times in the data set under different names and these were amalgamated into a

Category	Entries
Allergenic extracts	23
Bacterial toxins	12
Bee and wasp venom	12
Coloring agents	41
Contrast media	67
Cosmetics	18
Detergents	14
Diagnostic agents	46
Dietary fats	14
Diet, food, and nutrition	92
Food additives	29
Food allergenic extract	195
Food ingredients	29
Fungal allergenic extract	98
Herbs and natural products	31
Household products	16
Metals	72
Mineral supplements	34
Plant allergenic extract	62
Pollen allergenic extract	218
Solvents	20
Standardized chemical allergens	54
Sunscreen agents	27
Sweetening agents	21
Transition elements	38
Venoms	16

Table A.1: Categories of drugs removed from the data set, along with the number in each category. Note that the total number of drugs removed is less than the sum of the entries in the right-hand column because some drugs belong to more than one category.

single disease. Other diseases appear in both generic and specific forms and the drugs associated with these were rationalized so that every drug that treats a specific form is also recorded as treating the generic form (e.g., all lung cancer drugs are, by definition, also cancer drugs). In addition, some simple text parsing errors were fixed in the hand cleaning stage, as were (rare) cases of AI "hallucination," in which the LLM found an indication for a drug that was not supported by the original data.

A.2 Theoretical Limits on Performance

Missing data and measurement error can limit the performance of link prediction algorithms in cross-validation tests on real-world data. Even an algorithm that can predict missing edges perfectly will *appear* to fail if it correctly predicts an edge that is missing from the original data set—an action that will not be credited as a correct prediction in the cross-validation setting. As we show in this appendix, these effects place limits on the maximum value the AUROC score can attain.

Suppose out of all node pairs that are not connected by an edge in our data set, a fraction μ are in fact connected in reality, but those edges are missing from the data set for some reason—because of experimental error or simply because no measurement of them has ever been made. The remaining fraction $1 - \mu$ are genuinely not connected by an edge, either in the data set or in reality.

Now we run a cross-validation experiment on the network using some link prediction algorithm. A certain fraction of the observed edges are removed and we attempt to predict them. The link prediction algorithm produces a list of node pairs, drawn from those that are unconnected in the training set, in order from most likely predictions to least likely. Suppose we take the first k entries from the top of this list as our edge predictions. For any value of k, let us define p to be the true positive rate for this k, where we include as "positives" both those edges that were removed for cross-validation and those that were erroneously missing from the data from the outset. And let q to be the corresponding false-positive rate for this value of k.

As we let the value of k vary from zero up to the entirety of the list, the values of p and q describe an ROC curve, but unfortunately this is not a curve we can actually measure, because we don't know the identities of all the edges missing from the data set. We know some of them—the ones that we ourselves removed for cross-validation—but not the ones that were missing from the outset. Still, if we did somehow know all the missing edges then we could calculate the area A_0 under the curve from

$$A_0 = \int_0^1 p \, \mathrm{d}q. \tag{A.1}$$

Since we don't know the identity of the edges that are missing from the outset, any link predictions that identify these edges will be wrongly labeled in our cross-validation tests as false positives, when really they are true positives. If we make the assumption that our algorithm is equally good at predicting both the edges removed for cross-validation and edges missing from the outset, then this implies that the estimated true positive rate, as

calculated over the edges removed for cross-validation, will remain unchanged at p, but the false positive rate will be exaggerated and will now take on the larger value

$$q' = \mu p + (1 - \mu)q. \tag{A.2}$$

Here the term μp represents the genuine missing edges that are predicted to be present with probability p but then wrongly labeled as false positives, while the $(1-\mu)q$ represents actual false positives—non-edges that are falsely predicted with probability q.

It is worth pausing for a moment over our assumption that the algorithm is equally good at predicting both types of edges. While this is a reasonable baseline assumption, it could be violated under some circumstances. For instance, it could be that the edges missing from the data set are missing precisely because they are harder to predict: perhaps all the easy-to-find edges have already been added to the data set and those left in the "missing" set are the ones for which link prediction works poorly. Here we assume this not to be the case, so that the average probability of all true positive predictions takes the same value p as it does for the edges removed for cross-validation.

With this assumption, the AUROC value that we calculate in our cross-validation experiment is

$$\begin{aligned} A &= \int_0^1 p \, \mathrm{d}q' = \mu \int_0^1 p \, \mathrm{d}p + (1 - \mu) \int_0^1 p \, \mathrm{d}q \\ &= \tfrac{1}{2}\mu + (1 - \mu)A_0, \end{aligned} \tag{A.3}$$

where we have used Eq. (A.1) in the second line. The largest possible value of A_0, if we had a perfect prediction algorithm, would be 1. Hence an upper bound on the actual measured area under the curve is

$$A \leq 1 - \tfrac{1}{2}\mu. \tag{A.4}$$

Thus, for example, if 10% of true edges are missing from the data set, it will be impossible for any algorithm to achieve an AUROC of greater than $1 - \frac{1}{2} \times 0.1 = 0.95$. Alternatively, we can reverse the argument and say that if we observe an AUROC of A then the fraction μ of edges that could be missing from the data set satisfies

$$\mu \leq 2(1 - A). \tag{A.5}$$

This inequality gives us an upper bound on what we can extract from a given network. It tells us that the fraction μ of unobserved edges remaining to be found in the network is always less than $2(1 - A)$, where A is the value of the AUROC statistic computed from

any link prediction algorithm. Note that the inequality doesn't depend on the fraction of edges removed in cross-validation, so one is at liberty to try any fraction one wants, as well as any algorithm one wants, in order to make the AUROC as large as possible and hence obtain the tightest bound.

The inequality satisfies some basic sanity tests. The smallest value of A is $\frac{1}{2}$, which gives $\mu < 1$, implying that potentially all edges are true predictions in this case. This is trivially correct: if all edges were true predictions then even a perfect algorithm would be unable to distinguish between genuinely missing edges and edges removed in cross-validation, so by definition $A = \frac{1}{2}$. Conversely, the largest possible value of A is 1, implying perfect success at predicting the removed edges. This would give us $\mu = 0$, meaning that none of our predictions correspond to genuinely missing edges, which again is correct: if your algorithm is 100% correct at picking out the edges removed in cross-validation, then there can be no other correct predictions diluting the results.

The presence of missing edges in the data also affects the precision, although in a relatively simple way: the number of true positive predictions is reduced by a factor of $1 - \mu$ and hence the precision is reduced by the same factor, as are measures proportional to precision such as AUPR and top-k precision. From (A.4) and the fact that μ is positive we have $2A - 1 \leq 1 - \mu \leq 1$, which in practice places relatively tight bounds on $1 - \mu$ and hence tells us that precision values are not going to be greatly affected by missing edges.

Returning to AUROC values, suppose now that, in addition to the missing edges in the original data (false negatives) there are also false positives: a fraction ν of the observed edges are wrong. Again we run our cross-validation test, removing some of the edges in the data, including some of these false edges, then attempt to predict the removed edges. Predictions of the false edges will occur with probability q, not probability p, and hence the true-positive rate estimated in the experiment will be modified to a new value

$$p' = (1 - \nu)p + \nu q, \tag{A.6}$$

while the false-positive rate still has the same value as before, Eq. (A.2), so our AUROC score becomes

$$\begin{aligned} A &= \int_0^1 p' \, \mathrm{d}q' \\ &= \mu(1-\nu) \int_0^1 p \, \mathrm{d}p + (1-\mu)(1-\nu) \int_0^1 p \, \mathrm{d}q \\ &\qquad + \mu\nu \int_0^1 q \, \mathrm{d}p + (1-\mu)\nu \int_0^1 q \, \mathrm{d}q. \end{aligned} \tag{A.7}$$

Integrating by parts, we have

$$\int_0^1 q \, \mathrm{d}p = \big[pq\big]_{p=0}^1 - \int_0^1 p \, \mathrm{d}q = 1 - A_0, \tag{A.8}$$

and hence (A.7) becomes

$$\begin{aligned} A &= \tfrac{1}{2}\mu(1-\nu) + (1-\mu)(1-\nu)A_0 \\ &\qquad + \mu\nu(1-A_0) + \tfrac{1}{2}(1-\mu)\nu \\ &= \tfrac{1}{2}(\mu+\nu) + [1-(\mu+\nu)]A_0. \end{aligned} \tag{A.9}$$

We don't know the sign of the coefficient $1-(\mu+\nu)$ in this expression—it could be either positive or negative—but if we assume the fractions of false positives ν and false negatives μ to be small enough that $\mu+\nu<1$, then the coefficient is positive and setting $A_0=1$ again gives us an upper bound $A \le 1-\frac{1}{2}(\mu+\nu)$ and hence

$$\mu + \nu \le 2(1-A), \tag{A.10}$$

which is a generalization of Eq. (A.5). This expression places a bound on the sum of the fractions of false positives and false negatives in the data. Since both fractions are positive, it also implies that $\mu \le 2(1-A)$, so our result from before holds even in the presence of false positives in the data. Moreover we also have $\nu \le 2(1-A)$, so we have an upper bound on the fraction of false positives that could be present.

A.3 Link Prediction Using the Bipartite DCSBM

We use a degree-corrected stochastic block model for link prediction, adopting a bipartite version of the model proposed in [107]. The derivation remains largely unchanged, with a small number of nontrivial modifications to accommodate the bipartite network structure and link prediction. Recall that in the degree-corrected stochastic block model, $n = n_1 + n_2$ nodes are divided into k groups, where $g_u = r$ indicates that node u belongs to group r. This variant of the model requires choosing a set of parameters $\{\theta_u\}$ which control the degree propensity of the nodes and $\{\omega_{rs}\}$ which control the number of edges between groups. The expected number of edges between a given pair of nodes u, v is then given by $\theta_u\theta_v\omega_{g_u g_v}$.

There is, however, an ambiguity in the model: scaling the θ parameters by some factor γ and simultaneously reducing ω by γ^2, leaves $\theta_u\theta_v\omega_{g_u g_v}$ unchanged, meaning there

are multiple equivalent parameterizations of this model. This is typically resolved by constraining either the θ or ω parameters, here we draw directly Riolo *et al.* [107] and constrain the θ parameters by requiring that their sum within each group equals the number of nodes in that group:

$$\sum_{u=1}^{n} \theta_u \delta_{g_u,r} = n_r. \tag{A.11}$$

Consider an observed bipartite network $\Gamma = (U, V, E)$, represented by its $n_1 \times n_2$ incidence matrix B^o, where $n_1 = |U|$ and $n_2 = |V|$. The degree-corrected stochastic block model defines a probability distribution over network adjacency matrices, which can be constructed in terms of the incidence matrix as follows:

$$\tilde{B}^o = \begin{pmatrix} 0_{n_1 \times n_1} & B^o \\ B^{o\mathrm{T}} & 0_{n_2 \times n_2} \end{pmatrix},$$

where $0_{n\times n}$ denotes the square matrix of zeros with dimensions $n \times n$. The probability of generating a network represented by the adjacency matrix $\tilde{B}^o$ is:

$$\begin{aligned} P\left(\tilde{B}^o | g, \theta, \omega, k\right) &= \prod_{u<v} \frac{\left(\theta_u \theta_v \omega_{g_u,g_v}\right)^{\tilde{b}_{uv}}}{\tilde{b}_{uv}!} e^{-\theta_u \theta_v \omega_{g_u,g_v}} \\ &= \prod_u \theta_u^{d_u} \prod_{rs} \omega_{rs}^{m_{rs}} e^{-n_r n_s \omega_{rs}}, \end{aligned} \tag{A.12}$$

where we have used the constraint on the θ parameters, $d_u = \sum_v \tilde{b}_{uv}$ is the observed degree of node u, and $m_{rs} = \sum_{rs} \tilde{b}_{uv} \delta_{g_u,r} \delta_{g_v,s}$ is the number of edges between nodes in groups r and s. Additionally, Poisson stochastic block models usually include a term for possible self-edges (c.f. second term in Equation (1.22)), however, bipartite networks do not contain self-edges by definition. For the same reason, we constrain the expected number of edges between groups of nodes of the same type to be 0. Specifically, $\omega_{rs} = 0$ for any groups where $g_u = r$ and $g_{u'} = s$ with both $u, u' \in U$, thus enforcing the bipartite structure of the networks generated from this model.

The estimation of the θ and ω parameters is not necessary for inferring partitions g of the network, so we marginalize Equation (A.12),

$$P\left(\tilde{B}^o | g, k\right) = \iint P\left(\tilde{B}^o | g, \theta, \omega, k\right) P\left(\theta | g, k\right) P\left(\omega | k\right) \mathrm{d}\theta \mathrm{d}\omega.$$

Where we assume the priors on the parameters are conditionally independent. Moreoever, we use maximum-entropy priors on both θ and ω. The constraint given in Equation (A.11) suggests the values are uniformly distributed over the regular simplex with n_{g_u} vertices, one for each node in the group. Therefore, $P\left(\theta_u|g\right) = \left(n_{g_u} - 1\right)!$ and

$$P\left(\tilde{B}^o|g,\omega,k\right) = \prod_{rs} \omega_{rs}^{m_{rs}} e^{-n_r n_s \omega_{rs}} \times \prod_r (n_r - 1)! \int_\Delta \prod_r \prod_{u|g_u=r} \theta_u^{d_u} \mathrm{d}\theta_u.$$

Where $\int_\Delta$ denotes integration over a regular simplex. With the appropriate substitution: $\theta_u = x_u n_{g_u}$, the integral becomes the definition of the multivariate beta function and the equation simplifies to

$$\begin{aligned} P\left(\tilde{B}^o|g,\omega,k\right) &= \prod_{rs} \omega_{rs}^{m_{rs}} e^{-n_r n_s \omega_{rs}} \times \prod_r (n_r - 1)! \prod_r \prod_{u|g_u=r} n_r^{d_u} \frac{\prod_r \prod_{u|g_u=r} \Gamma\left(d_u + 1\right)}{\prod_r \Gamma\left(\sum_u \left(d_u + 1\right) \delta_{g_u,r}\right)} \\ &= \prod_u d_u! \prod_{rs} \omega_{rs}^{m_{rs}} e^{-n_r n_s \omega_{rs}} \prod_r \frac{(n_r - 1)!}{(\Delta_r + n_r - 1)!} n_r^{\Delta_r}. \end{aligned} \tag{A.13}$$

Where $\Delta_r = \sum_r d_u \delta_{g_u,r}$ is the sum of the degrees in group r and we have made use of $\Gamma\left(n+1\right) = n!$ for nonnegative integers n.

The ω parameters, which control the number of edges between groups, can take any value between 0 and ∞, however, the networks we treat in practice are sparse, meaning that the number of edges in the network is much less than $\mathrm{O}\left(n^2\right)$, the number of places to put an edge. We can enforce this sparsity, by favoring values closer to the average probability of an edge $\mu = m/(n_1 n_2)$. The maximum-entropy prior for a random variable $\xi \in [0,\infty)$ with fixed mean μ is the exponential distribution $P\left(\xi\right) = \left(1/\mu\right) e^{-\xi/\mu}$. Thus, ignoring constants, we have:

$$\begin{aligned} P\left(\tilde{B}^o|g,k\right) &= \prod_u d_u! \prod_r \frac{(n_r - 1)!}{(\Delta_r + n_r - 1)!} n_r^{\Delta_r} \prod_{rs} \int_0^\infty \omega_{rs}^{m_{rs}} e^{-\frac{\omega_{rs}}{\mu}\left(\mu n_r n_s + 1\right)} \mathrm{d}\omega_{rs} \\ &= \prod_u d_u! \prod_r \frac{(n_r - 1)!}{(\Delta_r + n_r - 1)!} n_r^{\Delta_r} \prod_{rs} \frac{m_{rs}!}{\left(\mu n_r n_s + 1\right)^{m_{rs}+1}} \mu^{m_{rs}+1}. \end{aligned} \tag{A.14}$$

Equation (A.14) represents the marginal likelihood of a bipartite degree-corrected stochastic block model. While there remains consideration of the prior distributions on the group assignments g and the number of groups k, these parameters do not affect the bipartite nature of the network and thus do not necessitate special treatment.

Consequently, we adopt the approach of Riolo *et al.* [107] and apply:

$$P(g,k) = (n-2)^{-k} \prod_{r}^{k} n_r!.$$

Therefore, the full posterior $P\left(g,k|\tilde{B}^o\right)$ is given by:

$$P\left(g,k|\tilde{B}^o\right) \propto \prod_u d_u! \prod_r \frac{(n_r-1)!}{(\Delta_r+n_r-1)!} n_r^{\Delta_r} \prod_{rs} \frac{m_{rs}!}{\left(\mu n_r n_s+1\right)^{m_{rs}+1}} \mu^{m_{rs}+1} \frac{\prod_r n_r!}{(n-2)^k}. \tag{A.15}$$

Returning to link prediction, we showed in Section 1.4.2 that if we have observed the network $\tilde{B}^o$, the probability of the network $\tilde{B}$ with a newly added edge (α,β) is:

$$P\left(\tilde{B}|\tilde{B}^o\right) = \sum_{g,k} \frac{P\left(\tilde{B},\tilde{B}^o|g,k\right)}{P\left(\tilde{B}^o|g,k\right)} P\left(g,k|\tilde{B}^o\right). \tag{A.16}$$

By the edge independence assumption of the stochastic block model, the probability $P\left(\tilde{B},\tilde{B}^o|g,k\right)$ is simple to express. If we are only adding the edge (α,β), then the only terms in Equation (A.14) that change are those involving $d_\alpha, d_\beta, \Delta_\rho, \Delta_\varsigma$, and $m_{\rho\varsigma}$ where $g_\alpha = \rho$ and $g_\beta = \varsigma$. The degree of each node in the edge increases by one, and the number of edges between their respective groups also increases by one. All other terms remain unchanged and thus cancel out in the division. Therefore, the ratio in Equation (A.16) can be expressed as:

$$\frac{P\left(\tilde{B},\tilde{B}^o|g,k\right)}{P\left(\tilde{B}^o|g,k\right)} = \frac{(d_\alpha+1)\left(d_\beta+1\right)}{\left(\Delta_\rho+n_\rho\right)\left(\Delta_\varsigma+n_\varsigma\right)} n_\rho n_\varsigma \times \frac{m_{\rho\varsigma}}{\mu n_\rho n_\varsigma+1}\mu. \tag{A.17}$$

Finally, to perform the link prediction, we sample partitions from Equation (A.15) and average the ratio over all the samples to calculate the prediction score.

Bibliography

1. Khushnood Abbas, Alireza Abbasi, Shi Dong, Ling Niu, Laihang Yu, Bolun Chen, Shi-Min Cai, and Qambar Hasan. Application of network link prediction in drug discovery. *BMC Bioinformatics*, 22:187, April 2021.

2. Lada A. Adamic and Natalie Glance. The political blogosphere and the 2004 US election. In *Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem*, New York, 2005. Association of Computing Machinery.

3. Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9:1981–2014, 2008.

4. José Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. K-core decomposition: A tool for the visualization of large scale networks. Preprint arxiv:cs/0504107, 2006.

5. José Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. K-core decomposition of internet graphs: Hierarchies, self-similarity and measurement biases. *Networks and Heterogeneous Media*, 2:371–393, 2008.

6. Unai Alvarez-Rodriguez, Federico Battiston, Guilherme Ferraz de Arruda, Yamir Moreno, Matjaž Perc, and Vito Latora. Evolutionary dynamics of higher-order interactions in social networks. *Nature Human Behaviour*, 5(5):586–595, May 2021.

7. Eyal Bairey, Eric D. Kelsic, and Roy Kishony. High-order species interactions shape ecosystem diversity. *Nature Communications*, 7(1):12285, August 2016.

8. Brian Ball, Brian Karrer, and M. E. J. Newman. An efficient and principled method for detecting communities in networks. *Phys. Rev. E*, 84:036103, 2011.

9. Marc Barthelemy. Class of models for random hypergraphs. *Phys. Rev. E*, 106:064310, Dec 2022.

10. Danielle S. Bassett, Nicholas F. Wymbs, M. Puck Rombach, Mason A. Porter, Peter J. Mucha, and Scott T. Grafton. Task-based core-periphery organization of human brain dynamics. *PLOS Computational Biology*, 9:1–16, 2013.

11. Federico Battiston, Enrico Amico, Alain Barrat, Ginestra Bianconi, Guilherme Ferraz de Arruda, Benedetta Franceschiello, Iacopo Iacopini, Sonia Kéfi, Vito Latora,

Yamir Moreno, Micah M. Murray, Tiago P. Peixoto, Francesco Vaccarino, and Giovanni Petri. The physics of higher-order interactions in complex systems. *Nature Physics*, 17(10):1093–1098, October 2021.

12. Federico Battiston, Giulia Cencetti, Iacopo Iacopini, Vito Latora, Maxime Lucas, Alice Patania, Jean-Gabriel Young, and Giovanni Petri. Networks beyond pairwise interactions: Structure and dynamics. *Physics Reports*, 874:1–92, 2020.

13. Anastasiya Belyaeva, Louis Cammarata, Adityanarayanan Radhakrishnan, Chandler Squires, Karren Dai Yang, GV Shivashankar, and Caroline Uhler. Causal network models of SARS-CoV-2 expression and aging to identify candidates for drug repurposing. *Nature Communications*, 12:1024, 2021.

14. Austin R. Benson, David F. Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.

15. Halil Bisgin, Zhichao Liu, Hong Fang, Reagan Kelly, Xiaowei Xu, and Weida Tong. A phenome-guided drug repositioning through a latent variable model. *BMC Bioinformatics*, 15:1–12, 2014.

16. Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech.*, 2008:P10008, 2008.

17. Béla Bollobás. A probabilistic proof of an asymptotic formula for the number of labelled regular graphs. *European Journal of Combinatorics*, 1:311–316, 1980.

18. Phillip F. Bonacich. A technique for analyzing overlapping memberships. In Herbert Costner, editor, *Sociological Methodology*. Jossey-Bass, San Francisco, 1972.

19. Stephen P. Borgatti and Martin G. Everett. Models of core/periphery structures. *Social Networks*, 21:375–395, 1999.

20. Anna D. Broido and Aaron Clauset. Scale-free networks are rare. *Nature Communications*, 10(1):1017, March 2019.

21. Navneet Bung, Sowmya R Krishnan, Gopalakrishnan Bulusu, and Arijit Roy. De novo design of new chemical entities for SARS-CoV-2 using artificial intelligence. *Future Medicinal Chemistry*, 13:575–585, 2021.

22. Alessio Cardillo, Jesús Gómez-Gardeñes, Massimiliano Zanin, Miguel Romance, David Papo, Francisco del Pozo, and Stefano Boccaletti. Emergence of network features from multiplexity. *Scientific Reports*, 3:1344, 2013.

23. Shai Carmi, Shlomo Havlin, Scott Kirkpatrick, Yuval Shavitt, and Eran Shir. A model of Internet topology using k-shell decomposition. *Proc. Natl. Acad. Sci. USA*, 104:11150–11154, 2007.

24. Feixiong Cheng, Chuang Liu, Jing Jiang, Weiqiang Lu, Weihua Li, Guixia Liu, Weixing Zhou, Jin Huang, and Yun Tang. Prediction of drug-target interactions and drug repositioning via network-based inference. *PLOS Computational Biology*, 8:e1002503, 2012.

25. Philip S Chodrow. Configuration models of random hypergraphs. *Journal of Complex Networks*, 8(3):cnaa018, 08 2020.

26. Philip S. Chodrow, Nate Veldt, and Austin R. Benson. Generative hypergraph clustering: From blockmodels to modularity. *Science Advances*, 7(28):eabh1303, 2021.

27. Fan Chung and L. Lu. The average distances in random graphs with given expected degrees. *Proc. Natl. Acad. Sci. USA*, 99:15879–15882, 2002.

28. Fan Chung and L. Lu. Connected components in random graphs with given degree sequences. *Annals of Combinatorics*, 6:125–145, 2002.

29. Aaron Clauset, Cristopher Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453:98–101, 2008.

30. Aaron Clauset, Cosma R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51:661–703, 2009.

31. Sarel Cohen, Moshik Hershcovitch, Martin Taraz, Otto Kißig, Davis Issac, Andrew Wood, Daniel Waddington, Peter Chin, and Tobias Friedrich. Improved and optimized drug repurposing for the SARS-CoV-2 pandemic. *PLOS One*, 18:1–13, 2023.

32. Martina Contisciani, Federico Battiston, and Caterina De Bacco. Inference of hyperedges and overlapping communities in hypergraphs. *Nature Communications*, 13(1):7229, November 2022.

33. Mihai Cucuringu, Puck Rombach, Sang Hoon Lee, and Mason A. Porter. Detection of core-periphery structure in networks using spectral methods and geodesic paths. *European Journal of Applied Mathematics*, 27:846–887, 2016.

34. W Fernandez De La Vega. Sur la cardinalité maximum des couplages d'hypergraphes aléatoires uniformes. *Discrete Mathematics*, 40(2-3):315–318, 1982.

35. Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Inference and phase transitions in the detection of modules in sparse networks. *Phys. Rev. Lett.*, 107:065701, 2011.

36. Paul Erdős and Alfréd Rényi. On random graphs. *Publicationes Mathematicae*, 6:290–297, 1959.

37. Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 8:128–140, 1741.

38. Giorgio Fagiolo, Javier Reyes, and Stefano Schiavo. The evolution of the world trade web: A weighted-network analysis. *Journal of Evolutionary Economics*, 20:479–514, 2010.

39. Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

40. Santo Fortunato. Community detection in graphs. *Phys. Rep.*, 486:75–174, 2010.

41. Santo Fortunato and Mark E. J. Newman. 20 years of network community detection. *Nature Physics*, 18:848 – 850, 2022.

42. Julie Fournet and Alain Barrat. Contact patterns among high school students. *PLOS ONE*, 9(9):1–17, 09 2014.

43. L. C. Freeman, S. C. Freeman, and A. G. Michaelson. On human social intelligence. *Journal of Social and Biological Structures*, 11:415–425, 1988.

44. Ryan J. Gallagher, Jean-Gabriel Young, and Brooke Foucault Welles. A clarified typology of core-periphery structure in networks. *Science Advances*, 7:eabc9800, 2021.

45. A. Gelman, W. R. Gilks, and G. O. Roberts. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7(1):110 – 120, 1997.

46. Michelle Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99:7821–7826, 2002.

47. Assaf Gottlieb, Gideon Y. Stein, Eytan Ruppin, and Roded Sharan. PREDICT: A method for inferring novel drug indications with application to personalized medicine. *Molecular Systems Biology*, 7:496, 2011.

48. Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 855–864, New York, NY, 2016. Association for Computing Machinery.

49. Roger Guimerà and Marta Sales-Pardo. Missing and spurious interactions and the reconstruction of complex networks. *Proc. Natl. Acad. Sci. USA*, 106:22073–22078, 2009.

50. William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. In *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, New York, 2017. Institute of Electrical and Electronics Engineers.

51. Brian Hayes. Connecting the dots: Can the tools of graph theory and social-network studies unravel the next big plot? *American Scientist*, 94(5):400–404, 2006.

52. Peter D. Hoff, Adrian E. Raftery, and Mark S. Handcock. Latent space approaches to social network analysis. *J. Amer. Stat. Assoc.*, 97:1090–1098, 2002.

53. T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM Conference on Research and Development in Information Retrieval*, New York, NY, 1999. Association of Computing Machinery.

54. T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22:89–115, 2004.

55. P. W. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5:109–137, 1983.

56. Petter Holme. Core-periphery organization of complex networks. *Phys. Rev. E*, 72:046111, 2005.

57. Yu-Fen Huang, Hsiang-Yuan Yeh, and Von-Wun Soo. Inferring drug-disease associations from integration of chemical, genomic and phenotype data using network propagation. *BMC Medical Genomics*, 6(Suppl. 3):S4, 2013.

58. Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et du jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37(142):547–579, 1901.

59. H. H. Jennings. *Leadership and isolation; a study of personality in inter-personal relations.* Leadership and isolation; a study of personality in inter-personal relations. Longmans, 1943.

60. Anthony Joern. Feeding patterns in grasshoppers (Orthoptera: Acrididae): Factors influencing diet specialization. *Oecologia*, 38(3):325–347, January 1979.

61. Jean-Pierre Jourdan, Ronan Bureau, Christophe Rochais, and Patrick Dallemagne. Drug repositioning: A brief overview. *Journal of Pharmacy and Pharmacology*, 72:1145–1151, August 2020.

62. Bogumił Kamiński, Valérie Poulin, Paweł Prałat, Przemysław Szufel, and François Théberge. Clustering via hypergraph modularity. *PloS one*, 14(11):e0224307, 2019.

63. Brian Karrer and M. E. J. Newman. Stochastic blockmodels and community structure in networks. *Phys. Rev. E*, 83:016107, 2011.

64. L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18:39–43, 1953.

65. Alec Kirkley and M. E. J. Newman. Representative community divisions of networks. *Communications Physics*, 5:40, 2022.

66. Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46:604–632, 1999.

67. Sadamori Kojaku and Naoki Masuda. Finding multiple core-periphery pairs in networks. *Phys. Rev. E*, 96:052313, 2017.

68. Jürgen Kritschgau, Daniel Kaiser, Oliver Alvarado Rodriguez, Ilya Amburg, Jessalyn Bolkema, Thomas Grubb, Fangfei Lan, Sepideh Maleki, Phil Chodrow, and Bill Kay. Community detection in hypergraphs via mutual information maximization. *Scientific Reports*, 14(1):6933, March 2024.

69. Maciej Kula. Metadata embeddings for user and item cold-start recommendations. In Toine Bogers and Marijn Koolen, editors, *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems*, pages 14–21, 2015.

70. D. D. Lee and H. S. Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401:788–791, 1999.

71. Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Proceedings of the 2000 Neural Information Processing Systems Conference*, pages 556–562, Cambridge, MA, 2001. MIT Press.

72. Sang Hoon Lee, Mihai Cucuringu, , and Mason A. Porter. Density-based and transport-based core-periphery structures in networks. *Phys. Rev. E*, 89:032810, 2014.

73. E. A. Leicht, Petter Holme, and M. E. J. Newman. Vertex similarity in networks. *Phys. Rev. E*, 73:026120, 2006.

74. Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

75. David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *J. Assoc. Inf. Sci. Technol.*, 58:1019–1031, 2007.

76. Zhongyang Liu, Feifei Guo, Jiangyong Gu, Yong Wang, Yang Li, Dan Wang, Liang Lu, Dong Li, and Fuchu He. Similarity-based prediction for anatomical therapeutic chemical classification of drugs by integrating multiple data sources. *Bioinformatics*, 31:1788–1795, 2015.

77. Linyuan Lu and Tao Zhou. Link prediction in complex networks: A survey. *ArXiv*, abs/1010.0725, 2010.

78. E. Marinari and G. Parisi. Simulated tempering: A new monte carlo scheme. 19(6):451, jul 1992.

79. Rossana Mastrandrea, Julie Fournet, and Alain Barrat. Contact patterns in a high school: A comparison between data collected using wearable sensors, contact diaries and friendship surveys. *PLOS ONE*, 10(9):e0136497, 2015.

80. Aaron F. McDaid, Thomas Brendan Murphy, Nial Friel, and Neil Hurley. Improved Bayesian inference for the stochastic block model with application to large networks. *Computational Statistics and Data Analysis*, 60:12–31, 2013.

81. R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.

82. Jacob L. Moreno. *Who Shall Survive?* Beacon House, Beacon, NY, 1934.

83. M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E*, 69:066133, 2004.

84. M. E. J. Newman. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA*, 103:8577–8582, 2006.

85. M. E. J. Newman. Message passing methods on complex networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 479:20220774, 2023.

86. M. E. J. Newman and G. T. Barkema. *Monte Carlo Methods in Statistical Physics*. Oxford University Press, Oxford, 1999.

87. M. E. J. Newman and Aaron Clauset. Structure and inference in annotated networks. *Nature Communications*, 7:11863, 2016.

88. M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, 2004.

89. M. E. J. Newman and Tiago P. Peixoto. Generalized communities in networks. *Phys. Rev. Lett.*, 115:088701, 2015.

90. M. E. J. Newman and Gesine Reinert. Estimating the number of communities in a network. *Phys. Rev. Lett.*, 117:078301, 2016.

91. Mark Newman. *Networks*. Oxford University Press, Oxford, 2 edition, 2018.

92. K. Nowicki and T. A. B. Snijders. Estimation and prediction for stochastic blockstructures. *J. Amer. Stat. Assoc.*, 96:1077–1087, 2001.

93. T. I. Oprea and J. Mestres. Drug repurposing: Far beyond new targets for old drugs. *The AAPS Journal*, 14:759–763, December 2012.

94. L. Page, S. Brin, R. Motwani, and T. Winograd. The Pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

95. Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435:814–818, 2005.

96. Tiago P. Peixoto. Efficient monte carlo and greedy heuristic for the inference of stochastic block models. *Phys. Rev. E*, 89:012804, 2014.

97. Tiago P. Peixoto. Nonparametric Bayesian inference of the microcanonical stochastic block model. *Phys. Rev. E*, 95:012317, 2017.

98. Tiago P. Peixoto. Reconstructing networks with unknown and heterogeneous errors. *Phys. Rev. X*, 8:041011, 2018.

99. Tiago P. Peixoto. Merge-split markov chain monte carlo for community detection. *Phys. Rev. E*, 102:012305, Jul 2020.

100. Tiago P. Peixoto. Revealing consensus and dissensus between network partitions. *Phys. Rev. X*, 11:021003, Apr 2021.

101. Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 701–710, New York, NY, 2014. Association for Computing Machinery.

102. Alexis Pister and Marc Barthelemy. Stochastic block hypergraph model. *Phys. Rev. E*, 110:034312, Sep 2024.

103. Austin Polanco and M. E. J. Newman. Hierarchical core-periphery structure in networks. *Phys. Rev. E*, 108:024311, Aug 2023.

104. Sudeep Pushpakom, Francesco Iorio, Patrick A. Eyers, K. Jane Escott, Shirley Hopper, Andrew Wells, Andrew Doig, Tim Guilliams, Joanna Latimer, Christine McNamee, Alan Norris, Philippe Sanseau, David Cavalla, and Munir Pirmohamed. Drug repurposing: Progress, challenges and recommendations. *Nature Reviews Drug Discovery*, 18:41–58, January 2019.

105. Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Phys. Rev. E*, 67:026112, 2003.

106. Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In Jeff A. Bilmes and Andrew Y. Ng, editors, *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461, Corvallis, OR, 2009. AUAI Press.

107. Maria A. Riolo, George T. Cantwell, Gesine Reinert, and M. E. J. Newman. Efficient method for estimating the number of communities in a network. *Phys. Rev. E*, 96:032310, 2017.

108. Maria A. Riolo and M. E. J. Newman. First-principles multiway spectral partitioning of graphs. *Journal of Complex Networks*, 2:121–140, 2014.

109. M. Puck Rombach, Mason A. Porter, James H. Fowler, and Peter J. Mucha. Core-periphery structure in networks. *SIAM J. Appl. Math.*, 74:167–190, 2014.

110. Jeffrey S. Rosenthal. Optimal proposal distributions and adaptive mcmc. In *Chapman and Hall/CRC eBooks*, pages 93–112. Chapman and Hall/CRC, 2011.

111. Martin Rosvall and Carl T. Bergstrom. An information-theoretic framework for resolving community structure in complex networks. *Proc. Natl. Acad. Sci. USA*, 104:7327–7331, 2007.

112. Nicolò Ruggeri, Martina Contisciani, Federico Battiston, and Caterina De Bacco. Community detection in large hypergraphs. *Science Advances*, 9(28):eadg9159, 2023.

113. Marta Sales-Pardo, Roger Guimerà, André A. Moreira, and Luís A. Nunes Amaral. Extracting the hierarchical organization of complex systems. *Proceedings of the National Academy of Sciences*, 104(39):15224–15229, 2007.

114. Gerard Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Auckland, 1983.

115. Jungwoon Shin, Mogan Gim, Donghyeon Park, Sunkyu Kim, and Jaewoo Kang. Bipartite link prediction by intra-class connection based triadic closure. *IEEE Access*, 8:140194–140204, 2020.

116. Vishal B Siramshetty, Ivan Grishagin, Dac-Trung Nguyen, Tyler Peryea, Yulia Skovpen, Oleg Stroganov, Daniel Katzel, Timothy Sheils, Ajit Jadhav, Ewy A Mathé, and Noel T Southall. NCATS Inxight Drugs: A comprehensive and curated portal for translational research. *Nucleic Acids Research*, 50(D1):D1307–D1316, 10 2021.

117. R. Solomonoff and A. Rapoport. Connectivity of random nets. *Bulletin of Mathematical Biophysics*, 13:107–117, 1951.

118. Juliette Stehlé, Nicolas Voirin, Alain Barrat, Ciro Cattuto, Lorenzo Isella, Jean-François Pinton, Marco Quaggiotto, Wouter Van den Broeck, Corinne Régis, Bruno Lina, and Philippe Vanhems. High-resolution measurements of face-to-face contact patterns in a primary school. *PLoS ONE*, 6(8):e23176, 2011.

119. Yuji Sugita and Yuko Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chemical Physics Letters*, 314(1):141–151, 1999.

120. Toni Vallès-Català, Tiago P. Peixoto, Marta Sales-Pardo, and Roger Guimerà. Consistencies and inconsistencies between model selection and link prediction in networks. *Phys. Rev. E*, 97:062316, Jun 2018.

121. H. Wang, Q. Gu, J. Wei, Z. Cao, and Q. Liu. Mining drug-disease relationships as a complement to medical genetics-based drug repositioning: Where a recommendation system meets genome-wide association studies. *Clinical Pharmacology & Therapeutics*, 97:451–454, 2015.

122. David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, Nazanin Assempour, Ithayavani Iynkkaran, Yifeng Liu, Adam Maciejewski, Nicola Gale, Alex Wilson, Lucy Chin, Ryan Cummings, Diana Le, Allison Pon, Craig Knox, and Michael Wilson. DrugBank 5.0: A major update to the DrugBank database for 2018. *Nucleic Acids Research*, 46(D1):D1074–D1082, 11 2017.

123. Chao Wu, Ranga C Gudivada, Bruce J Aronow, and Anil G Jegga. Computational drug repositioning through heterogeneous network clustering. *BMC Systems Biology*, 7:1–9, 2013.

124. Renchi Yang, Jieming Shi, Keke Huang, and Xiaokui Xiao. Scalable and effective bipartite network embedding. In *Proceedings of the 2022 International Conference on Management of Data*, page 1977–1991, New York, NY, 2022. Association for Computing Machinery.

125. Zhao Yang, René Algesheimer, and Claudio J. Tessone. A comparative analysis of community detection algorithms on artificial networks. *Scientific Reports*, 6:30750, 2016.

126. Tzu-Chi Yen and Daniel B. Larremore. Community detection in bipartite networks with stochastic block models. *Phys. Rev. E*, 102:032309, Sep 2020.

127. Jean-Gabriel Young, Giovanni Petri, and Tiago P. Peixoto. Hypergraph reconstruction from network data. *Communications Physics*, 4(1):135, June 2021.

128. Shan Yu, Hongdian Yang, Hiroyuki Nakahara, Gustavo S. Santos, Danko Nikolić, and Dietmar Plenz. Higher-Order Interactions Characterized in Cortical Activity. *Journal of Neuroscience*, 31(48):17514–17526, 2011.

129. Wen Zhang, Xiang Yue, Weiran Lin, Wenjian Wu, Ruoqi Liu, Feng Huang, and Feng Liu. Predicting drug-disease associations by using similarity constrained matrix factorization. *BMC Bioinformatics*, 19:233, June 2018.

130. Xiao Zhang, Travis Martin, and M. E. J. Newman. Identification of core-periphery structure in networks. *Phys. Rev. E*, 91:032803, 2015.

131. Xiao Zhang and M. E. J. Newman. Multiway spectral community detection in networks. *Phys. Rev. E*, 92:052808, 2015.