ABSTRACT

Title of dissertation: MINING AND TESTING ON HIERARCHICAL STOCHASTIC BLOCK MODELS

AlFahad AlQadhi, Doctor of Philosophy, 2025

Dissertation directed by: Professor Vince Lyzinski
University of Maryland
Department of Mathematics

The rise in complexity of network data in neuroscience, social networks, and protein-protein interaction networks has been accompanied by efforts to model and understand these data on different scales. A key multiscale network modeling technique posits a hierarchical structure in the network. One such example of hierarchical modeling is the hierarchical stochastic blockmodel, which seeks to model complex networks as being composed of community structures repeated across the network. Incorporating repeated structure allows for parameter tying across communities, reducing the model complexity compared to the traditional block model. In this work, we describe a model that naturally expresses networks as a hierarchy of sub-networks with a set of motifs repeating across it and formally define the subgraph nomination framework with an emphasis on the notion of a user-in-the-loop in the subgraph nomination pipeline.

MINING AND TESTING ON HIERARCHICAL STOCHASTIC BLOCK MODELS

by

AlFahad AlQadhi

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2025

Advisory Committee:
Professor Vince Lyzinski, Chair/Advisor
Professor Deep Ray
Professor Lizhen Lin
Professor Yun Yang
Professor Robert Patro

Acknowledgments

I would like to express my gratitude to my advisor Dr. Vince Lyzinski for his guidance, extreme patience, and feedback that were crucial to my success. His knowledge and experience in the field have inspired my work, and his kindness helped me push through the difficult parts of this journey. I thank Dr. Kedem for introducing us.

I also extend my appreciation to my co-advisor Dr. Kieth Levin for his help on my second project. His insight and encouragement helped me form a better understanding of my work.

In addition, I would like to thank all the faculty and staff, who were teachers and friends. The wonderful staff at the math department are caring and involved; they are why any of us can do what we do. Special thanks to Liz Wincek, Stephanie Padgett, and Jessica Sadler, from running social events to being there for students in hard times, they made this journey easier.

I was lucky to have to find many friends during graduate school, thank you all, Kostantinos Pantazis for being a good friend and encouraging me throughout, Vaughn Osterman and Ben Goldschlager, my roommates, who made the solitary periods easier, Foivos Chnaras for lending a sympathetic ear when I needed one, your support has helped me pull this off.

Finally, I extend my thanks to the HLTCOE, DARPA, and the University of Maryland for all the funding I received.

Table of Contents

Acknowledgements ii

1 Introduction 1
- 1.1 Subgraph Nomination . . . 1
- 1.2 Hierarchical structures in network science . . . 2
- 1.3 Motifs and repeated structures . . . 4
- 1.4 Hierarchical Stochastic Block Model . . . 6

2 Preliminaries and Definitions 8
- 2.1 Generalized Graphs . . . 8
- 2.2 Graph Properties . . . 11
- 2.3 Stochastic Graphs . . . 16
  - 2.3.1 Block Models . . . 16

3 Analysis Methods for Graphs 28
- 3.1 Tasks and Problems Related to Graphs . . . 28
  - 3.1.1 Clustering . . . 28
  - 3.1.2 Graph and Subgraph Matching . . . 29
  - 3.1.3 Vertex and Subgraph Nomination . . . 31
- 3.2 Spectral Methods . . . 42
- 3.3 Graph Neural Networks . . . 47

4 Simulations and Applications to Hierarchical Stochastic Block Models 56
- 4.1 User-in-the-loop Supervision . . . 56
  - 4.1.1 The (Theoretical) Benefit of the User-in-the-loop . . . 60
  - 4.1.2 Simulations and Real Data Experiments . . . 65
- 4.2 Testing for Repeated Motifs and Hierarchical Structure in Stochastic Block-models . . . 78
  - 4.2.1 Main Results . . . 79
  - 4.2.2 Experiments . . . 87

5 Conclusions and open problems 100

A Detailed Proofs 104
A.1 Proof of Theorem 4.1 104
A.2 Supporting results 109
A.3 Proof of Lemma 4.1 110
A.4 Proof of Theorem 4.2 112
A.5 Proof of Theorem 4.3 114
A.6 Proof of Theorem 4.4 116

Bibliography 118

Chapter 1: Introduction

Networks, which encode interactions among entities, are ubiquitous in the sciences, arising in fields as diverse as ecology [20, 52, 101], sociology [18, 44], economics [39, 96], and neuroscience [102, 103]. As they have become more prevalent, network data has become increasingly complex, and the need to model networks at multiple scales is necessary for efficient modeling and analysis; see, for example, [2]. A key idea for understanding hierarchical networks is the idea of repeated motif structure [66], as this allows for parsimonious model descriptions and enhanced structure discovery in complex network settings.

In this chapter we will briefly describe the network inference tasks pursued in this dissertation.

1.1 Subgraph Nomination

Stated succinctly, the subgraph nomination problem is as follows: given a subgraph or subgraphs of interest in a network G_1, we seek to find heretofore unknown subgraphs of interest in G_1 or in a second network G_2. The subgraph nomination problem can be viewed as an amalgam of the problems of noisy subgraph detection [22, 50, 63, 75, 104] and vertex nomination [29, 37, 65, 70], in which our task is to produce a rank-list of candidate subgraphs, with (ideally) unknown subgraphs of interest concentrating at the top of the rank list. While seemingly simple on its surface, subgraph nomination necessarily entails the often non-trivial combination of subgraph detection (i.e., we have to find can-

didate subgraphs to rank) and multiple graph comparison methodologies (i.e., we have to rank the candidate subgraphs). We note here that the subgraph nomination problem considered herein is closely related to many existing graph query problem formulations (see, for example, the work on query by example in [62, 76]). Our novel contribution is the novel statistical framework underlying our subgraph nomination formulation, which allows us to frame and rigorously study the problem (and the effect of a user-in-the-loop) in a number of popular statistical network models, with the ultimate goal of establishing results of statistical consistency for subgraph nomination akin to the analogues in the vertex nomination literature [64, 65]. For more detail on Subgraph nomination, see Sections 3.1.3 and 4.1.

One application area for subgraph nomination is discovering repeated structure in hierarchical networks.

1.2 Hierarchical structures in network science

The presence of hierarchical structure in networks has long been observed and exploited in network science and statistical network inference [see, for example, 26, 27, 40, 60, 82]. One of the early settings in which hierarchical structure was emphasized is the setting of graph clustering, as the presence of such structure motivates the application of hierarchical or agglomerative clustering methods rather than more classical techniques (e.g., k-means). Early work to estimate the tree structure of the hierarchy in a network was often optimization based [e.g., 6, 26]. Although these methods tend to be computationally efficient, theoretical optimality is difficult to guarantee. Nonetheless, these methods are widely used in practice. For example, [27] estimates the hierarchical structure in a metabolic network, a terrorist association network, and a food chain network. They also demonstrate the utility of hierarchical structures for edge prediction using the resulting attachment probability parameters.

Similarly, analysis and clustering at multiple levels of resolution have been a popular line of inquiry in network analysis. In [6], modularity is used to cluster graphs at multiple scales by adjusting the degree of each node without affecting the topology of the network. This is achieved by an additive shift to the weight of the self-edges. By varying the strength of this shift, clusters at different resolutions are retrieved. However, this algorithm is based on an expensive algorithm that would be applied for each of these shifts, but using heuristics for the optimization of modularity, as in [28], makes the method computationally feasible.

One feature of hierarchical structure that we will emphasize in what follows is the role of the tree-structured partition in describing the hierarchy. From a generative model perspective, the recent work on $\mathbb{T}$-stochastic graphs [34] encompasses a wide range of statistical network models as special cases. In these models, a latent tree structure drives network formation. The leaves of this tree correspond to the vertices in the observed graph, and the probabilities of edges among these vertices are inversely related to the distances between the leaves on the tree. To infer underlying hierarchical tree structure in a network, [60] and [58] use spectral clustering to partition the network into two subgraphs, and the process is repeated recursively until a stopping criterion is reached. This partitioning yields a binary tree structure in which the root node represents the full graph, its children represent the two subgraphs obtained in the first partition, and so on. The first obvious limitation of these methods is the restriction to binary trees, as in each step the graph is partitioned into exactly two subgraphs. In [60], it is further required that the tree is a complete binary tree: if one subgraph triggers the stopping criterion, other subgraphs at the same depth of the tree are not tested for further subdivisions. [58] do away with this requirement, allowing for a more general class of models. Their method generalizes beyond binary trees and other symmetry assumptions, as it captures a much more general class of hierarchical graphs. Alternatively, [13] uses spectral embeddings for a K-way clustering algorithm by choosing the centers to be the nodes whose embedding is the furthest from other centers. What is of

note is that their model includes perturbations on the adjacency matrix, which is relevant to the models we describe in this work.

Often, these methods—and indeed most clustering methods—are posited in the setting of assortative models: they assume that connections to vertices in the same community are more likely than connections to vertices in other community at each level of the tree. In the setting of hierarchical structure, this assortativity manifests as follows. For two nodes (from possibly different communities), the connection probability is inversely related to the distance between their communities on the partition tree. While this assortativity assumption is appropriate for many real networks, it need not always be present and methods to tackle disassortative networks (and the continuum between) are needed. [82] discusses a method, optimizing a posterior likelihood, which depends not on assortativity but instead on the similarity in the connection probabilities.

1.3 Motifs and repeated structures

Alongside hierarchical structures, networks frequently exhibit repeated structures, often termed *motifs*. The simplest notion of these repeated structures is that of isomorphic subgraphs [see 9, for discussion and recent advancements on the graph isomorphism problem]. Isomorphism between subgraphs is often too rigid in practice, and the stringent conditions of isomorphism between subgraphs can be relaxed. One such relaxation can be done using a metric on the similarity between two subgraphs and a threshold for when two subgraphs are considered the similar enough [86, 104, 116]. Such a framework is useful in the context of random graphs, as two realizations of the same graph model need not be isomorphic but *are* likely to exhibit structural similarities. The subgraph similarity search problem, analogous to subgraph isomorphism search, searches for subgraphs similar to a given structure. Recently, numerous efficient algorithms for motif detection and subgraph comparison have

been developed with applications in protein-protein interaction (PPI) networks [119], the Wikipedia database [47], and the AIDS Antiviral Screen dataset [98], to name but a few.

In our present hierarchical network setting, motifs can represent repeated structures within the network [66]. This repetition can be envisioned as occuring at the generative model level, which allows for repetition of coarse structures (i.e., at the level of communities) while allowing variation at finer scales (i.e., within or among communities). As an example, in the context of connectomics [102, 103], while we see symmetry between brain hemispheres at gross anatomical scales, there is significant deviation at the local level [90].

In this work, we propose a generative process that can be used to describe different levels of refinement and similarity across levels of a hierarchy, statistical methodology to test for this repetition, and test these methods on simulated and real data. For example, in a binary graph G we may want to search for isomorphic subgraphs in P considering it a weighted graph G_P, since this corresponds to two subgraphs that have the same distribution. By the nature of the observed graph being a random instantiation of this distribution, the isomorphic subgraphs in G_P may not be isomorphic in G. This does not improve the complexity of subgraph matching; instead, it extends it to the probabilistic setting. For example, finding cliques (i.e., complete subgraphs) has applications in social interaction networks; describing social roles [33], distinguishing types of social capital [83], communication patterns [111], and detecting the access of geography and technology in trade networks [74]. However, this extension can help reduce complexity under certain conditions; for example, [72] provides a methodology for obtaining an approximate solution by starting with a seed of vertices that are known to be matched and under the condition that their distributions are correlated, an asymptotic guarantee can be given. Assuming that we have some knowledge of how to decompose the graph, we can extend their work by decomposing the graph in a manner similar to [73].

Another line of relaxation of the isometry can be realized from the lens of the hier-

archy discussed earlier. In other words, the fine structure of two subgraphs may not be isometric, but by looking at larger structures within these subgraphs, an isomorphic structure appears. In fact, in the context of the brain connectome, while we see symmetry on the large scale, there is asymmetry on the finer scales [90]. Recently, numerous efficient algorithms for motif detection and subgraph comparison have been developed with, for example, applications in the context protein-protein interaction (PPI) networks [119], the Wikipedia database [47], and the AIDS Antiviral Screen dataset [98] In this work, we propose an agglomerative process that can be used to describe different levels of refinement, statistical methodology to test for repetition, and test these methods on simulated and real data.

1.4 Hierarchical Stochastic Block Model

Effective methods for statistical analysis of network data require that models accurately capture both large-scale network topology and fine-grained network structure [60]. Moreover, recent work has incorporated hierarchical structure into existing statistical network models, with the stochastic blockmodel and random dot product graph being a natural setting for this [see, for example, 4, 60, 82]. In these settings, the hierarchical stochastic blockmodel (HSBM) is often posited as a low-rank latent space model, and various spectral methods can be used to approximate the latent vectors. Then, recursive k-means clustering can be applied to retrieve the nested community structure. To motivate the repeated motif structure in the HSBM further, we consider the simple example of a social network of a large research university. Within a department, one is likely to have faculty, staff, and graduate students. These different sub-populations are likely to interact in ways that are (approximately) repeated across departments. For example, faculty tend to interact preferentially with staff and other faculty, graduate students interact with specific staff and/or

faculty, etc., and we expect similar patterns of this sort to manifest in many different departments across campus. This suggests a model in which we allow connection probability patterns to be replicated across different parts of a network. Beyond the presence of repeated high-level structures across a network, we may wish to allow hierarchical structures in our model.

Continuing with our example of modeling a university social network, within a single department, there are groups of students, staff, or faculty who interact differently with one another; e.g., groups of faculty who work on different research sub-areas, or cohorts of students who joined a program in different years. Zooming out to the level of modeling interactions across multiple universities, we may have institutions similar interaction patterns across their departments. Such patterns may be indicative of certain features. For example, they may share the role as the largest major in their respective universities, or an interest in the same areas of research. This presents then two aims: to capture both hierarchical *and* repeated structure in a network. This motivates the focus of the present manuscript, the *repeated motif hierarchical stochastic blockmodel* (RMHSBM). This model extends the well-studied stochastic blockmodel [SBM 46] to incorporate repeated structure into the hierarchical stochastic blockmodel [HSBM 66].

Chapter 2: Preliminaries and Definitions

This chapter will first develop a working definition of graphs that will be used throughout the work. In addition, we will introduce concepts and properties of graphs that will help in further analysis; these properties in the setting of random graphs are sometimes referred to as statistics. Finally, we end with a discussion of graphs as random variables and introduce relevant distributions that will build on each other, leading to the distribution that is the subject of this work, the *Hierarchical Stochastic Block Model*. We make a distinction between two classes of graphs, one based on a probability attachment matrix and another based on latent variables in the Cartesian space.

2.1 Generalized Graphs

The first element needed to build a graph is a set of vertices we label V. In general V can be any set, but it is enough to consider $V \subseteq \mathbb{N}$ or $V \subseteq \mathbb{R}$ for countable and uncountable sets. To this end, we must introduce the second element of a graph, the set of vertex feature functions $\mathscr{F}_V$, a set of functions $f : V \to S_f$ where S_f is the set of features for function f. These functions help us encode information on these nodes. For example, when considering social networks as a graph, we may want to encode people's names, addresses, and birthdays as in this survey of graph analysis methods, [100]. In [67] the authors consider a graph of websites, we may need to incorporate the website address or date of creation as features on the nodes. For each of these cases, we use integers to number

the nodes and define functions that map these integers onto the set of names, birthdays, etc.

Perhaps the most important aspect of a graph is its ability to convey relations between its vertices; to this end, we introduce the next element, the set of edge functions E. This set is composed of functions $f : V^{k_f} \to W_f$, where k_f is the size of the tuples that form an edge and $W_f \subseteq \mathbb{R}$ is the set of weights. Usually, in the case where $k_f = 2$ we refer to them as edges and in the case where $k_f > 2$ as hyperedges. Graphs with hyper edges are sometimes called hypergraphs. Alternatively, these functions can be defined using power sets, in which case $f : \mathscr{E} \to W_f$ where $\mathscr{E} \subseteq P(V)$. The latter definition allows us to encode edges with an arbitrary number of edges in a single edge function, mathematically if $n,m,l \in V$ both $\{n,m\}$ and $\{n,m,l\}$ can be in the domain of f. We can make these definitions equivalent by imposing the restriction $|e| = k_f$ for all $e \in \mathscr{E}$. In a variety of cases, it is enough to consider $W_f = \{0,1\}$, for example on a follower-based social network such as Instagram. Note that in this case, the function is not symmetric; sometimes $f(n,m) \neq f(m,n)$. In other cases such as Facebook this is not the case. In [67], weights are used to indicate the number of people who traveled from one website to another. In addition, we may need to include information related to these edges, such information is encoded using the edge feature function set $\mathscr{F}_E$, a set of functions $g_f : V^{k_f} \to S_{g_f}$, each associated with an edge function and a set of features S_{g_f}. For our example of Instagram, we may want to include information on the date m followed n or metrics of the extent to which they viewed their content. Note that weights could be considered as features instead of range sets on the edge functions. The utility of vertex and edge features is prevalent in Graph Neural Networks (GNNs).

Definition 2.1 (Generalized Graph)**.** *A generalized graph $G = (V,E,\mathscr{F}_V,\mathscr{F}_E)$ is defined by a set of vertices $V \subseteq \mathbb{R}$ with edges assigned by the set of edge functions $E = \{f | V^{k_f} \to W_f,\ W_f \subseteq \mathbb{R}\}$, where W_f is the set of weights for the edge function f, with a set of vertex*

feature functions $\mathscr{F}_V = \{f | f : V \to S_f\}$ *and a set of edge feature functions* $\mathscr{F}_E = \{g_f | g_f : V^{k_f} \to S_{g_f},\ f \in E\}$*, where* $S_\cdot$ *are the sets of features associated with these functions.*

Remark 1 (Directed Hypergraphs)**.** *The set* E *of edge functions in directed hypergraphs takes two input sets, the heads* H *and the tails* T *of the edge. In other words* $f : P(V) \times P(V) \to W_f$*, to maintain the size of the edge in this case we can restrict the function to the set* $\mathscr{E} : \{(H,T) \in P(V) | \text{such that } |H \cup T| = k_f \|\}$.

Restricting our attention to graphs where $|V| \in \mathbb{N}$ and $k_f = 2$, the set of edge functions E are of the form $f : V \times V \to \mathbb{R}$, and disregarding features momentarily, a graph can then represented as a matrix $\mathscr{A} = \{A_f \in \mathbb{R}^{|V| \times |V|}, f \in E | A_f[i,j] = f(i,j)\}$. We can generalize this construction for hypergraphs using multidimensional arrays where

$$\mathscr{A} = \{A_f \in \mathbb{R}^{\times^{k_f}|V|}, f \in E | A_f[\vec{v}] = f(\vec{v}) \text{ where } \vec{v} \in V^{k_f}\}$$

where $\times^{k_f}|V| = |V| \times |V| \times |V| \ldots$ k_f times. This view of graphs will be crucial to understanding the motivation of many analysis methods and distributions when considering graphs as random variables. We refer to the resulting arrays as adjacency arrays, or matrices, of a graph.

Furthermore, if the set of edge feature functions $\mathscr{F}_E$ consists of functions $g_f : V^{k_f} \to \mathbb{R}$ the we can represent these features via the array

$$\mathscr{A} = \{A_{g_f} \in \mathbb{R}^{\times^{k_f}|V|}, g_f \in \mathscr{F}_E | A_{g_f}[\vec{v}] = g_f(\vec{v}) \text{ where } \vec{v} \in V^{k_f}\}$$

One use for such constructions returns us to the social networks example of Facebook. One can represent the date on which a friend is added using its distance from a specific period, let us call the resulting matrix A and represent the friendship edge function with matrix E. Then we may use $E \circ A$ for analysis, which will put more emphasis on connections made

in that window without losing information regarding the connections made outside it. Of course, we could have defined the original edge function to represent these weights, and indeed in general these two are equivalent, but the framework will depend on the source of the data we analyze. It is therefore useful to understand this framework for the analysis of networks in the absence of our control over the collection and representation of the data. In figure 2.1, we have an example of a graph for a network of 13 runners. To describe it using this frame work we have the following sets,

$$V = \{1,2,3,4,5,6,7,8,9,10,11,12,13\}$$

$$E = \{f_1 : \text{Relationships},\ f_2 : \text{Teams for 3 person races},\ f_3 : \text{Teams for 4 person races}\}$$

$$\mathscr{F}_V = \{f_{name}, f_{age}\}$$

$$\mathscr{F}_E = \{g_{f_1} : \text{Type},\ g_{f_2} : (\text{Date}, \text{Rank}),\ g_{f_3} : (\text{Date}, \text{Rank})\}$$

For the edges we have three functions, the first describes if two runners have a relationship with a feature describing the nature of the relationship. The second edge functions describe forming 3 person race teams shown in the figure as pink regions. Associated with these team edges are features that describe the time of the team's last race and their placement in that race. Similarly, the blue regions form 4 person teams with features describing the time of the team's last race and their placement in that race. Finally, there are two feature functions for the node that describe the name (redacted) and age of each runner.

2.2 Graph Properties

In this section we will use our frame work to formalize some important properties of graphs. The first property is related to the symmetry of the edge functions of the graph, namely the notion of *directed* and *undirected* graphs. An *undirected* graph is one where

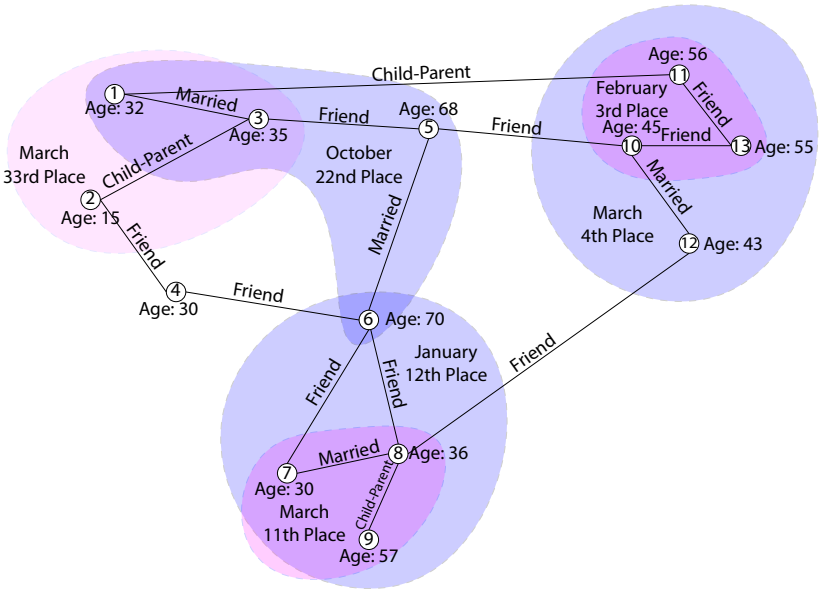


Figure 2.1: Example of a generalized graph with 13 nodes.

for all $f \in E$ and nodes $i, j \in V$ $f(i,j) = f(j,i)$, while *directed* graphs do not pose such a restriction. However, we could instead use two symmetric functions f_{in}, f_{out} to represent the function f, where

$$f_{in}(i,j) = f_{in}(j,i) = \begin{cases} f(i,j) \text{ if } i < j \\ f(j,i) \text{ if } i > j \end{cases}$$

$$f_{out}(i,j) = f_{out}(j,i) = \begin{cases} f(i,j) \text{ if } i > j \\ f(j,i) \text{ if } i < j \end{cases}$$

which means that

$$f(i,j) = \begin{cases} f_{in}(i,j) \text{ if } i < j \\ \\ f_{out}(i,j) \text{ if } i > j \end{cases},$$

which means that a directed graph can be represented as an undirected graph using this construction. The utility of this reconstruction lies in our ability to use analysis methods designed for undirected graphs by combining these two functions in an appropriate way. This method of analysis for undirected graphs is termed graph symmetrization, for example, [92] discusses symmetrization methods for clustering.

However, notice that we have not yet discussed the cases for $f(i,i)$, termed *self-edges*. Graphs that have $f(i,i) = 0$ for all $i \in V$ and $f \in E$ are called hollow or loop-free graphs; since most of the applications we will discuss will not have a notion of self-edges, all graphs will be hallow unless stated otherwise.

Sometimes we would like to restrict ourselves to a subset of nodes $V' \subset V$ and the edges between them in our discussion, in that case we say the subgraph induced by G on the nodes V', denoted $G(V')$. Formally, let $G = (V, E, \mathscr{F}_V, \mathscr{F}_E)$ then the subgraph induced by graph G on the set of nodes $V' \subset V$ is $G(V') = (V', E, \mathscr{F}_V, \mathscr{F}_E)$.

The neighbors of a node $i \in V$ with respect to an edge function $f \in E$ denoted $\mathscr{N}_f(i)$, that is, all the nodes of V that share an edge with i, more formally

$$\mathscr{N}_f(i) = \{j \in V | f(i,j) \neq 0\}$$

and we refer to $G(\mathscr{N}_f(i))$ as the neighborhood of i. However, we will break from this convention and use neighbors and neighborhoods interchangeably for $\mathscr{N}_f(i)$, and call $G(\mathscr{N}_f(i))$ the subgraph induced on the neighborhood of i.

Remark 2. *Note that this definition specifies a direction in directed graphs by accepting nodes that satisfy $f(i,j) \neq 0$ but not $f(j,i) \neq 0$. In general, a neighborhood is made up of nodes that can be reached starting at a node i. There are then two notions of neighborhoods in directed graphs, out-edge neighborhood $\mathscr{N}_{f,out}(i)$ defined above and in-edge neighborhood*

$$\mathscr{N}_{f,in}(i) = \{j \in V | f(j,i) \neq 0\}.$$

Remark 3 (Neighborhoods in Directed Hypergraphs)**.** *We can generalize these definitions for hypergraph edges, in which case we define the neighborhood of vertex $i \in V$ as*

$$\mathscr{N}_f(i) = \{j \in H \cup T, (H,T) \in \mathscr{E} | f(H,T) \neq 0 \text{ and } i \in H \cup T\}.$$

The in-neighborhood can be defined by

$$\mathscr{N}_{f,in}(i) = \{j \in H, (H,T) \in \mathscr{E} | f(H,T) \neq 0 \text{ and } i \in T\}.$$

We contend that all the definitions that follow can be extended similarly without stating them for brevity.

This definition can be expanded to the neighborhood of a set of nodes $v \subset V$,

$$\mathscr{N}_f(v) = \{j \in V | \text{ there exists } i \in v \text{ such that } f(i,j) \neq 0\}$$

This set is sometimes referred to as the 1-neighborhood of i or v with respect to f, leading to a generalization to the l-neighborhood of v with respect to f denoted $\mathscr{N}_f^l(v)$. Formally defined using the following recursive relation,

$$\mathscr{N}_f^l(v) = \mathscr{N}_f^{l-1}(v) \cup \mathscr{N}_f^1(\mathscr{N}_f^{l-1}(v))$$

with $\mathscr{N}_f^1(v) = \mathscr{N}_f(v)$. The set of connected components $\mathscr{C}$ of a graph with respect to an edge function $f \in E$ is a partition of V such that for all $C \in \mathscr{C}_f$ we have $\mathscr{N}_f(C) = C$, and the largest connected component is $\underset{C \in \mathscr{C}_f}{\text{argmax}}|C|$. If $\underset{C \in \mathscr{C}_f}{\text{argmax}}|C| = V$, we say that the graph G is a connected graph.

Another notion related to neighborhoods is the degree of a node $i \in V$ denoted $degree_f(i)$ and is equal to $|\mathscr{N}_f(i)|$ for undirected graphs. For directed graphs, we have an out-degree and an in-degree which are the size of the out-edge neighborhood and the in-edge neighborhood, respectively. Finally, the distance $d_f(i,j)$ between two nodes $i, j \in V$, which is the length of the smallest path between node i and node j, can also be written in terms of neighborhoods as follows,

$$d_f(i,j) = \min\{L | j \in \mathscr{N}_f^L(i)\} \tag{2.1}$$

If there is indeed any such L and ∞ if $|\{L | j \in \mathscr{N}_f^L(i)\}| = 0$. For directed graph we replace $\mathscr{N}_f^l(i)$ with $\mathscr{N}_{f,out}^l(i)$ and note that the distance is no longer symmetric, i.e. sometimes $d_f(i,j) \neq d_f(j,i)$.

In practice, however, adjacency matrices give a fast way to retrieve these properties for graphs with a finite set of vertices V. Focusing our attention on a single edge function and its associated adjacency matrix $A_{i,j} = f(i,j)$ for all $i, j \in V$, an edge function produces an undirected graph if $A - A^T = 0$. The 1 neighborhood is

$$\mathscr{N}_f^1(i) = \{j \in V | A_{ij} \neq 0\}$$

while the l neighborhood takes the form

$$\mathscr{N}_f^l(i) = \bigcup_{k=1}^{l} \{j \in V | A_{ij}^k \neq 0\}$$

Where A^k is the standard k-th power of the matrix k. Meanwhile, $degree(i) = sum(A_{i.})$ where $A_{i.}$ is the i-th column of the matrix A. The distance $d(i,j) = \min\{d > 1 | A^d_{i,j} \neq 0\}$. For directed graphs, the 1 in-neighborhood is

$$\mathscr{N}^1_{f,in}(i) = \{j \in V | A_{ji} \neq 0\}$$

while the l neighborhood takes the form

$$\mathscr{N}^l_{f,j}(i) = \bigcup_{k=1}^{l} \{j \in V | A^k_{ji} \neq 0\}$$

and $degree_{in}(i) = sum(A_{.i})$ where $A_{.i}$ is the i-th row.

2.3 Stochastic Graphs

In this section we will describe how we can construct random variables for graphs, then give specific example distributions leading into the hierarchical stochastic block model. For simplicity, we will focus on a special case of graphs $G = (V,E)$ where $|V| \in \mathbb{N}$ and E is a function $E : V \times V \to \{0,1\}$. In this special case, the graph can be represented as a single adjacency matrix $A_{ij} = E(i,j)$. We consider graph distributions with edges that take a Bernoulli distribution, that is, $A_{ij} = A_{ji} \sim Bernoulli(P_{ij})$, then the matrix $\mathbb{E}(A) = P \in (0,1)^{|V| \times |V|}$ is referred to as the attachment probability matrix. The matrix P can be a random variable itself.

2.3.1 Block Models

One useful random construction of P is *Random Dot Product Graphs*, which will allow us to estimate P from the spectral decomposition of A, this will be explored in detail in Section 3.2.

Definition 2.2 (Random Dot Product Graph (RDPG)). *Let $\Omega_d \subset \mathbb{R}^d$ where for any $x, y \in \Omega_d$, $x^T y \in [0,1]$ with a distribution F over Ω_d. A graph $G = (V,E)$ with vectors $x_1, \ldots, x_{|V|} \overset{i.i.d}{\sim} F$, $n = |V|$, and adjacency matrix A, where for every $i, j \in [n]$, $A_{ij} = A_{ji} \sim Bernoulli(x_i^T x_j)$, then G is a* Random Dot Product Graph. *We say*

$$G \sim RDPG(d, F, n).$$

We will define distributions which impose certain structures on P. The base case, where the entries of P are all equal (i.e., all x_i are equal in the RDPG) , is the Erdös-Rényi random graph.

Definition 2.3 (The Erdös-Rényi Random Graph). *Let $G = (V,E)$ be a graph with $V = [n]$ and let $p \in (0,1)$ then the graph G follows the Erdös-Rényi distribution $G(n,p)$ if for all $i, j \in V$ we have*

$$A_{ij} \sim Bernoulli(p)$$

An analogous definition can be obtained by considering the total number of edges in the graph to follow a binomial distribution, as described in [41]. In this definition, a graph is constructed by assigning K edges randomly to the indices of E_{ij}, denoted $G(n,K)$. Modeling a graph in this way assumes homogeneity across the entire network, a very strict assumption in some settings. For example, in the setting of interaction networks, as in social networks, we observe that communities form in which the interaction inside the community is larger than outside of them. For example, people who live in the same country are more likely to interact with each other.

In [46], the stochastic block model (*SBM*) is suggested for this purpose. In an *SBM* with K communities we have a community assignment probability vector π and a community assignment vector τ so that for all $i \in V$ and $k < K$ we have $P(\tau_i = k) = \pi_k$. In addition,

we have the block probability attachment matrix $B \in (0,1)^{K\times K}$, where for any $i, j \in V$ we have $A_{ij} \sim Bernoulli(B_{\tau_i\tau_j})$. Formally,

Definition 2.4 (Stochastic Block Model (*SBM*))**.** *We say that an n-vertex random graph G is an instantiation of a stochastic blockmodel (SBM) with parameters* (n,K,B,π)*, and write* $A \sim \text{SBM}(n,K,B,\pi)$*, if*

i. The block membership vector $\pi \in \mathbb{R}^K$ *satisfies* $\pi(i) \geq 0$ *for all* $i \in [K]$*, and* $\sum_i \pi(i) = 1$*;*

ii. The vertex set $V = V(G)$ *is the disjoint union of K blocks,* $V = \mathscr{B}_1 \sqcup \mathscr{B}_2 \sqcup \cdots \sqcup \mathscr{B}_K$*, where each vertex* $v \in V$ *is independently assigned to a block according to a* Multinomial$(1,\pi)$ *distribution. For each vertex* $v \in V(G)$*, let* τ_v *be the block that v is assigned to.*

iii. The probability attachment matrix $B \in [0,1]^{K\times K}$ *is a symmetric matrix. Conditional on the block assignment vector* τ*, for each pair of vertices* $\{i,j\} \in \binom{V}{2}$*,* $A_{ij} = A_{ji} \overset{ind.}{\sim}$ Bern$(B[\tau_i,\tau_j])$.

Note that the SBM is a collection of Erdös-Rényi random graphs with connection probabilities between them. We can also consider an RDPG based block model.

Definition 2.5 (Block Model Random Dot Product Graph)**.** *Let* $\pi \in [0,1]^K$ *where* $\sum \pi_i = 1$*. A graph* $G = (V,E)$ *with vectors* $x_1,\ldots,x_k$*, block assignment function* $\tau : [n] \to [K]$*, and adjacency matrix A. If for every* $i \in [n]$ *and* $k < K$ *then* $\mathbb{P}(\tau(i) = k) = \pi_k$*, and for every* $i, j \in [n]$ $A_{ij} \sim Bernoulli(x^T_{\tau(i)}x_{\tau(j)})$*, then G is a* Block Model Random Dot Product Graph*. We say*

$$G \sim BRDPG(d,F,n,\pi).$$

An important special case of graphs for us are trees, since we will use them to dive deep into Hierarchical Stochastic Block Models. These trees and their properties will be

instrumental for describing hierarchies and understanding them will make understanding hierarchical models easier, their utility in describing data, properties, and methods of analysis.

As a graph, a tree is undirected, where any two vertices have exactly one path between them, and there are no cycles. Since any two vertices must have a path between them we notice that this graph would be connected, if we relax the condition so that we have no more than one path the structure would not be connected. In fact, this would be the union of two or more trees, we refer to these as forests. Since we would like to have a hierarchy we will root our tree, this means that we will select a vertex designate it as the root. Formalizations of this notion have previously been pursued in [66]. Here, we adopt the following convention to define the Hierarchical SBM. For a rooted tree $T = (V, E, r)$ (so that the tree is rooted at r, V denotes the set of vertices in the tree and E the set of edges), we define the following notation:

- We let $\mathscr{L}(T)$ denote the leaves of T and $\mathscr{I}(T)$ denote the internal nodes of T, so that $V = \mathscr{L}(T) \cup \mathscr{I}(T)$;
- For any node t in the tree $T = (V, E)$, we write $\mathscr{D}(t) \subseteq V$ to denote the descendants of t; note that $t \notin \mathscr{D}(t)$ by convention;
- For a node $t \in V(T)$, we let $\delta(t)$ denote its depth, i.e., the length of the path from the root to t; by convention, $\delta(r) = 0$. If $\delta(v) = k$ for all $v \in \mathscr{L}(T)$, then we say the tree is a *depth-k* tree;
- For two nodes t_1, t_2 in tree T, we let $\mathrm{LCA}(t_1, t_2)$ denote their lowest common ancestor (i.e., the common ancestor with the largest depth);
- For two nodes t_1, t_2 in tree T we let $\mathrm{LCA}\downarrow(t_1, t_2) \in \binom{V(T)}{2}$ denote the children of first encountered on the paths from $\mathrm{LCA}(t_1, t_2)$ to t_1 and t_2.

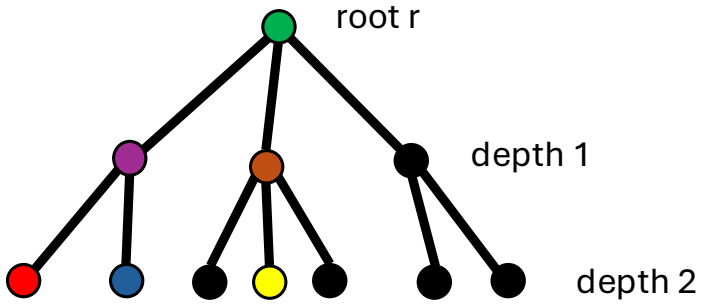


Figure 2.2: An example of a depth 2 rooted tree. Here (denoting nodes by their color for simplicity) LCA(red,blue)=purple, and LCA(yellow,blue)=green. We also have that LCA $\downarrow$(red,blue)= {red,blue} and LCA $\downarrow$(yellow,blue)= {purple,orange}.

See Figure 2.2 for an example of this notation in a simple depth 2 tree structure. Note that in the definition below, a 1-level Hierarchical SBM is simply a standard SBM.

Similar to the idea of $\mathbb{T}$-*stochastic graphs* [34], the Hierarchical SBM will be defined with an implicit tree structure. Here, the tree will encode block-membership at different levels of the hierarchy.

Definition 2.6 (Hierarchical SBM)**.** *We say that an n-vertex random graph G is an instantiation of a L-level Hierarchical Stochastic Block Model (HSBM) with parameters*

- $n \in \mathbb{Z}^+$*, the number of vertices in the graph;*
- $K \in \mathbb{Z}^+$*, the number of metablocks at Level-*1 *(i.e., at depth 1 in the tree);*
- $B \in [0,1]^{K\times K}$ *(a symmetric matrix), the Level-1 metablock–to–metablock communication matrix*
- $\pi \in \Delta^{K-1}$ *(the $K-1$ simplex), the Level-1 metablock probability assignment vector;*
- $\{S_i\}_{i=1}^K$*, a collection of $L-1$-level HSBM's*

if the following conditions hold (we write this as $G \sim \mathrm{HSBM}(n,K,B,\pi,\{S_i\}_{i=1}^K)$):

i. The vertex set $V = V(G)$ is the disjoint union of K metablocks $V = \mathscr{B}_1 \sqcup \mathscr{B}_2 \sqcup \cdots \sqcup \mathscr{B}_K$, where each vertex $v \in V$ is independently assigned to a metablock according to

a Multinomial$(1,\pi)$ *distribution. For each vertex* $v \in V(G)$*, let* τ_v *be the metablock that* v *is assigned to.*

ii. Conditional on the metablock assignment vector $\tau \in [K]^n$*, for each pair of vertices* $\{i,j\}$ *with* $\tau_i \neq \tau_j$*, the level-1 HSBM structure is modeled via*

$$A_{ij} \overset{ind.}{\sim} \text{Bern}(B[\tau_i, \tau_j]).$$

iii. For each $i \in [K]$*, conditional on the block assignment vector* $\vec{\tau} = (\tau_v)$ *and* $|\mathscr{B}_i| = n_i > 0$*,* S_i *denotes the model for the* i*-th metablock at level-*2 *in the HSBM.* S_i*, the model for* $G[\mathscr{B}_i]$*, is a* $L-1$*-level HSBM of the form HSBM*$(n_i, K_{(i)}, B_{(i)}, \pi_{(i)}, \{S_{(i,j)}\}_{j=1}^{K_{(i)}})$ *(so that each* $S_{(i,j)}$ *is itself an* $L-2$*-level HSBM*$(|\mathscr{B}_{i,j}|, K_{(i,j)}, B_{(i,j)}, \pi_{(i,j)}, \{S_{(i,j,k)}\}_{k=1}^{K_{(i,j)}})$*, and so on). Moreover, conditional on the block assignment vector* $\vec{\tau} = (\tau_v)$*, the collection* $\{G[\mathscr{B}_k]\}_{k=1}^{K}$ *are mutually independent.*

iv. The structure within *each of the* $\{S_i\}_{i=1}^{K}$ *represent the level-2 HSBM structure, the structure* within *each of the* $\left\{\{S_{(i,j)}\}_{j=1}^{K_{(i)}}\right\}_{i=1}^{K}$ *the level-3 structure, and so on. At level-L, the* $S_{\vec{v}}$*'s (*$\vec{v} \in \mathbb{Z}^L$*) are structured as SBM's according to Definition 2.4; we will denote the block membership vector associated with* $S_{\vec{v}}$ *via* $\tau_{\vec{v}}$*, so that for a vertex conditioned on being in* $u \in \mathscr{B}_{\vec{v}}$*, we have* $\tau_{\vec{v}}(u) = k$ *with probability* $\pi_{\vec{v}}(k)$ *for each* k *in* $[K_{\vec{v}}]$*.*

Remark 4. Note that an SBM according to Definition 2.4 can be seen as a L-level HSBM (for any L) by simply having the level-1 structure encode the SBM blocks, and letting the HSBM's at subsequent levels be 1-block Erdős-Rényi graphs. As such, in item iii. of the definition above each S_i could be a standard SBM (according to Definition 2.4) extended to be level $L-1$.

Remark 5. In Definition 2.6, there is an implicit tree structure T defining the hierarchy in the graph. For a L-level HSBM, the tree here is a depth-L rooted tree with K vertices at depth-1. These vertices represent the K metablocks $\{S_{(i)}\}_{i=1}^{K}$. At depth-2, there are $\sum_{i=1}^{K} K_{(i)}$ nodes, with the node representing $S_{(i)}$ at depth-1 having $K_{(i)}$ offspring in depth-2. At depth-3, there are $\sum_{i=1}^{K}\sum_{j=1}^{K_{(i)}} K_{(i,j)}$ nodes, with the node representing $S_{(i,j)}$ at depth-2 having $K_{(i,j)}$ offspring in depth-3. We then define the tree inductively to depth-L. Given this tree, we define the *traversal function* $\mathscr{T} : V \to \mathbb{Z}^L$, where $\mathscr{T}(i) = \vec{v}$ denotes that vertex i belongs to metablock $S_{\vec{v}}$ at level-L in the hierarchy. The depth $\delta(t)$ then refers to the index at which the membership is observed, i.e. $\mathscr{T}(i)_{\delta(t)}$.

Note that the HSBM is an instantiation of an SBM, although the HSBM model allows us to use fewer parameters to describe the model. For example, the number of parameters for a general SBM with $K > 4$ communities is $(K^2 - K + 2)/2$, meanwhile an *HSBM* with 2 communities each with $K/2$ communities would have $(K^2 - 2K + 12)/4$. The hope is that this reduction in complexity facilitates better estimation of the parameters.

This consideration allows us to define an RDPG based Hierarchical Block Model.

Definition 2.7 (Hierarchical Block Model Random Dot Product Graph)**.** *We say that an n-vertex random graph G is an instantiation of a L-level Hierarchical Block Random Dot Product Graph (HBRDPG) with parameters*

- $n \in \mathbb{Z}^+$*, the number of vertices in the graph;*
- $K \in \mathbb{Z}^+$*, the number of metablocks at Level-*1 *(i.e., at depth 1 in the tree);*
- F *a distribution over* Ω_d
- $x_i \overset{i.i.d.}{\sim} F$ *for* $1 < i < K$.
- $B \in [0,1]^{K \times K}$ *(a symmetric matrix), the Level-1 metablock–to–metablock communication matrix where* $B_{ij} = x_i^T x_j$.

- $\pi \in \Delta^{K-1}$ *(the* $K-1$ *simplex), the Level-1 metablock probability assignment vector;*

- $\{S_i\}_{i=1}^K$*, a collection of* $L-1$*-level HSBM's*

if $G \sim HSBM(n,K,B,\pi,\{S_i\}_{i=1}^K)$ *(we write this as* $G \sim HBRDPG(n,K,F,\pi,\{S_i\}_{i=1}^K)$*).*

We observe that any L-level HSBM model can always be written as an SBM as follows: let $G \sim \text{HSBM}(n,K,B,\pi,\{S_{(i)}\}_{i=1}^K)$. Then $G \sim \text{SBM}(n,K^*,B^*,\pi^*)$ where K^* is the total number of possible block assignments at the lowest level, more precisely $K^* = |\mathscr{L}(T)|$; for block $S_{\vec{v}}$ representing a leaf in the tree, the block membership is given by

$$\left(\pi(\vec{v}(1))\prod_{i=1}^{L-1}\pi_{\vec{v}[1:i]}(\vec{v}(i+1))\right)\pi_{\vec{v}}$$

(i.e., the product of the probabilities along the path from the root of the tree to $S_{\vec{v}}$). The edge probability between two distinct blocks $S_{\vec{v}}$ and $S_{\vec{w}}$ (so that $\vec{w} \neq \vec{v}$), representing distinct leaves in the tree, is equal to the following. Letting $i^*(\vec{v},\vec{w}) = \min\{i : v(i) \neq w(i)\}$ so that $i^*(\vec{v},\vec{w}) - 1$ is the level of $\text{LCA}(\vec{v},\vec{w})$ in the tree, we have (writing $i^* = i^*(\vec{v},\vec{w})$ for ease of notation; if $\vec{v} = \vec{w}$, then $i^* = 0$ by convention)

$$B_{\vec{v},\vec{w}} = B_{\vec{v}[1:(i^*-1)]}[\vec{v}(i^*),\vec{w}(i^*)] = B_{\vec{w}[1:(i^*-1)]}[\vec{v}(i^*),\vec{w}(i^*)], \text{ if } i^* > 0.$$

If if $\vec{v} = \vec{w}$, then the connection probabilities among vertices in $S_{\vec{v}}$ are dictated by the block probability matrix $B_{\vec{v}} = B_{\vec{w}}$.

Similar to the conditional SBM, sometimes we are interested in the behavior of the network when the community structure is known or non-random. This conditional HSBM definition is entirely analogous to Definition 2.6, except that the tree structure T and traversal function $\mathscr{T}$ are known a priori; we write

$$G \sim \text{HSBM}(n,K,B,\{S_i\}_{i=1}^K,T,\mathscr{T}).$$

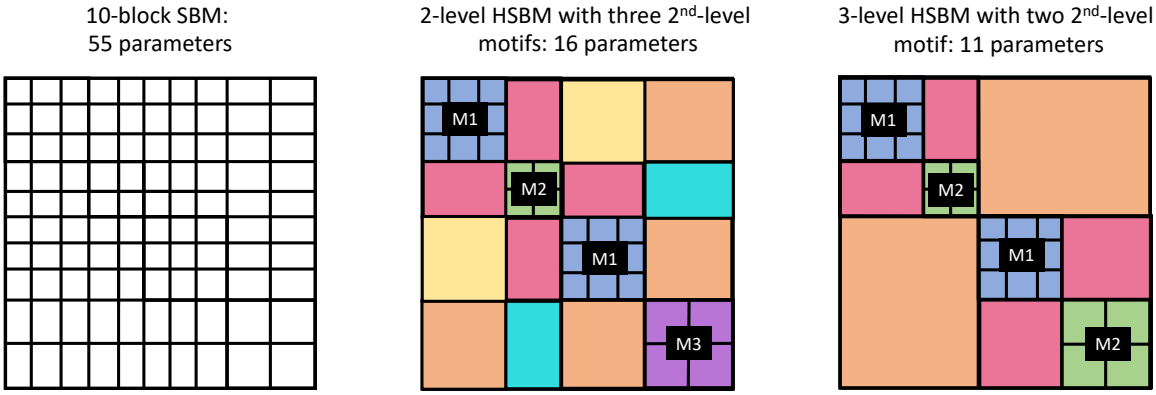


Figure 2.3: Example of refinement of block structure in the HSBM. In the center and right figures, each unique color represents a distinct parameter in the model. The middle (HSBM) model is a refinement of the right (HSBM) model, and the left (SBM) model is a refinement of the middle model. We can use the BIC formulation below to compare any pair of these three network models.

Stated simply, instead of the membership of $\mathscr{B}_{\vec{v}[1:i]}$ being randomly assigned according to probability membership vector $\pi_{\vec{v}[1:i-1]}$, these memberships are given a priori via the traversal function $\mathscr{T}$ which assigns each vertex to its path from root to node in the tree.

Definition 2.8. *(Assortativity Conditions)*

- *Weakly Assortative:* $B_{\vec{u},\vec{v}} < B_{\vec{w},\vec{v}}$ *if* $\mathrm{LCA}(\vec{w},\vec{v}) \in \mathrm{LCA} \downarrow (\vec{u},\vec{v})$
- *Assortative:* $B_{\vec{u},\vec{v}} < B_{\vec{u}_*,\vec{v}_*}$ *if* $\delta\left(\mathrm{LCA}(\vec{u},\vec{v})\right) < \delta\left(\mathrm{LCA}(\vec{u}_*,\vec{v}_*)\right)$
- *Disassortative:* $B_{\vec{u},\vec{v}} > B_{\vec{u}_*,\vec{v}_*}$ *if* $\delta\left(\mathrm{LCA}(\vec{u},\vec{v})\right) < \delta\left(\mathrm{LCA}(\vec{u}_*,\vec{v}_*)\right)$

The *T-stochastic Block Model* is another tree model introduced in [34], where the nodes of the graph correspond to the leaves of a tree T. With an attachment probability matrix based on the distance between these leaves. We use the distance function defined in 2.1 by considering the tree T as a graph. Formally,

Definition 2.9 ($\mathbb{T}$-stochastic Graph)**.** *Let G be a graph with node set V, and let T be a tree*

with leaf set L where $|L| = |V|$ with a bijection between them. For any $i, j \in V$, let

$$\lambda_{ij} = ce^{-d(i,j)}$$

for some $c > 0$. Then G is a T-stochastic Model.

Remark 6. *The role λ_{ij} plays here is a general parameter for the edge distribution. Their definition is very general and encompasses a large class of models. If we specify that the edges between the vertices $i, j \in V$ follow a Bernoulli(λ_{ij}) distribution, we arrive at an HSBM.*

Remark 7. *$\mathbb{T}$-Stochastic graphs from [34] can describe* HSBM*s, but restricted only to graphs that are Weakly Assortative, Assortative, or Disassortative with a Bernoulli edge distribution where $B_{\vec{u},\vec{v}} = ce^{-d(\vec{u},\vec{v})}$ for a positive constant c, where $d(\vec{u},\vec{v})$ is the length of the shortest path between $\vec{u}$ and $\vec{v}$ on the tree T. Using our notation, let $k = \delta(\mathrm{LCA}(\vec{u},\vec{v}))$, $L = \delta(i)$ for $i \in \mathscr{L}$ and $\ell(\vec{u})$ be the length of vector $\vec{u}$ then $d(\vec{u},\vec{v}) = \ell(\vec{u}) + \ell(\vec{v}) - 2k = 2(L-k)$. Our definition allows for models that are not weakly assortative, assortative, or disassortative.*

The network structure we are after is one where the models for the metablocks (i.e., the $S_{\vec{v}[1:i]}$) are restricted to come from a set of $\Sigma(t)$ possible models. This is captured in the following definition, Definition 2.10; though we first establish some helpful notation. Let $G \sim$SBM(n,K,B,π). We write $\mathscr{M}(G) = (K,B,\pi)$ to be the *order-independent model* associated with G. Similarly, if $G \sim$ HSBM$(n,K,B,\pi,\{S_i\}_{i=1}^K)$, we write

$$\mathscr{M}(G) = (K,B,\pi,\{\mathscr{M}(S_i)\}_{i=1}^K)$$

to be the *order-independent model* associated with G. If G_1 is an $L^{(1)}$-level HSBM, and G_2 is an $L^{(2)}$-level HSBM where, wlog, $L^{(2)} \leq L^{(2)}$), then we say that G_1 and G_2 are

equivalent models if, extending G_2 to be an $L^{(1)}$-level HSBM as in Remark 4, we have that $\mathscr{M}(G_1) = \mathscr{M}(G_2)$. Note that if $\mathscr{M}(G_1) = \mathscr{M}(G_2)$, then necessarily for all $\vec{v} \in \mathbb{Z}^{\ell}$ ($\ell \leq L^{(1)}$) we have (where $S_{\bullet}^{(1)}$ and $S_{\bullet}^{(2)}$ denote metablock models for G_1 and G_2 resp.) that $\mathscr{M}(S_{\vec{v}}^{(1)}) = \mathscr{M}(S_{\vec{v}}^{(2)})$.

Definition 2.10 (Repeated Motif HSBM)**.** *Let $G \sim \text{HSBM}(n, K, B, \pi, \{S_i\}_{i=1}^{K})$ be an L-level HSBM. We say that G is a Repeated Motif HSBM (RMHSBM) with motifs $\{\mathscr{M}_i\}_{i=1}^{M}$, at level-$\tilde{L}$ (where $\tilde{L} < L$), written $G \sim \text{RMHSBM}(n, K, B, \pi, \{S_i\}_{i=1}^{K}, \tilde{L}, \{\mathscr{M}_i\}_{i=1}^{M})$, if the following holds:*

i. For each metablock model $S_{\vec{v}}$ of G (where $\vec{v} \in \mathbb{Z}^{\tilde{L}}$), there is a j such that $\mathscr{M}(S_{\vec{v}}) = \mathscr{M}_j$;

ii. If metablock models $S_{\vec{v}}, S_{\vec{w}}$ of G, where $\vec{v}, \vec{w} \in \mathbb{Z}^{\tilde{L}}$ are such that $\vec{v}[1 : \tilde{L}-1] = \vec{w}[1 : \tilde{L}-1]$, and $\mathscr{M}(S_{\vec{v}}) = \mathscr{M}(S_{\vec{w}})$, then for all metablock models $S_{\vec{u}}$ where $\vec{u} \in \mathbb{R}^{\tilde{L}}$ are such that $\vec{v}[1 : \tilde{L}-1] = \vec{u}[1 : \tilde{L}-1]$, we have that

$$B_{\vec{v}[1:\tilde{L}-1]}[w(\tilde{L}-1), u(\tilde{L}-1)] = B_{\vec{u}[1:\tilde{L}-1]}[v(\tilde{L}-1), u(\tilde{L}-1)];$$

i.e., all metablocks with the same parent node as $S_{\vec{v}}, S_{\vec{w}}$ have the same probability of forming an edge to vertices in $S_{\vec{v}}$, and $S_{\vec{w}}$.

Note that the conditional (conditioning on the traversal structure) HSBM with repeated motifs is defined analogously, and is denoted via (note that the parameter $\mathfrak{m}$ is explained below)

$$G \sim \text{RMHSBM}(n, K, B, \{S_i\}_{i=1}^{K}, T, \mathscr{T}, \tilde{L}, \mathfrak{m}, \{\mathscr{M}_i\}_{i=1}^{M});$$

note that we will assume moving forward that this conditional structure *explicitly* identifies which motif $\mathscr{M}_i$ each metablocks at level $\tilde{L}$ is equivalent to, and this is represented by the

metablock mapping parameter:

$$\mathfrak{m} : \text{metablocks at level } \tilde{L} \mapsto \{\mathscr{M}_i\}_{i=1}^{M}.$$

One of the principle advantages of the repeated motif HSBM framework is that, given the structure of the motifs, the number of parameters for the HSBM model are greatly reduced. This is because all substructures equivalent to a given motif share the same block probability parameters and block membership parameters; these need only be estimated once per represented motif instead of separately over each substructure. See Figure 2.3 for example.

Chapter 3: Analysis Methods for Graphs

In this chapter we will discuss how our graph and random graph definitions can help us analyze graphs. We begin by describing popular tasks and problems related to graphs, then we discuss methods and how they can be used to solve them.

3.1 Tasks and Problems Related to Graphs

This section is dedicated to describing problems in graph analysis. We will define the clustering, edge prediction, graph/subgraph matching, and vertex/subgraph nomination problems, then we will explore loss functions related to these problems.

3.1.1 Clustering

Partitioning a dataset into groups can help us understand large data structures, for example, these partitions can aid in edge prediction as seen in [35] with an application on a coauthorship network or recommender systems as seen in [88] with an application in nutrition. We call inferring these groups clustering with the following formal definition in the context of graphs.

Definition 3.1 (Clustering)**.** *Given $G = (V,E)$ be a random graph on n vertices and $f : \mathscr{P}(V) \times \mathscr{P}(V) \to \mathbb{R}^+$ find $K \in \mathbb{N}$ and a partition of V with K subsets $\mathscr{C} = \{\mathscr{B}_i\}_{i=1}^K$ that minimize $F(\mathscr{C}) = \sum_{k=1}^K f(\mathscr{B}_i, \mathscr{B}_i^c)$.*

One may want to compare two partitions, in the sense of how closely two partitions $\mathscr{C}, \mathscr{C}'$ resemble each other. For this task, there are context-independent measures such as the Adjusted Random Index [49] and the Variational Index [71]. These measures help us validate the efficacy of clustering methods when using data with a known classification, as the true classification may not necessarily minimize our selected objective function F.

3.1.2 Graph and Subgraph Matching

Two graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic when $|V_1| = |V_2| = n$ and there is some permutation $\sigma \in S_n$, such that for each $f_1 \in E_1$ we can find $f_2 \in E_2$ where $f_1(i, j) = f_2(\sigma(i), \sigma(j))$ for all $i, j \in V_1$. The problem of identifying whether there is such a permutation and finding it is the graph isomorphism problem; see [9] for complexity of the isomorphism problem. However, if we are considering random distributions of graphs, we will need to relax this problem from finding an isomorphism to minimizing the disagreement between the two graphs. This is because it is highly unlikely that any two graphs are isomorphic even when drawn from the same distribution.

Definition 3.2 (The Graph Matching Problem)**.** *For the graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where $|V_1| = |V_2| = n$, $E_1 = \{f_i\}_{i \in [m]}$ and $E_2 = \{g_i\}_{i \in [m]}$, find permutations $\sigma \in S_n$, $\gamma \in S_m$ that minimize*

$$F(\sigma, \gamma) = \sum_{i \in [m]} \sum_{i,j \in V_1} |f_i(i, j) - g_{\gamma(i)}(\sigma(i), \sigma(j))|$$

Remark 8 (Graphs with one edge function)**.** *If we want to match two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where $|V_1| = |V_2| = n$, each with a single edge function f, g respectively, in other words, graphs that can be represented as a single adjacency matrix via $A_{ij} = f(i, j)$ and $B_{i,j} = g(i, j)$, we can perform graph matching by considering a search on permutation*

matrices. More formally, find the permutation matrix W that minimizes

$$F(W) = \|A - WBW^T\|_F$$

where $\|\cdot\|_F$ is the Frobenius norm.

A related problem is the subgraph isomorphism problem, where we are given a graph $G_1 = (V_1, E_1)$ and a target graph $G_2 = (V_2, E_2)$ with $|V_1| < |V_2|$. The objective in this problem is to find a subgraph $G_2(V)$, where $V \subset V_2$, which is isomorphic to G_1. The subgraph matching problem is the relaxation of the subgraph isomorphism problem for random graphs.

Definition 3.3 (The Subraph Matching Problem)**.** *For the graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where $|V_1| < |V_2|$, $|V_1| = n$, $E_1 = \{f_i\}_{i\in[m]}$ and $E_2 = \{g_i\}_{i\in[m]}$, find an injection $\sigma : V_1 \to V_2$ and a permutation $\gamma \in S_m$ such that the loss function*

$$F(\sigma, \gamma) = \sum_{i\in[m]} \sum_{i,j\in V_1} |f_i(i,j) - g_{\gamma(i)}(\sigma(i), \sigma(j))|$$

is minimized.

In [91], graphlet kernels are used to establish an efficient algorithm to retrieve subgraphs that are similar to a query graph. In the context of graph databases, where information is stored as a knowledge graph, there are different efficient algorithms with applications in the context of protein-protein interaction (PPI) networks [119], the Wikipedia database [47], and the AIDS Antiviral Screen dataset [98].

One flavor of the subgraph matching problem tasks us with finding multiple subgraphs that are isomorphic to each other. Generally, a random binary graph with one edge function and n vertices is guaranteed to have two subgraphs of size $\log(n)$ that are isomorphic,

[8], but in general these subgraphs are trivial, as in two sets of independent nodes. By induction, such a graph with two edge functions is guaranteed to have two subgraphs of size $\log(\log(n))$ that are isomorphic. We can extend this definition by requiring that the subgraphs have at least K edges, but in [11] this problem is shown to be NP-complete. Meanwhile, if we are looking for a subgraph with a specific pattern, there are efficient approximation algorithms such as Ullmann's algorithm [112] which uses the degrees of the nodes to constrain a subset of permutations which are then tested sequentially, or algorithms based on the decomposition of the search graph [73].

3.1.3 Vertex and Subgraph Nomination

In subgraph nomination, our task is as follows: given a subgraph or subgraphs of interest in a network G_1, we seek to find heretofore unknown subgraphs of interest in a second network G_2. Our approach to subgraph nomination proceeds by

1. Partitioning the vertices in G_2 into candidate subgraphs (note that the choice of non-overlapping candidate subgraphs is merely a theoretical/notational convenience, and in practice, the subgraphs to be ranked can have overlap);

2. Ordering the candidate subgraphs in G_2 into a rank list with the unknown subgraphs of interest ideally concentrating at the top of the rank list.

Subgraph nomination is a natural extension of vertex nomination (VN) [1, 29, 37, 65, 70, 106], and we will begin by providing a brief background on recent developments in VN, as this will provide useful context for our later novel formulation of subgraph nomination.

Vertex Nomination

Semi-supervised querying of large graphs and databases is a common graph inference task. For example, in a graph database with multiple features, one may be given a collection of book names all with the common *latent* feature "Horror Fiction Best Sellers," and the goal would be to query the database for more titles that fit this description; see, for example, the work in [5, 85]. Learning this unifying feature from the query set is a challenging problem unto itself [84], especially in settings where the features that define interestingness are nuanced or multi-faceted.

While we can interpret such a problem as a vertex classification problem where vertices are either labeled interesting or not, in the presence of very large networks with large class imbalance between interesting and non-interesting vertices, there is often limited training data and limited user resources for verifying returned results. In this setting, an information retrieval (IR) framework may be more appropriate, wherein unlabeled vertices would be ordered in a rank list based on how interesting they are deemed to be. This inference task is synonymous with *vertex nomination* (or personal recommender systems on graphs).

In [1, 59, 65, 80], the vertex nomination problem is defined as follows. Given vertices of interest $V^* \subset V(G_1)$ in a network G_1, and corresponding unknown vertices of interest $U^* \subset V(G_2)$ in a second network G_2 (whose identities are hidden to the user), use G_1, G_2, V^* to rank the vertices in G_2 into a nomination list, with vertices in U^* concentrating at the top of the nomination list. In its initial formulation [29, 37, 70, 118], the feature that defined vertices as interesting was membership in a community of interest, and the community memberships of vertices in V^* were used to nominate the vertices in G_1 (no second network was introduced, or $G_2 = G_1 \setminus \{V^*\}$) with unknown community memberships. Subsequent work [65, 80, 84, 85] sought to generalize the features that defined vertices as interesting beyond simple community membership, and lifted the vertex nomination problem to the

two graph setting.

In order to allow for a broadly general class of networks to be considered, the general vertex nomination problem framework of [1, 65] referenced above is situated in the context of nominatable distributions, a broad class of random graph distributions defined in [1]. Within this broad class of models, the concepts of Bayes optimality and consistency were developed in [1, 65] and the important result that universally consistent VN schemes do not exist is proven in [65]. These results were leveraged in [1] to develop an probabilistic adversarial contamination model in the context of VN as well as regularization schemes to counter the adversary. The results of [1, 65] were further extended to the richly featured network setting in [59], in which the (potentially) complementary roles of features and network structure are explored in the VN task.

Beyond the development of novel VN algorithms [118], one of the key aspects of the recent theoretical developments in VN is the notion that vertex labels in g_2 are uninformative in the ranking scheme, which is sensible if we are mirroring setting in which the vertex labels do not aid in the delineation between interesting and uninteresting vertices. In subgraph nomination, the uninformative nature of the labels can be accounted for in the dissimilarity Δ (see Eq. 3.1). Note that while similar consistency results to those in VN can be derived in the setting of subgraph nomination, we do not focus on that here. Rather we will focus on the role of users-in-the-loop in the subgraph nomination framework.

Remark 9. *We note that it is possible to view the subgraph nomination methods as naturally being built upon existing VN methods via uncovering vertices of interest (from VN) and building out from them to uncover subgraph structure of interest. One immediate hurdle to this approach would be if the subgraph structure is the interesting trait rather than the individual vertex identities, as then the nominated vertices in VN would need to look at local structure to be correctly retrieved, and essentially subgraphs would end up being the*

nominated structures. It is also natural to use subgraph nomination for VN, by nominating structures first and then vertices within the structures.

Hierarchical Subgraph Models

We will use $\mathscr{G}_n$ to denote the set of n-vertex labeled graphs on a common vertex set $V = V_n$. Given the (informal) explanation of the motivating task in subgraph nomination—given light supervision, partition a graph into subgraphs and rank the subgraphs based on how interesting they are judged to be—hierarchical network models are a natural setting for initially formalizing the subgraph nomination inference task. We note here that the subgraph nomination inference task can be formulated in the context of any graph division mechanism that yields partition subgraph structure (e.g., graph cuts [31, 55], graph clustering methods [17, 77, 109], etc.); the Hierarchical blockmodel considered below is one such setting, and it also provides a convenient model setting in which to formulate subgraph nomination formally.

Hierarchical models have seen a surge in popularity in the network literature (see, for example, [27, 60, 66, 79, 82, 89]) and naturally allow for multiple layers of structure to be simultaneously modeled in a network, allowing us to imbue a graph G with subgraph structures of a desired form or motif-type. We begin with our definition of a hierarchical network as a network together with a *Network Hierarchical Function*.

Definition 3.4. *Let $g = (V,E) \in \mathscr{G}_n$, and for each $i \in \mathbb{Z} > 0$ let $[i] = \{1,2,\cdots,i\}$. Let $k \leq n$. A function*

$$H = H_k : V \times [k] \longrightarrow [n]$$

is a k-level Network Hierarchical Function of [n] *if*

i. For each $i \in [k]$, $H(\cdot,i)$ represents a partition of the vertices of V into $n_i := n_i^{(H)}$ nonempty parts (so that $\max_v H(v,i) = n_i$). These n_i satisfy $1 \leq n_1 \leq n_2 \leq \cdots \leq n_k \leq$

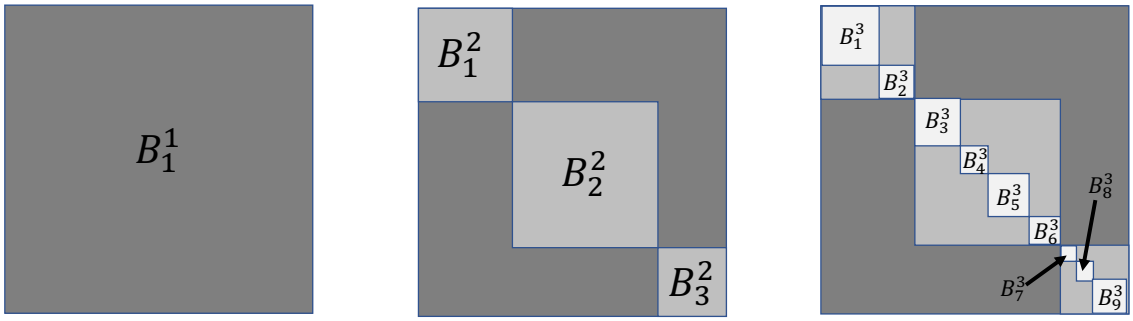


Figure 3.1: An example of a 3-Level hierarchical network (where the three levels are shown in the right panel, the top two levels in the middle panel, and the highest level in the left panel). The partition provided by $H(\cdot,1)$ is in darkest grey; the partition provided by $H(\cdot,2)$ is in medium grey; and the partition provided by $H(\cdot,3)$ is in lightest grey.

n. The signature *of H is defined to be the vector $\vec{n} = (n_1, n_2, \cdots, n_k)$. Note that by considering appending 0's onto the end of $\vec{n}$ as needed to make it length n, we can consider the signature of an n vertex hierarchical graph to be a length n vector.*

ii. $H(v_1, j) = H(v_2, j)$ *only if* $H(v_1, i) = H(v_2, i)$ *for all* $i \leq j$*; i.e., the partition is nested.*

For an example of network hierarchical functions, see Figure 3.1. A graph $g \in \mathscr{G}_n$ together with a Network Hierarchical Function H_k of $[n]$ is a k-level Hierarchical Graph. *For $k \leq n$, we denote*

$$\mathscr{H}_{k,n} = \{H_k \,|\, H_k \text{ is a k-level Network Hierarchical Function of } [n]\},$$

and we define

$$\mathscr{HG}_n := \{(g,H) \,|\, g \in \mathscr{G}_n, \text{ and } H \in \cup_{k=1}^{n} \mathscr{H}_{k,n}\}$$

to be the set of all Hierarchical graphs of order n.

Letting $k \leq n$, let $H \in \mathscr{H}_{k,n}$. For each $i \in [k]$ and $j \in [n_i]$, we define the sets

$$B^i_j = \{v \in V(g) | H(v,i) = j\}$$

to be the set of vertices in the j-th part of the i-th level of the hierarchy;

$$B^{i-1}_{(j)} = \{v \in V(G) \text{ s.t. for all } v' \in B^i_j, H(v,i-1) = H(v',i-1)\}$$

to be the one-step upward merge of B^i_j in the hierarchy;

$$B^i = \{B^i_j \,|\, 1 < j < n_i\}$$

to be the set of all parts at level i in the hierarchy; and

$$B = \{B^i : i \in [k]\}.$$

to be the set of blocks.

As an example, consider the hierarchical network in Figure 3.1, which shows a 3-level hierarchical network, where we have (for example):

$$\begin{aligned}
n_1 = 1; n_2 = 3; n_3 = 9;\\
B^1_1 = V = B^1_2 \cup B^2_2 \cup B^3_2;\\
B^2_{(3)} = B^3_3 \cup B^3_4 \cup B^3_5 \cup B^3_6;\\
B^3 = \{B^3_1, B^3_2, B^3_3, B^3_4, B^3_5, B^3_6, B^3_7, B^3_8, B^3_9\};\\
B = \{B^1_1, B^2_1, B^2_2, B^2_3, B^3_1, B^3_2,\\
B^3_3, B^3_4, B^3_5, B^3_6, B^3_7, B^3_8, B^3_9\}.
\end{aligned}$$

Hierarchical Subgraph Nomination

A key component to the hierarchical subgraph nomination (HSN) inference task is the notion of subgraphs of interest within and across the network pair. This is easily accomplished in non-random settings, where a fixed set of indices can be used to define the subgraphs of interest within and across networks. In order to expedite this in random networks, for a fixed pair of signatures $\vec{n}$ and $\vec{m}$, we will consider distributions restricted to $\mathscr{HG}_n^{\vec{n}} \times \mathscr{HG}_m^{\vec{m}}$; this will allow us to define a consistent set of indices for subgraphs of interest across the random network pair.

In hierarchical subgraph nomination, we consider a pair of hierarchical graphs $(g_1, H_1) \in \mathscr{HG}_n^{\vec{n}}$ and $(g_2, H_2) \in \mathscr{HG}_m^{\vec{m}}$ where H_1 and H_2 are unobserved and only the graphs g_1 and g_2 are observed. We are additionally given a set of "training" subgraphs of interest in g_1, denoted

$$T_1 = \{B^{s_1}_{s_2,1}\}_{s=(s_1,s_2)\in S_1}$$

where S_1 is a set of ordered pair indices $S_1 = \{s = (s_1, s_2)\}$ of the given subgraphs of interest, with s_1 denoting the level and s_2 the index of the subgraph in H_1. Note that the second index in the subscript of $B^{\bullet}_{\bullet,1}$ is used to denote the subgraph in g_1, whereas $B^{\bullet}_{\bullet,2}$ will be used to indicate subgraphs in the hierarchy of g_2. Note that the possible indices of the seed graphs depends on the structure of H_1, and so we will explicitly tether these two in the sequel, writing $((g_1, H_1), S_1)$, and writing $\mathscr{HGS}_n^{\vec{n}}$ for the collection of all such feasible triples.

The aim then is to

1. First estimate the latent hierarchical structure H_2 of g_2; we will denote this estimate

via $\widehat{H}_2$, and we will let

$$\widehat{B}^i_{j,2} = \{v \in V(g_2) | \widehat{H}_2(v,i) = j\}.$$

Let the collection of all subgraphs in the partition defined by $\widehat{H}_2$ be denoted $\widehat{B}$.

2. Compute dissimilarity measurements

$$\Delta : T_1 \times \widehat{B} \mapsto [0,1]$$

between each subgraph of interest in (g_1, H_1) and each subgraph defined by $\widehat{H}_2$. Note that Δ ideally is a function that can compute the dissimilarity between any graph of order up to n and any graph of order up to m; and the larger the value of Δ between two graphs, the more dissimilar the networks. As Δ needs to compute dissimilarities across graphs of different sizes and orders, it may encompass a collection of dissimilarity measures.

After a few preliminary definitions, we will be ready to define a HSN scheme.

Definition 3.5. *For a k-level hierarchical graph (g,H) and $i \in [k]$, let $\mathscr{T}^H_{g,i}$ denote the set of all total orderings of B^i. Let $\mathscr{T}^H_g = \bigotimes_{i=1}^k \mathscr{T}^H_{g,i}$.*

For example, in the hierarchical subgraph depicted in Figure 3.1, we have

$$\mathscr{T}^H_{g,2} = \left\{ \begin{bmatrix} B^2_1 \\ B^2_2 \\ B^2_3 \end{bmatrix}, \begin{bmatrix} B^2_1 \\ B^2_3 \\ B^2_2 \end{bmatrix}, \begin{bmatrix} B^2_2 \\ B^2_1 \\ B^2_3 \end{bmatrix}, \begin{bmatrix} B^2_2 \\ B^2_3 \\ B^2_1 \end{bmatrix}, \begin{bmatrix} B^2_3 \\ B^2_1 \\ B^2_2 \end{bmatrix}, \begin{bmatrix} B^2_3 \\ B^2_2 \\ B^2_1 \end{bmatrix} \right\}$$

where

$$\begin{bmatrix} B_i^2 \\ B_j^2 \\ B_k^2 \end{bmatrix}$$

is shorthand for the ordering

$$B_i^2 > B_j^2 > B_k^2.$$

An example element of $\mathscr{T}_g^H$ is given by

$$\left([B_1^1], [B_3^2, B_2^2, B_1^2], [B_2^3, B_9^3, B_8^3, B_1^3, B_3^3, B_6^3, B_5^3, B_7^3, B_4^3]\right).$$

As in the case of vertex nomination, if vertex labels in (g_2, H_2) are uninformative, then an HSN must account for the indistinguishability of subgraphs with isomorphic structure. To wit, we have the following definition: For a k-level hierarchical network (g, H) and $i \in [k]$, $j \in [n_i]$, we define

$$\mathfrak{I}(B_j^i; g) = \{\ell \in [n_i] \,|\, B_\ell^i \subset B_{(j)}^{i-1} \text{ and with an automorphism } \sigma \text{ of } g \text{ with } \sigma(B_\ell^i) = B_j^i\}.$$

These are indices of the elements of the partition at level i that are indistinguishable from B_j^i without further information/supervision, and it is sensible to require that a HSN rank all these subgraphs as equally interesting. In vertex nomination, accounting for label uncertainty was achieved by means of an obfuscation function; in HSN, the uncertainty is not at the level of labels so much as at the level of subgraphs and this this can be achieved by further requiring the dissimilarity Δ satisfies the following condition:

$$\begin{aligned} &\Delta(B_{s_2,1}^{s_1}, \cdot) \text{ is constant over graphs indexed by} \\ &\mathfrak{I}(\widehat{B}_{j,2}^i; g_2) \text{ for each } s = (s_1, s_2) \in S_1. \end{aligned} \tag{3.1}$$

We are now ready to define hierarchical subgraph nomination schemes formally.

Definition 3.6. [*Hierarchical Subgraph Nomination Scheme (HSN)] Let* $((g_1,H_1),S_1) \in \mathscr{HGS}_n^{\vec{n}}$ *and* $((g_2,H_2),S_2) \in \mathscr{HGS}_m^{\vec{m}}$ *and let* T_1 *be the parts of* H_1 *indexed by* S_1. *A* Hierarchical subgraph nomination scheme *is composed of two parts:*

i. An estimator $\widehat{H}_2 = \widehat{H}_2(g_1,g_2,T_1)$ *of the hierarchical clustering of* g_2 *provided by* H_2*; let the signature of* $\widehat{H}_2$ *be denoted* $\hat{m}_2 = (\hat{m}_{k,2})$*;*

ii. A dissimilarity Δ *satisfying Eq. (3.1) which produces a ranking scheme* $\Phi_{n,m}$*, where*

$$\Phi_{n,m}\left(g_1,g_2,T_1,\widehat{H}_2\right) \in \mathscr{T}_{g_2}^{\widehat{H}_2}.$$

For each level i of $\widehat{H}_2$*, the subgraphs are ordered in* $\Phi_{n,m}$ *via increasing value of*

$$\min_{s=(s_1,s_2)\in S_1} \Delta(B^{s_1}_{s_2,1}, \widehat{B}^{i}_{j,2}),$$

with ties broken in a fixed but arbitrarily manner.

Loss in HSN Schemes

The goal of HSN is to effectively query g_2 given limited training resources from g_1. Given the resources required for a user to verify the interestingness of the returned subgraphs, we seek to maximize the probability that a priori unknown subgraphs of interest in g_2 are close to the top of the returned rank list.

For evaluation purposes, it is necessary (at least in theory) to be able to compare the true-but-unknown subgraphs of interest in $((g_2,H_2),S_2) \in \mathscr{HGS}_m$ with elements of the HSN ranked list. In order to account for the fact that the dissimilarity used to construct the HSN scheme may be mis-specified (i.e., not captured exactly), the dissimilarity for

verification will be denoted Δ_E (the dissimilarity used for evaluation that defines the true but unknown subgraphs of interest), though we do not discount the possibility that $\Delta_E = \Delta$ for any given scheme.

A logical loss function for HSN is motivated by the concept of precision in information retrieval. As the goal of HSN is to efficiently query large networks for structures of interest, at a given level k, considering precision at i for $1 < i \ll \hat{m}_{k,2}$ enables us to model the practical loss associated with using a HSN scheme to search for $\{B^{s_1}_{s_2,2}\}_{(s_1,s_2)\in S_2}$ in (g_2, H_2) given limited resources. There are two levels to the loss function for a given scheme $\Phi_{n,m} = (\widehat{H}_2, \Delta)$:

i. The error in approximating H_2 via $\widehat{H}_2$;

ii. Given $\widehat{H}_2$, the potential mismatch between Δ and Δ_E.

Our loss function will account for these error sources as follows. Let F be a distribution supported on $\mathscr{HGS}_n \times \mathscr{HGS}_m$, and let

$$((G_1, H_1), (G_2, H_2)) \sim F.$$

Consider fixed subgraph of interest index sets S_1 for G_1 and S_2 for G_2, and let T_1 (resp., T_2) be those subgraphs in (G_1, H_1) (resp., (G_2, H_2)) indexed by S_1 (resp., S_2). Let $\Phi_{n,m} = (\widehat{H}_2, \Delta)$ be an HSN scheme and let the ranking provided at level k of $\Phi_{n,m}$ be denoted via $\Phi^k_{n,m}$ (with the implicit assumption that this is an empty list if $\hat{m}_{k,2} = 0$).

Definition 3.7 (HSN loss function, level-(i,k) error at hierarchical level)**.** *With setup as above, for $i,k < n$ we define the* level-(i,k) nomination loss with threshold $t > 0$ under dissimilarity Δ_E *via (where to ease notation, we will use $\Phi^k_{n,m}$ for $\Phi^k_{n,m}(g_1, g_2, T_1, \widehat{H}_2)$ and*

$\Phi^k_{n,m}[j]$ *the j-th ranked subgraph in this list)*

$$\ell_{i,k,t} := \ell_{i,k,t}(\Phi_{n,m}, g_1, H_1, S_1, g_2, H_2, S_2, \Delta_E)$$

where if we let $J_{i,k} = \min(i, \hat{m}_{k,2})$ *and* $\mathfrak{W}_{\Phi_{n,m}[j]} = \left\{ \bigcap_{(s_1,s_2)\in S_2} \{\Delta_E(\Phi_{n,m}[j], B^{s_1}_{s_2,2}) > t\} \right\}$

$$\ell_{i,k,t} := \begin{cases} \frac{1}{J_{i,k}} \sum_{j=1}^{J_{i,k}} \mathbb{1}\left\{ \mathfrak{W}_{\Phi_{n,m}[j]} \right\} & \text{if } \hat{m}_{k,2} > 0 \\ 1 & \text{if } \hat{m}_{k,2} = 0 \end{cases} \tag{3.2}$$

For distribution F as above, the level-(i,k) error with threshold $t > 0$ under dissimilarity Δ_E *of* $\Phi_{n,m}$ *for recovering* S_2 *is defined to be* $L_{i,k,t} = \mathbb{E}_F(\ell_{i,k,t})$. *Letting* $\mathfrak{H}_{n,m}$ *be the collection of all HSN schemes (i.e., the set of all ordered pairs of estimators and dissimilarities* $(\widehat{H}_2, \Delta)$*), the Bayes optimal level-*(i,k) *scheme with threshold* $t > 0$ *under dissimilarity* Δ_E *is defined to be any scheme that achieves the Bayes' error, which here is defined to be*

$$\min_{\Phi_{n,m}\in\mathfrak{H}_{n,m}} \mathbb{E}_F(\ell_{i,k,t}(\Phi_{n,m}))$$

While the number of possible dissimilarities is indeed uncountably infinite, there are nonetheless a finite number of possible rankings that can be achieved via a combination of $\widehat{H}_2$ and Δ (hence min rather than inf in the Bayes error definition), and so the Bayes error is indeed achieved by at least one such pair.

3.2 Spectral Methods

Representing a graph as a matrix, or tensor in the case of a hypergraph, raises a curiosity as to what information we can get about the graph by considering the *Spectral Vector Decomposition* of this representation. Since these matrices are not defined as linear oper-

ators on a vector space, it may come as a surprise that any information is encoded in their spectral decomposition. However, [78] showed that for symmetric graphs, the eigen decomposition of the adjacency matrix A approaches that of $E(A) = P$ with high probability. Recall the definition 2.2 for the RDPG, such a graph of size n with latent positions $\{x_i\}_{i=1}^n$ the matrix $P = X^T X$. Combining these ideas, [7] show that up to a rotation, the largest d eigenvalues of A and their associated eigenvectors approximate latent vectors with high probability. The rotation here is due to the nature of the inner products of the latent vectors being a function of distances and angles between them which are both invariant under rotations. This motivates our first tool for analyzing graphs, *Adjacency Spectral Embedding*, often shortened to *ASE*.

Definition 3.8 (Adjacency Spectral Embedding)**.** *For an undirected graph $G = (V, E)$ with adjacency matrix A whose spectral decomposition is $U_A S_A U_A^T$ with singular values in decreasing order, the adjacency spectral embedding of dimension d is given as follows*

$$ASE(A_G, d) = U_{A,d} S_{A,d}^{\frac{1}{2}}$$

Where $U_{A,d}$ is the first d columns of U_A, $S_{A,d} = S_A[1:d, 1:d]$ and for a diagonal matrix M, $M^{\frac{1}{2}}[i,i] = M_{i,i}^{\frac{1}{2}}$ and 0 otherwise.

Algorithm 1 Adjacency Spectral Embedding

Input: $d > 0$, $A \in \{0,1\}^{n \times n}$ for a symmetric graph $G(V, E)$ on $n > d$ vertices.
Output: $\hat{X} \in \mathbb{R}^{n \times d}$

1: Find the spectral decomposition of A to retrieve SVD the sorted pairs $\{(s_i, u_i)\}_{i=1}^n$.
2: Let $S^{\frac{1}{2}} \in \mathbb{R}^{d \times d}$ with $S^{\frac{1}{2}}[i,i] = \sqrt{s_i}$ and $U \in \mathbb{R}^{n \times d}$ where $U_{\cdot,i} = u_i$ for $1 \leq i \leq d$
3: Return $\hat{X} = U S^{\frac{1}{2}}$

The ASE estimates parameters for the distribution of the self-edges even when the adjacency matrix A is hollow [7]. This is because the RDPG formulation does not exclude

self-edges, they can be drawn in the same way the rest of the edges are drawn. In the case where we want to assume that there are no edges, we can simply ignore them since they have no interpretation.

Another consideration to take into account here is estimating the parameter d, in other words how do we select the dimension of the embedding space. To start, consider $\hat{X}$ as an estimator $\hat{P} = \hat{X}\hat{X}^T$ for the probability attachment matrix P, in keeping with the notation of algorithm 3.2 we write $\hat{P}_{i,j} = \sum_{k=1}^{d} s_k U_{i,k} U_{j,k}$. Then this question becomes one of how much tolerance in the estimate of $P_{i,j}$ we want to allow. One way to make this decision is to consider the variance of the parameter $P_{i,j}$, $Var(x_i^T x_j)$, according to the distribution F of the latent space. Another is to consider penalized model selection, based on the likelihood of the distribution of the latent vector (see Section 3.1.1 on Gaussian Mixture Models), or the adjacency matrix A , with a penalty scaling with d [23, 121]. We could also consider heuristic methods such as the elbow method, for example $d = \text{argmax}_{j>1}\{s_{\sigma(j)} - s_{\sigma(j-1)}\}$. In Section 3.1.1, we will see that the *ASE* can applied in the context of the SBM and the HSBM as part of a clustering algorithm.

In certain settings, using spectral embeddings with augmented adjacency matrices can show favorable performance, for example, [70] replaces the diagonal of a hallow adjacency matrix with $A_{i,i} = \frac{degree(i)}{n+1}$ and [94] iterates $\hat{X}_i = ASE(A + I \circ \hat{X}_{i-1}\hat{X}_{i-1}^T)$ until we reach a fixed point ($\hat{X}_i = \hat{X}_{i-1}$), in practice they stopped when the difference was small enough. The most notable augmented adjacency matrix is the normalized graph Laplacian $\mathscr{L}(A)$,

$$\begin{aligned} \mathscr{L}(A) =& I - DAD \\ D_{i,j} =& \begin{cases} 0, \ i \neq j \\ \frac{1}{\sqrt{degree(i)}}, \ i = j \end{cases} \end{aligned}$$

with its spectral embedding dubbed the *Laplacian Spectral Embedding* (*LSE*), noting that

the Laplacian of an undirected graph is positive semi-definite [108]. The spectral decomposition of the graph Laplacian is also used to define the graph Fourier transform [99]. Suppose that we have a function $f : V \to \mathbb{R}$, which we previously defined as a node feature but in this context is usually referred to as a signal on the graph. We define the Fourier transform of the graph of the signal by $\mathfrak{F}[f] = \sum_{i=1}^{|V|} f(i)u_i$ where u_i are the eigen vectors of the Laplacian graph, with its inverse $\mathfrak{F}^{-1}[f] = \sum_{i=1}^{|V|} f(i)u_i$ where u_i are the columns.

We are now ready to see the utility of *ASE* and *LSE* in the context of the SBM and the HSBM as part of a clustering algorithm. The assortative SBMs 7 are natural models for classification, since finding the partition defined by their blocks minimizes the objective function F once we identify that they are a good fit for modeling a graph. Before we describe how the blocks can be retrieved, we first start by describing Gaussian Mixture Models [87] as they will be instrumental.

Definition 3.9 (Gaussian Mixture Models (GMM))**.** *Let $n, d, K \in \mathbb{N}$, suppose that we have $\{\mathcal{N}(\mu_i, \Sigma_i)\}_{i=1}^K$ a family of multivariate Gaussian distributions with means $\{\mu_i\}_{i=1}^K \subset \mathbb{R}^d$ and variance covariance matrices $\{\Sigma_i\}_{i=1}^K \subset \mathbb{R}^{d\times d}$ with probability mass functions $\{f_i\}_{i=1}^K$ and let $\pi \in [0,1]^K$ where $\sum_i \pi_i = 1$. A matrix X with rows $\{x_i\}_{i=1}^n$ where $x_i \overset{i.i.d}{\sim} f$ for*

$$f(x) = \sum_{i=1}^{K} f_i(x)\pi_i$$

is said to be a Gaussian Mixture Model, we write this as $X \sim GMM(n, \pi, \{\mu_i\}, \{\Sigma_i\})$.

Remark 10. *This distribution could be conditioned on an assignment vector $\tau \in [K]^n$, in which case each $x_i \sim f_{\tau_i}$ independently.*

Motivated by the CLT for the ASE [7], the following algorithm described in [115] uses *ASE* from Definition 3.8 and *GMM* to retrieve clusters for a *BMRDPG* from Definition 2.5.

Algorithm 2 GMM-ASE Clustering

Input: $d > 0, A \in \{0,1\}^{n \times n}$ for a symmetric graph $G(V,E)$ on $n > d$ vertices.
Output: A function $\tau : [n] \to [K]$

1: Use the *ASE* from 3.2 to retrieve and embedding $\hat{X} \in \mathbb{R}^{n \times d}$.
2: For each integer $1 < K < \sqrt{n}$ fit the latent position matrix $\hat{X}$ to GMM, and accept the best fit.
3: Return the vector τ defined by $\tau_i = \text{argmin}_{k \in [K]} f_i(\hat{X}_{i,\cdot})$ as the assignment function.

Remark 11 (Alternatives to *GMM*)**.** *Similar to this algorithm are* Kmean-ASE *[105] where this is extended to directed graphs with an application on the Wikipedia graph and* Kmedian-ASE *[57], in fact, we may use any clustering algorithm for points in* $\mathbb{R}^d$ *such as K-plane clustering [19]. This is done by replacing step two of the algorithm where we apply GMM, applying a different method instead, and step three with an appropriate equivalent. In the case of* Kmean-ASE*, for example, we apply the K-means algorithm in Step 2 to retrieve the centers* $\{\mu_k\}$ *and use* $\tau_i = argmin_{k \in [K]} |\mu_k - \hat{X}_i|$ *for Step 3. This choice is a modeling assumption on the distribution of the latent variables F, above I described* GMM *for the* GMM-ASE *algorithm. In practice, with an unknown distribution, we would test whether* GMM *is a good fit for the data at hand. In the case of* GMM*, use the R package* mclust *[97] which includes a convenient function that fits multiple sub-models with different numbers of clusters k and selects the best fit, which allows for an easy comparison of the results between different embedding dimensions, or Python's* scilearn-kit *[81] for which you can get similar functionality with a little more work.*

This can be extended to assortative (Remark 7) *HBMRDPG* (Definition 2.7), but first we must consider what the distribution of the latent positions F looks like for an assortative *HBMRDPG* if its latent positions were drawn from a *GMM*. Since $x_i x_j^T = \|x_i\| \|x_j\| \sin \theta_{ij}$, assortativity implies that the angle between two latent positions is smaller the further down their hierarchy their assignment diverges. [66] leverage this property in their algorithm to detect the hierarchical structure of the community based on *ASE* and the *GMM* modeling

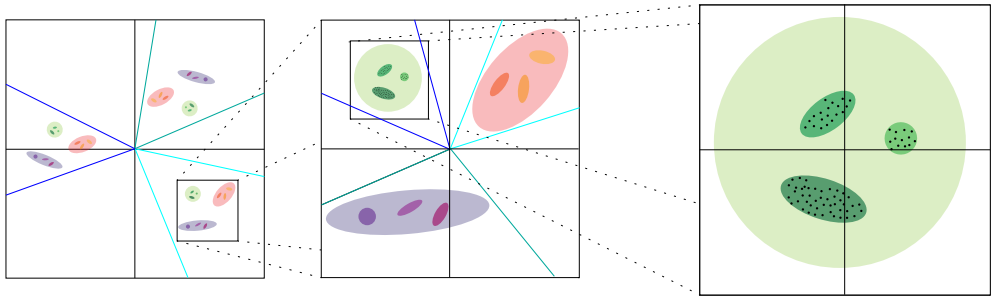


Figure 3.2: An example illustration of the HSBMClust algorithm from [66]

assumption on the distribution of the latent positions. Figure 3.2 shows an example of an assortative *HBMRDPG* with an illustration of the algorithm.

3.3 Graph Neural Networks

Thus far our exploration of analysis methods have relied on simplified graphs, this treatment has allowed for strong theoretical guarantees. Although such treatments can be generalized, we may want a more adaptable approach to analyze graph data. Neural networks allow for such a treatment that can handle all manner of data types, in this section we will discuss how we can incorporate neural networks in the setting of graph data analysis.

We will return to our definition of generalized graphs (Definition 2.1) since we can now incorporate all manner of features. In order to do so, we will need to consider all feature functions to be real valued, that is

$$\mathscr{F}_V = \{f | f : V \to \mathbb{R}_f^d\}$$
$$\mathscr{F}_E = \{g_f | g_f : V^{k_f} \to \mathbb{R}_{g_f}^d,\ f \in E\}$$

Remark 3.3.1 (Encoders)**.** *Depending on the original form of the data, we consider some encoding into* $\mathbb{R}^n$*. For example, [10] encodes node features from a finite set S as integers* $1,\ldots,|S|$*, while [120] uses a one-hot encoding, and for text features [117] used transformers. In these cases the embedding weights were trained alongside the rest of the graph neural network. Meanwhile, [69] described an encoding method for video data but used pre-trained models. We call this step* Encoding *and these functions* Encoders*, in general, Encoders are functions that embed data into* $\mathbb{R}^n$*, or other useful representations.*

In fact, the primary mechanism by which graph neural networks operate imbues the graph with *latent* features. Therefore, we need to define a node latent feature function $f_V^{(l)}$, a set of edge latent feature functions $\mathscr{F}_E^{(l)}$, and a graph latent feature vector by $f_G^{(l)} \in \mathbb{R}^{d_G}$, where

$$f_V^{(l)} : V \to \mathbb{R}^{d_V} \qquad \mathscr{F}_E^{(l)} : \{g_f^{(l)} | f \in E, g_f^{(l)} : V^{k_f} \to \mathbb{R}^{d_{E,f}}\}$$

We call $d_V, \{d_{E,f}\}, d_G$ the embedding dimensions. For a graph $G = (V,E,\mathscr{F}_V,\mathscr{F}_E)$, we can study graph neural networks by considering their output as a modified graph $G^* = (V,E,\mathscr{F}_V^*,\mathscr{F}_E^*,f_G^{(l)})$ where

$$\mathscr{F}_V^* = \mathscr{F}_V \cup \{f_V^{(l)}\} \qquad \mathscr{F}_E^* = \mathscr{F}_E \cup \mathscr{F}_E^{(l)}$$

The primary characteristic of graph data is its ability to describe relationships between elements; as such, graph neural networks make use of this information by using communication functions h_v for $v \in V$. We will first need to introduce the hyper-parameter ℓ, which controls the neighborhood communication distance and can be tuned to the specific application. Then we will construct a subgraph of G^* denoted $G^*(v) = (\mathscr{N}_v, E_v, \mathscr{F}_V^*, \mathscr{F}_E^*)$, where $\mathscr{N}_v = \bigcup_{f \in E} \mathscr{N}_{f,in}(v)$ and $E_v = \{R_v(f) | f \in E\}$ where

$$R_v(f) = f\mathbb{1}_{\mathscr{E}_v}$$

$$\mathscr{E}_v = \{e \in \mathscr{E} | e \text{ is an in-edge for } v\}$$

Then we define the operator

$$G^*(v) \oplus G^*(v') = (\mathscr{N}_v \cup \mathscr{N}_{v'}, E_{\{v,v'\}}, \mathscr{F}_V^*, \mathscr{F}_E^*)$$
$$E_{\{v,v'\}} = \{R_{\{v,v'\}}(f) = f\mathbb{1}_{\mathscr{E}_v \cup \mathscr{E}_{v'}} | f \in E\}$$

Then we define the subgraph

$$G^*(v,\ell) = G^*(v) \bigoplus_{v' \in \mathscr{N}_{v,\ell-1}} G^*(v')$$
$$\mathscr{N}_{v,\ell} = \bigcup_{f \in E} \mathscr{N}_{f,in}(v,\ell)$$

In other words $G^*(v,\ell)$ is a subgraph of the ℓ-neighborhood of v restricted to edges that form paths to v. Finally, to construct the communication function, we will need the following sets,

- $F_v = \{f(v) | f \in \mathscr{F}_V\}$ the set of features on the node.
- $F_N = \bigcup_{i \in \mathscr{N}_{v,\ell}} \{f(i) | f \in \mathscr{F}_V\}$ the set of features on nodes in the ℓ-neighborhood of v.
- $F_N^{(l)} = \bigcup_{i \in \mathscr{N}_{v,\ell}} \{f_V^{(l)}(i)\}$ the sets of latent features of neighbors of node v.
- $\mathscr{E}_{v,f} = \{e \in \mathscr{E}_f | f(e) \neq 0\}\}$ for $f \in E_{\mathscr{N}_{v,\ell-1}}$, the set of active in-edges.
- $F_E = \bigcup_{f \in E_v} \{g_f(e)\}_{e \in \mathscr{E}_{v,f}}$ the sets of edge features of in-edges for the vertex v.
- $F_E^{(l)} = \bigcup_{f \in E_v} \{g_f^{(l)}(e)\}_{e \in \mathscr{E}_{v,f}}$ the sets of edge features of in-edges for the vertex v.

then

$$h_v(F_v, F_N, F_N^{(l)}, F_E, F_E^{(l)}) \in \mathbb{R}^{d_V}.$$

As may be apparent now, this function will be used to calculate the latent features of node v, that is, $f_V^{(l)}(v) = h_v(F_v, F_N, F_N^{(l)}, F_E, F_E^{(l)})$. Since h_v depends on the latent features of neighbors of v, we could arrive at an infinite recursion, notably, this always occurs in undirected graphs. To address this, we apply this function to the nodes in a random order using the previously calculated latent features with an initial feature that depends on the type of the graph neural network. The next function we need to define is what we use to update the latent vectors on edges $h_{f,e}$. For an edge function $f \in E$ and an edge $e \in \mathscr{E}_f$, let

- $F_e = g_f(e)$ the edge feature.
- $F_V = \bigcup_{i \in e} \{f(i) | f \in \mathscr{F}_V\}$ the features on the nodes forming this edge.
- $F_V^{(l)} = \{f_V^{(l)}(i) | i \in e\}$ the latent features on the nodes forming this edge.

then we update the latent feature on the node according to $g_f^{(l)}(e) = h_{f,e}(F_e, F_V, F_V^{(l)})$. We also define the function h_G which we use to update the latent feature for the graph, let

- $F_V = \bigcup_{f \in \mathscr{F}_V} \{f(v) | v \in V\}$ the set of node features
- $F_V^{(l)} = \{f_V^{(l)}(v) | v \in V\}$ the set of latent node features
- $F_E = \bigcup_{f \in E} \{g_f(e) | e \in \mathscr{E}_f \text{ and } f(e) \neq 0\}$ the set of edge features on active edges.
- $F_E^{(l)} = \bigcup_{f \in E} \{g_f^{(l)}(e) | e \in \mathscr{E}_f \text{ and } f(e) \neq 0\}$ the set of latent edge features on active edges.

then we update the latent feature on the graph according to $f_G^{(l)} = h_G(F_V, F_V^{(l)}, F_E, F_E^{(l)})$. Each set of these functions $H^{(t)} = \{h_v^{(t)}\} \cup \{h_{f,e}^{(t)}\} \cup \{h_G^{(t)}\}$ constitutes a layer, and the architecture of a GNN is the set $H = \{H^{(t)}\}$ together with the encoders and the objective function.

The final crucial component of a neural network architecture is the loss function. We need an objective function to optimize; the objective function defines what the GNN is trying to achieve and will depend on the following:

- $F_V^{(l)} = \{f_V^{(l)} | v \in V\}$, the set of latent features on the node.
- $F_E^{(l)} = \bigcup_{f \in E} \{g_f^{(l)}(e) | e \in \mathscr{E}_f\}$, the set of latent features on all possible edges.
- $F_G^{(l)} = f_G^l$, the latent vector on the graph.

In other words, the loss function takes the form $L(F_V^{(l)}, F_E^{(l)}, F_G^{(l)})$. As an example based on 3.1, if our objective is classification, then assuming that $\tau : V \to [K]$ is the true block assignment vector for a graph G, the classification loss function can be defined by

$$L_{classification} = \sum_{i \in V} \left| \tau_i - \mathscr{K}\left(f_V^{(l)}(i)\right) \right|$$

where $\mathscr{K}\left(f_V^{(l)}(i)\right)$ is a decision function; we can here use any classification algorithm.

The first architecture, and one of the earliest examples, is the recurrent GNN [93] that is usually shortened to *RGNN* or *RecGNN*. As one can imagine from the name, RGNNs iteratively apply the communication functions until a steady state is reached. The node communication function is then defined by

$$h_v^{(1)} = \sum_{f \in E} \sum_{i \in \mathscr{N}_{f,in}(v)} h\left(F_v, F_v^{(l)}, F_i, F_i^{(l)}, g_f(i,v), g_f^{(l)}(i,v), w_v\right)$$
$$h_v^{(t)} = \sum_{f \in E} \sum_{i \in \mathscr{N}_{f,in}(v)} h\left(F_v, h_v^{(t-1)}, F_i, h_i^{(t-1)}, g_f(i,v), g_f^{(l)}(i,v), w_v\right)$$

Where F_v are features on the node V,$F_v^{(l)}$ is the current latent feature on node v, F_i are features on node i, $F_i^{(l)}$ is the current latent feature on node i, $g_f(i,v)$ are features on the edge between i and v, $g_f^{(l)}(i,v)$ is the latent feature on that edge, and w_v is a set of parameters

for the function. Sometimes we use the same set of parameters for all vertices, in other words, $w_v = w$ for all V. We stipulate that h_v is a contraction or Lipschitz continuous, that is, for some $C \in (0,1)$ the function satisfies $d(h(x) - h(y)) \leq Cd(x,y)$.

To simplify matters, here we are restricting our attention to graphs rather than hypergraphs. Additionally, for now at least, let us assume that we have no latent features on the edges or the graph, $h_G = 0, h_{e,f} = 0$, finally we will consider graphs with only one edge function, in other words we have,

$$h_v^{(1)} = \sum_{i \in \mathcal{N}_{in}(v)} h\left(F_v, F_v^{(l)}, F_i, F_i^{(l)}, g(i,v), w\right)$$
$$h_v^{(t)} = \sum_{i \in \mathcal{N}_{in}(v)} h\left(F_v, h_v^{(t-1)}, F_i, h_i^{(t-1)}, g(i,v), w\right)$$

In [93], one of the originators of GNN in general, the authors used a linear transformation,

$$h\left(F_v, h_v^{(t-1)}, F_i, h_i^{(t-1)}, g(i,v), w\right) = A_{v,i} h_v^{(t-1)} + b_v$$

Where $h_v^{(0)} = F_v^{(l)}$, the latent feature on node v from the previous step. To construct $A_{v,i}$ and b_v a Feedforward NN (FNN) was used to transform the features $\left(F_v, F_i, h_i^{(t-1)}, g_f(i,v)\right)$ into a square matrix, for each pair of nodes $v, i \in V$ and another FNN was used to transform them into b_v; these are the neural networks whose weight we train. The authors used a single set of parameters w_A and w_b for all h_v, but noted that this was a design choice to simplify the notation. As long as add the constraint $\max_{v \in V} \|\sum_{i \in \mathcal{N}_{in}(v)} A_{v,i}\|_1 < 1$, this function is a contraction. They show that this method can be used for subgraph matching, classification, and page rank.

The next architecture we will discuss is the Convolutional GNN (*ConvGNN*), based on graph convolutions, the one we discuss here is the spectral convolutional GNN [21]. The spectral graph convolution can be defined by the graph Fourier transform and the convolu-

tion theorem. We can represent the feature function f, as a vector where $f_i = f(i)$. For two node feature functions f and g, define= the graph convolution $*_G$ is defined by

$$\begin{aligned} f *_G g =& \mathfrak{F}^{-1}\left(\mathfrak{F}(f) \odot \mathfrak{F}(g)\right) \\ =& U^T(Uf \odot Ug) \end{aligned}$$

where $\odot$ is the element wise product. The function g is referred to as the signal, we can approximate this filter to gain computational advantage. [30] used the Chebyshev polynomial expansion for this approximation in their *ChebNet*. The advantage of using these polynomials is that by defining the operator $\mathscr{L}^*(A) = \frac{2\mathscr{L}}{\lambda_{\max}} - I$, we can show that for the ith Chebyshev polynomial $T_i(x)$, $T_i(\mathscr{L}^*) = UT_i(\Lambda)U^T$, where U and Λ are the matrices of the eigenvector and eigenvalues of $\mathscr{L}^*$. Since T_i is a polynomial and taking powers of Λ is easy, we gain a computational advantage. A further simplification was made in [54], where they showed that the layers of ChebNet can be approximated by

$$h_v^{(t)} = \sigma\left(W^{(t)} \sum_{i \in \mathscr{N}(v) \cup \{v\}} \bar{\mathscr{L}}_{i,v} h_v^{(t-1)}\right)$$

Where $\bar{\mathscr{L}}$ is the signless graph Laplacian, $h_v^{(0)} = f_v$ the node feature from the previous training step, $\bar{\mathscr{L}}(A) = I + DAD$, $W^{(t)}$ are the trainable weight matrix, and σ is an activation function.

The last architecture we consider is the Graph Matching Network proposed in [61]. We will later use this architecture in 4.1.2 for an application in social network analysis. For this network, we will use latent features on the edges and graphs. The main idea of GMN is to communicate information between multiple graphs, $G_i = (V_i, f_i, f_{V,i}, f_{E,i})$. We limit our attention to graphs that have nodes $V_i = \{k \in \mathbb{N}, \sum_{j=1}^{i-1} n_i < k \leq \sum_{j=1}^{i} n_i\}$, edge function $f_i : V_i \times V_i \to \{0,1\}$, node feature function $f_{V,i} : V_i \to \mathbb{R}^{d_v}$, and edge feature function

$f_{E,i} : V_i \times V_i \to \mathbb{R}^{d_e}$. There are multiple constructions for the architecture, the case we used is based on *Multi-Level Perceptron*, let $E_i = \{(i,j) \in V_i^2 | f_i(i,j) = 1\}$ then for each edge $e = (i,j) \in E_k$ each layer we define the edge latent feature functions as

$$\begin{aligned} h_e^{(t)} =& MLP_E^{(t)}\left(h_e^{(t-1)}, h_i^{(t-1)}, h_j^{(t-1)}\right) \\ h_{(i,j)}^{(0)} =& MLP_E^{(0)}\left(f_{E,k}(i,j), f_{V,k}(i), f_{V,k}(j)\right) \end{aligned}$$

We define the cross graph edge latent feature function for graphs G_ℓ, G_k for each edge $e \in V_\ell \times V_k$ by

$$\begin{aligned} h_{(i,j),c}^{(t)} =& \frac{e^{\|h_i^{(t-1)} - h_j^{(t-1)}\|_2}}{\sum_{i' \in V_\ell} e^{\|h_{i'}^{(t-1)} - h_j^{(t-1)}\|_2}} \left(h_j^{(t-1)} - h_i^{(t-1)}\right) \\ h_{(i,j),c}^{(t)} =& \frac{e^{\|f_{V,\ell}(i) - f_{V,k}(j)\|_2}}{\sum_{i' \in V_\ell} e^{\|f_{V,\ell}(i') - f_{V,k}(j)\|_2}} \left(f_{V,k}(j) - f_{V,\ell}(i)\right) \end{aligned}$$

and similarly for each edge $e \in V_k \times V_\ell$. The node communication function is defined by

$$\begin{aligned} h_v^{(t)} =& MLP_V^{(t)}\left(h_v^{(t-1)}, \sum_{i \in \mathcal{N}(v)} h_e^{(t)}(i,v), \sum_{i \in V_k} h_{(i,v),c}^{(t)}\right) \\ h_v^{(0)} =& MLP_V^{(0)}\left(f_{V,\ell}(v), \sum_{i \in \mathcal{N}(v)} h_e^{(0)}(i,v), \sum_{i \in V_k} h_{(i,v),c}^{(0)}\right) \end{aligned}$$

For the graph latent feature function, let T be the number of layers and define

$$h_{G_i} = MLP_G\left(\sum_{v \in V_i} \sigma\left(MLP_{gate}\left(h_v^{(T)}\right)\right) \odot MLP\left(h_v^{(T)}\right)\right)$$

To define the loss function for this network we will define the true label function $\tau : \mathcal{G} \times \mathcal{G} \to \{-1, 1\}$, where $\tau(G_i, G_j) = 1$ for a positive match and $\tau(G_i, G_j) = -1$ for negative matches, and suppose we have the set $\{(G_{\ell_i}, G_{k_i}, \tau(G_{\ell_i}, G_{k_i}))\}_{i=1}^{n_\tau}$ of graph pair examples.

The loss function is then defined by

$$L_{GMN} = \frac{1}{n_\tau} \sum_i \max \left\{ 0, \gamma - \tau(G_i, G_j) \left(1 - \| h_{G_{\ell_i}} - h_{G_{k_i}} \|_2 \right) \right\}$$

where γ is a tuning parameter.

Chapter 4: Simulations and Applications to Hierarchical Stochastic Block Models

In this chapter, we will discuss novel results in analyzing user-in-the-loop supervision in the context of subgraph matching in the first chapter and testing for repeated motifs and inferring hierarchical structure for block models in the second. We will test these results on simulated and real data examples.

4.1 User-in-the-loop Supervision

Interactive machine learning, via incorporating user-in-the-loop supervision, can lead to an enhanced user experience and better downstream inference performance [3]. In subgraph nomination, the need for user-in-the-loop supervision can be understood as follows. Consider nominating in $((g_2, H_2), S_2)$ at level k, where $S_2 = \{(k, \ell)\}$, and $|\mathfrak{I}(B^k_{\ell,2}; g_2)| > 1$. Even if $\widehat{H}_2$ agrees with H_2 at level k, there are multiple subgraphs in (g_2, H_2) that are isomorphic to the unknown subgraph of interest, and the HSN scheme has no information available to distinguish these. In this setting, it is reasonable to model the relative order of the elements in $\mathfrak{I}(B^k_{\ell,2}; g_2)$ in an HSN scheme as (effectively) uniformly random.

For example, if $|\mathfrak{I}(B^k_{\ell,2}; g_2)| = c$, then considering top h of our nomination list ($h < c$),

Code and details can be found at https://github.com/alfahadalqadhi

a scheme that identifies the correct structure for $B^k_{\ell,2}$ would still have probability

$$\frac{\binom{c-1}{h}}{\binom{c}{h}} = \frac{c-h}{c} = 1 - \frac{h}{c}$$

of not finding $B^k_{\ell,2}$. This can be mitigated by the following user-in-the-loop system:

i. The user can take a limited number of single vertex inputs and can output whether that vertex is interesting or not (i.e., part of an interesting subgraph). This output can be modeled as error-free (oracle-user) or errorful.

ii. The supervision can then be used to re-rank the subgraphs in the nomination scheme.

While practically, we do not suspect that there will be many perfect matches to the subgraphs of interest in the hierarchy provided by H_2, it may be the case that there are many subgraphs "close" to the subgraph of interest in the (possibly lossy) estimate $\widehat{H}_2$, in which case this supervision is equally necessary.

What follows is a formalization of the above heuristic. We first define the concept of a user-in-the-loop; noting that our definition is not the most general, as we focus in our analysis on binary users; i.e., they can only output 1 (yes) or 0 (no) for the interestingness of a vertex. In general, one might allow for the user to use a rating system with more levels, or even add a continuous space for responses.

Definition 4.1 (VN User)**.** *Let $(g_1, H_1) \in \mathscr{HGS}_n$ with indices of interest S_1 and $(g_2, H_2) \in \mathscr{HGS}_m$ with indices of interest S_2. Let $(\theta, \gamma) \in [0,1] \times [0,1]$ and $t \in \mathbb{Z} > 0$ with $t < m$. We define the capacity t,* HSGN user-in-the-loop *for $((g_2, H_2), S_2)$ with parameters (θ, γ) (denoted U_t) as follows:*

i. Letting Ω be the underlying sample space, the user is a function

$$U_t : \mathscr{T}_{V_2,t} \times \Omega \mapsto \{0,1\}^t$$

where $\mathscr{T}_{V_2,t}$ is the set of ordered t-tuples of distinct elements from $V_2 = V(g_2)$. Note that dependence on Ω will be suppressed when appropriate.

ii. For each $\eta \in \mathscr{T}_{V_2,t}$ and $x \in \{0,1\}^t$, let $\mathfrak{F}_{\in,i} = \{\eta_i \in \bigcup_{(s_1,s_2)\in S_2} B^{s_1}_{s_2,2}\}$, $\mathfrak{F}_{\notin,i} = \{\eta_i \notin \bigcup_{(s_1,s_2)\in S_2} B^{s_1}_{s_2,2}\}$, define

$$\mathbb{P}(U_t(\eta) = x) = \prod_{i=1}^{t} \left[\left(\mathbb{1}\{\mathfrak{F}_{\in,i}\} \theta^{x_i} (1-\theta)^{1-x_i} \right) + \left(\mathbb{1}\{\mathfrak{F}_{\notin,i}\} \gamma^{x_i} (1-\gamma)^{1-x_i} \right) \right] \tag{4.1}$$

In essence, the user has independent binary components where the Bernoulli success probabilities depend only on the membership (or lack thereof) in a subgraph of interest.

For a given $((g_2,H_2),S_2)$ and t, we define the set of all such users by $\mathfrak{U}_t = \mathfrak{U}_{((g_2,H_2),S_2),t}$.

Effectively, the HSN user-in-the-loop outputs a sequence of $\{0,1\}$ values, one for each vertex in a training set η. An output of 1 is considered the positive answer, i.e. an interesting vertex; for vertices in $\cup_{(s_1,s_2)\in S_2} B^{s_1}_{s_2,2}$, this is a correct answer, and an error otherwise. An output of 0 is the negative answer meaning that this vertex is not of interest; for vertices not in $\cup_{(s_1,s_2)\in S_2} B^{s_1}_{s_2,2}$, this is a correct answer and is an error otherwise. This allows us to simultaneously model oracle users ($\theta = 1$, and $\gamma = 0$) and errorful users ($\theta < 1$, and $\gamma > 0$). The capacity of the user (i.e., t) allows us to model the practical setting in which user-in-the-loop resources are costly and only limited supervision is available. Users can be incorporated into HSN schemes as follows.

Definition 4.2. *User Aided HSN Scheme (UHSN) Consider the setting in Definition (3.6). Let $t \leq n_j^{(2)}$ and $U_t \in \mathfrak{U}_t$. Consider an HSN $\Phi_{n,m} = (\widehat{H}_2, \Delta)$, and let η be an ordered t-tuple*

of distinct elements of $V_2 = V(g_2)$. For each k, U_t acts on $\Phi^k_{n,m}$ as follows:

i. *Given η, U_t is distributed according to Eq. 4.1. Let the output of the user process be denoted $x \in \{0,1\}^t$.*

ii. *Let*

$$\begin{aligned} I_k := &\{\ell \in [\hat{m}_k] \text{ s.t. more than half} \\ &\text{of the vertices in } \eta \cap \widehat{B}^k_{\ell,2} \\ &\text{were labeled interesting (i.e., as 1)} \\ &\text{by the user}\} \\ N_k := &\{\ell \in [\hat{m}_k] \text{ s.t. at least half} \\ &\text{of the vertices in } \eta \cap \widehat{B}^k_{\ell,2} \\ &\text{were labeled as not interesting (i.e., as 0)} \\ &\text{by the user}\} \end{aligned}$$

If $\eta \cap \widehat{B}^k_{\ell,2} = \emptyset$, then $\ell \notin I_t \cup N_t$.

iii. *Let the indices of I_t be indexed via $(\mathfrak{i}_1, \cdots, \mathfrak{i}_{|I_t|})$ where (according to $\Phi^k_{n,m}$)*

$$\widehat{B}^k_{\mathfrak{i}_1,2} > \widehat{B}^k_{\mathfrak{i}_2,2} > \cdots > \widehat{B}^k_{\mathfrak{i}_{|I_t|},2}.$$

Similarly index N_t via $(\mathfrak{n}_1, \cdots, \mathfrak{n}_{|N_t|})$, and $M_t := [\hat{m}_k] \setminus \{I_t, N_t\}$ (the unsupervised sub-

graph indices) via $(\mathfrak{m}_1, \cdots, \mathfrak{m}_{|M_t|})$ *The user improved ranking is then given by*

$$\begin{aligned}\Phi^{k,U}_{n,m} = \Bigg(& \widehat{B}^k_{\mathfrak{i}_1,2} > \widehat{B}^k_{\mathfrak{i}_2,2} > \widehat{B}^k_{\mathfrak{i}_{|I_t|},2} > \widehat{B}^k_{\mathfrak{m}_1,2} > \cdots \\ & \cdots > \widehat{B}^k_{\mathfrak{m}_2,2} > \widehat{B}^k_{\mathfrak{m}_{|M_t|},2} > \widehat{B}^k_{\mathfrak{n}_1,2} > \cdots \\ & \cdots > \widehat{B}^k_{\mathfrak{n}_2,2} > \widehat{B}^k_{\mathfrak{n}_{|N_t|},2} \Bigg).\end{aligned}$$

Remark 12. We note here the possibility of even an oracle user introducing large errors into a ranking scheme based on an approximate $\widehat{H}_2$. Indeed, even if $\widehat{B}^k_{\ell,2}$ is very similar to an interesting subgraph in g_1 according to Δ, it is still possible that uninteresting vertices are chosen to provide to the user and $\ell \in N_t$. This can be mitigated by choosing multiple vertices from each of some of the top ranked subgraphs for the user to evaluate.

Remark 13. For iterative applications of the user-in-the-loop, we can sequentially run the user-in-the-loop, first on the output of $\Phi_{n,m}$, then on $\Phi^U_{n,m}$ (with t new user training points), and so on.

4.1.1 The (Theoretical) Benefit of the User-in-the-loop

We turn our attention now to take a look at the theoretical benefit of the user-in-the-loop in the setting of HSN. As in the motivating example for the user, the setting for this section will be as follows. Letting $(g_1, H_1) \in \mathscr{HGS}_n$ and $(g_2, H_2) \in \mathscr{HGS}_m$ with respective interesting index sets S_1 and S_2, consider a HSN scheme $\Phi_{n,m}$ with $\widehat{H}_2 = H_2$. Consider the nomination provided by $\Phi^k_{n,m}$ and the simple setting in which $S_2 = \{(k,\ell)\}$ with $|\mathfrak{I}(B^k_{\ell,2}; g_2)| = c > 1$. In this setting, it is reasonable to model the relative order of the elements in $\mathfrak{I}(B^k_{\ell,2}; g_2)$ in an $\Phi^k_{n,m}$ as (effectively) uniformly random (though in practice, they are a fixed arbitrary order).

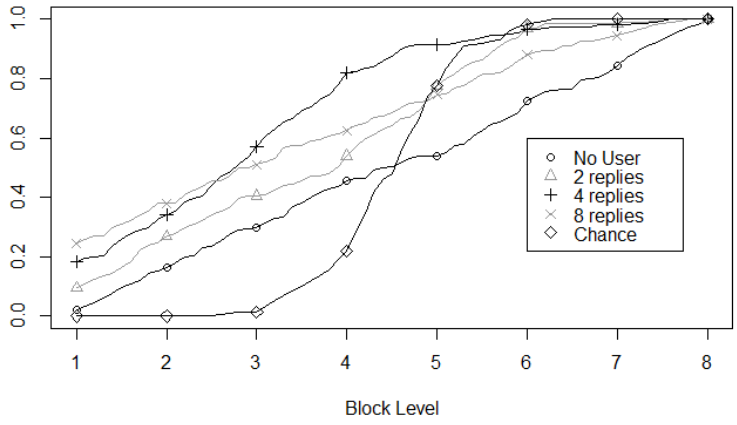


Figure 4.1: We consider 2000 Monte Carlo replicates of distributions as in Section 4.1.2, and from each distribution, we generate 100 networks. We use Method 1, and performance with an oracle user (varying the replies for the user; i.e., the amount of supervision provided) and the chance algorithm are compared. In all cases, we plot the proportion of trials (y-axis) that ranked the correct block in the nth place (x-axis) which represents $1 - L_n(\phi, B^*)$.

Theorem 4.1. *Given the setup above where the scheme $\Phi_{n,m}$ effectively ranks the elements of $\mathfrak{I}(B^k_{\ell,2}; g_2)$ uniformly at random at the top of the rank list, let $t \leq c < m_k$ be the capacity of the user-in-the-loop. Consider any training set for the user where η contains exactly one element of each $\Phi^k_{n,m}[h]$ with $1 \leq h \leq t$. Let E_h denote the event that $\Phi^U_{n,m}$ ranks $B^k_{\ell,2}$ not in the top h. Considering $c > t, h$, we have the following.*

i. Let U_t be an oracle user, then $\mathbb{P}(E_h) = \max(1 - \frac{h+t}{c}, 0)$.

ii. Let $(\theta, \gamma) = (1-p, 0)$ for $p < 1$, then

- *If $t + h \leq c$, then $\mathbb{P}(E_h) = 1 - \frac{h}{c} - (1-p)\frac{t}{c}$; note that this is strictly less than $1 - h/c$ for all $p \in (0,1)$.*
- *If $t + h > c$, then $\mathbb{P}(E_h) = \frac{tp}{c}$; note that if $p < (c-h)/t$, this is strictly less than $1 - h/c$.*

iii. Let $(\theta, \gamma) = (1-p, q)$ for $p < 1$ and $q > 0$. Then, letting $F(i; n, p)$ be the CDF of a

Binomial(n,p) *random variable evaluated at i, we have*

- *If* $h+t \leq c$, *and* $t \leq h$, *then* $\mathbb{P}(E_h) = 1 - \frac{h}{c} - \frac{(1-p-q)t}{c}$; *note that if* $p+q<1$, *this is less than* $1-\frac{h}{c}$.
- *If* $h+t > c$, *and* $t \leq h$, *then* $\mathbb{P}(E_h) = \frac{pt}{c} + \frac{1}{c}\sum_{i=1}^{c-h} F(i-1;t,1-q)$; *this is upper bounded by* $(p+q)t/c$ *and if* $(p+q) < (c-h)/t$, *this is strictly less than* $1-h/c$.
- *If* $h+t \leq c$, *and* $t > h$, *then*

$$\begin{aligned}\mathbb{P}(E_h) =& 1 - \frac{h}{c} - \frac{(1-p)t}{c} \\ &+ \frac{1-p}{c}\sum_{i=1}^{t-h} F(i-1;h+i-1,1-q) \\ &+ \frac{1}{c}\sum_{i=t-h+1}^{t} F(i-1;t,1-q)\end{aligned}$$

  Note that a rough upper bound of this is given by $1-h/c+qt/c-(1-p)h/c$, *and if* $qt < (1-p)h$, *this is strictly less than* $1-h/c$.

- *If* $h+t > c$, *and* $t > h$, *then*

$$\begin{aligned}\mathbb{P}(E_h) =& \frac{pt}{c} \\ &+ \frac{1-p}{c}\sum_{i=1}^{t-h} F(i-1;h+i-1,1-q) \\ &+ \frac{1}{c}\sum_{i=t-h+1}^{c-h} F(i-1;t,1-q)\end{aligned}$$

  Note that a rough upper bound of this is given by $\frac{(1+q)t-(1-p)h}{c}$, *and if* $(1+q)t+ph < c$, *this is strictly less than* $1-h/c$.

We, perhaps surprisingly, see that there are p and q values that guarantee improvement

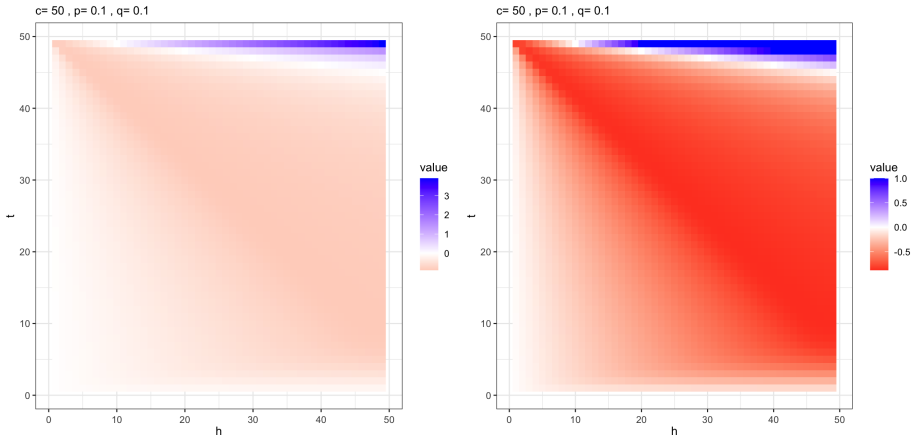


Figure 4.2: For $c = 50$ we plot (for $h,t \in \{1,2,\ldots,49\}$) the relative loss $R(c,t,h,p,q)$ in the setting of Theorem 4.1 part (iii) in the left panel $\min(R(c,t,h,p,q),1)$ in the right panel.

(over the user-in-the-loop-free setting) in all cases. The conditions in part *iii*. bear further consideration. The interaction of p,q,h,t here is nuanced, and is most easily analyzed numerically, as is shown in Figures 4.2 and 4.3. Letting $L(c,t,h,p,q)$ denote the loss from Theorem 4.1 part (iii) (where we can interpret this in the context of Definition 3.7 by considering Δ as a $0/1$ oracle dissimilarity function), for $c = 50$ we plot the relative loss improvement,

$$R(c,t,h,p,q) = \frac{L(c,t,h,p,q) - (1-h/c)}{1-h/c}, \tag{4.2}$$

for h between 1 and 49 (on the y-axis), and t between 1 and 49 (on the x-axis). Different panels correspond to different values of p (varies by row) and q (varies by column). Note that in Figures 4.2, we plot the relative loss in the left panel $R(c,t,h,p,q)$ and $\min(R(c,t,h,p,q),1)$ in the right panel; this truncating aids in distinguishing the areas of improvement from those where $R(c,t,h,p,q) > 0$.

In each plot, darker red areas correspond to parameter settings where the loss is improved with user-supervision, and darker blue areas to parameter settings where the loss is increased with user-supervision (due to the user error). A few trends are clear from the figures. Unsurprisingly, supervision becomes detrimental (i.e., more supervision leads to

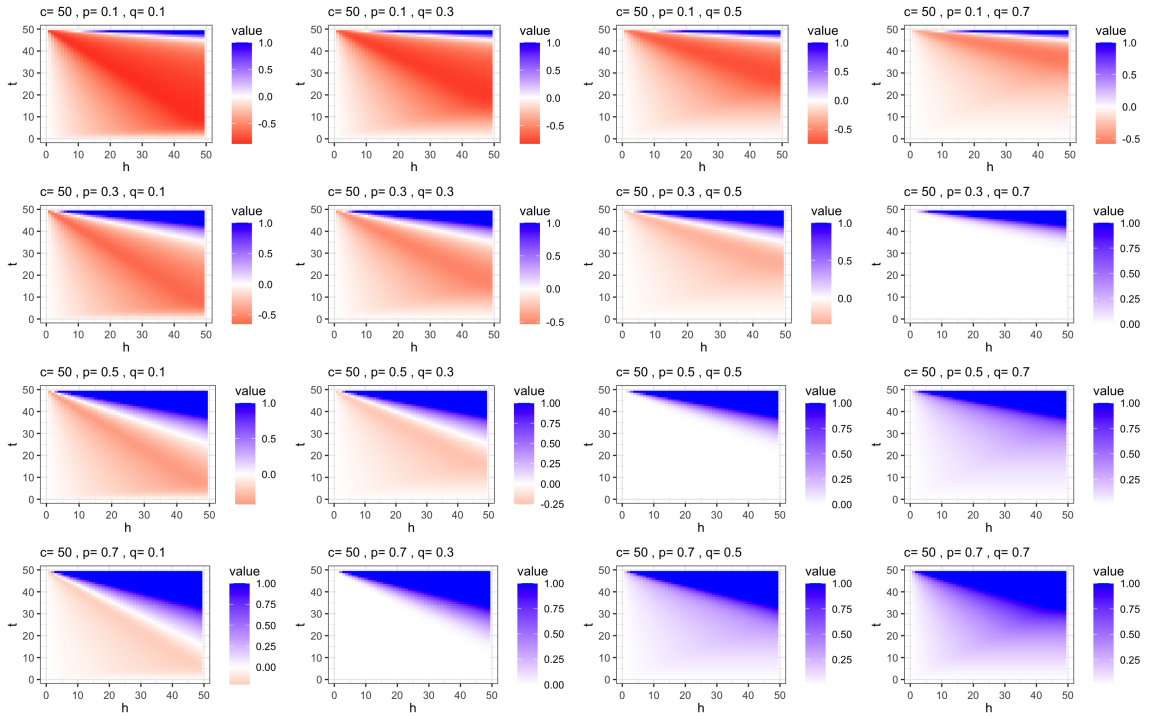


Figure 4.3: For $c = 50$ we plot (for $h,t \in \{1,2,\ldots,49\}$) $\min(R(c,t,h,p,q),1)$ for the loss function in the setting of Theorem 4.1 part (iii) and R defined in Equation (4.2). Different panels correspond to different values of p and q.

more loss) as p and q increase and $q \geq 0.5$, although the loss appears to be more tolerant of higher values of p than it is of q. In all cases, large values of t and h simultaneously lead to training becoming less effective, though $h < t$ seems to yield resilience to larger user-supervised loss for all p,q pairs.

In the event that the hierarchical clustering is noisily recovered in g_2 (i.e., $\widehat{H}_2 \neq H_2$), even the supervision provided by an oracle user-in-the-loop may be worse than no extra supervision. Indeed, consider the setting where $S_2 = \{(k,\ell)\}$ and

$$\alpha = \frac{|\widehat{B}^k_{\ell,2} \cap B^k_{\ell,2}|}{|\widehat{B}^k_{\ell,2}|}; \quad \beta = \min_{i \neq \ell} \frac{|\widehat{B}^k_{i,2} \cap B^k_{\ell,2}|}{|\widehat{B}^k_{i,2}|},$$

then, adopting the notation and setting above, the probability that $\widehat{B}^k_{\ell,2}$ is not ranked in

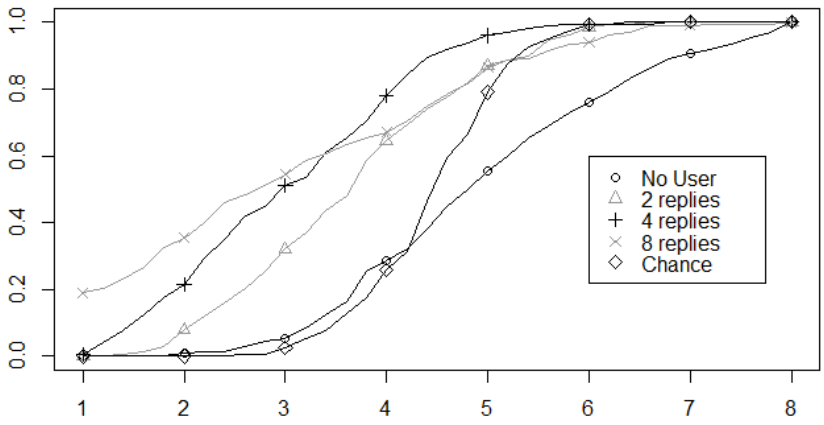


Figure 4.4: We consider 2000 Monte Carlo replicates of distributions as in Section 4.1.2, and from each distribution, we generate 100 networks. We use Method 2, and performance with an oracle user (varying the replies for the user; i.e., the amount of supervision provided) and the chance algorithm are compared. We plot the proportion of trials (y-axis) that ranked the correct block in the nth place (x-axis) which represents $1 - L_n(\phi, B^*)$

the top h after oracle U_t supervision is bounded below by the probabilities in part (iii) of Theorem 4.1 with $p = \alpha$ and $q = \beta$. In particular, there are values of α and β under which this error is worse than $1 - h/c$ (the error sans additional supervision).

4.1.2 Simulations and Real Data Experiments

We now provide empirical evidence for the theory we have outlined. Namely, we will show through simulations and real data examples the impact of a user-in-the-loop. We first consider simulations where graphs are drawn from an HSBM distribution. In this setting we evaluate our methodology on the task of nominating subgraphs with a similar motif to our subgraph of interest. We then consider 57 pairs of analyzed human neural connectomes for which brain regions (i.e., communities) are well defined and the community for each voxel is known. We evaluate our methodology on the task of nominating the same brain

region within a given pair of connectomes with varying levels of a user-in-the-loop. In

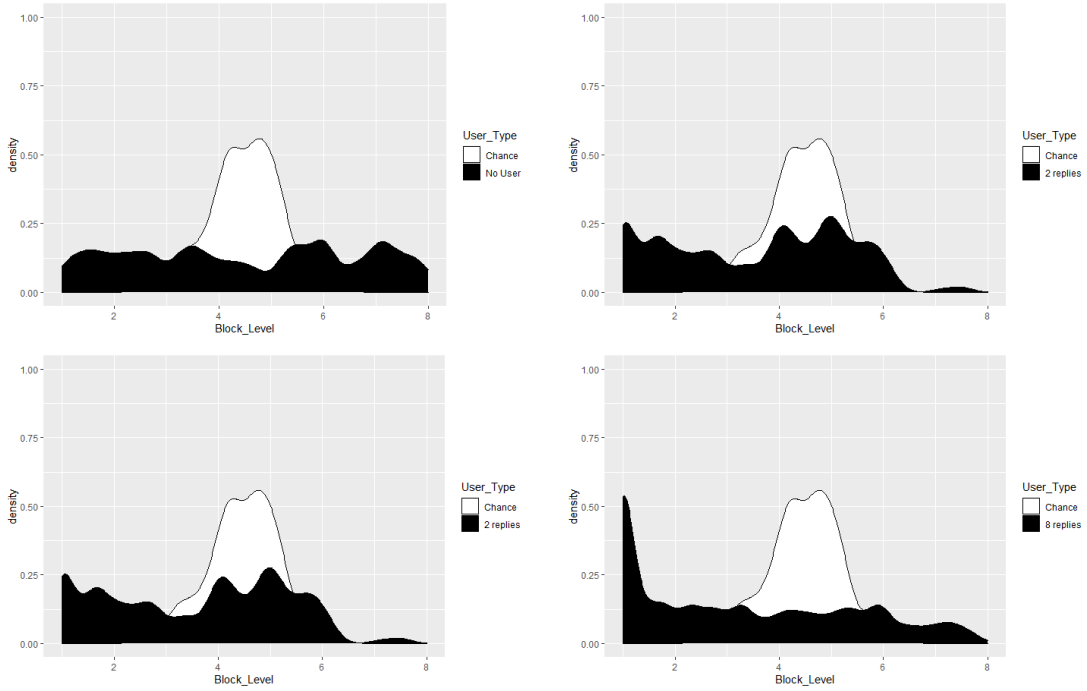


Figure 4.5: We consider 2000 Monte Carlo replicates of distributions as in Section 4.1.2, and from each distribution, we generate 100 networks. We use Method 1, and performance with an oracle user (varying the replies for the user; i.e., the amount of supervision provided) and the chance algorithm are compared. We plot the density of all trials (y-axis) that ranked the correct block in the nth place (x-axis).

this section we will first provide a complete description of the model and procedures that were employed to test our theory in a simulated data setting. We then discuss results and how they fit within our theory.

Model description

For our first example, we consider nominating within the Hierarchical Stochastic Block-model setting of [66]. We consider sampling from a 2-level HSBM that has 16 blocks in the first level and 3 sub-blocks in each of the first level blocks (so that this model can also be realized as a 48 block standard SBM). Further, the blocks of the first level belong to

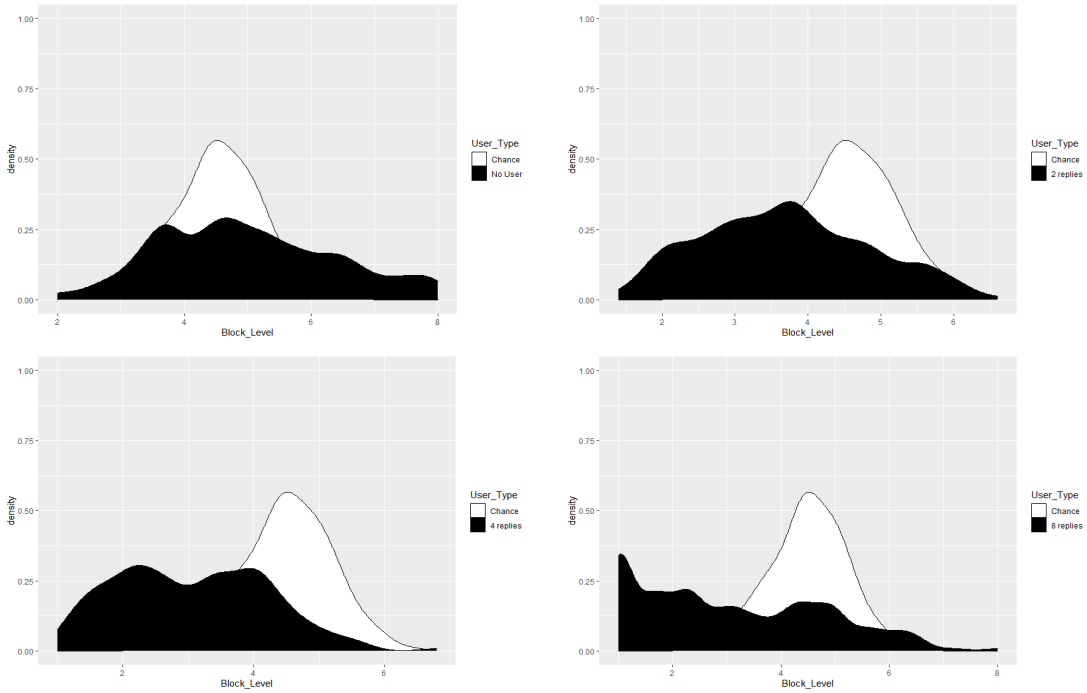


Figure 4.6: We consider 2000 Monte Carlo replicates of distributions as in Section 4.1.2, and from each distribution, we generate 100 networks. We use Method 2 and performance with an oracle user (varying the replies for the user; i.e., the amount of supervision provided) and the chance algorithm are compared. We plot the density of all trials (y-axis) that ranked the correct block in the nth place (x-axis).

one of three motifs B_1, B_2, or B_3, with constant cross-community connection probability $p = 0.01$ for the first level. Formally, we are sampling from the following 2-level HSBM:

i. There are $K_1 = 16$ blocks in the first level of the hierarchy, and the size of the blocks is i.i.d. $10 * \lfloor 20 + 50 * Unif(0,1) \rfloor$; note that this differs slightly from the random block assignment HSBM defined previously.

ii. The motifs $\mathbf{B}_i \in \mathbb{R}^{3\times 3}$ for $i = 1,2,3$ are i.i.d. samples satisfying

$$\mathbf{B}_i = X_i^T X_i \text{ where } X_i \in \mathbb{R}^{3\times 3}$$
$$\text{satisfies } X_i[\cdot, j] \overset{i.i.d.}{\sim} Dirichlet(1,1,1);$$

of course, this distribution is artifical, but it provides a maximum entropy sampling of the X's which provides a difficult testing setting for our algorithm.

iii. For each $j \in [16] \setminus \{9\}$, we have $B_2^j = \mathbf{B}_i$ independently with probability $1/3$ for $i = 1,2,3$. B_2^9 is set to equal B_2^1;

iv. In the second level of the hierarchy ($K_2^j = 3$ for all j), the block sizes are defined via (where $\lfloor \cdot \rceil_1$ rounds the number to the nearest tenth)

$$\begin{aligned} |b_{(j,i)}^{(2)}| &= |\{v \in V | b^{(1)}(v) = j, b^{(2)}(v) = i\}| \\ &\overset{i.i.d.}{\sim} |b_j^{(1)}| \lfloor Dirichlet\,(\omega) \rceil_1(i), i < 3 \end{aligned}$$

where the entries of $\omega = (\omega_1, \omega_2, \omega_2)$ are

$$\omega_i \overset{i.i.d.}{\sim} Unif(2,10).$$

Lastly, we set

$$\begin{aligned} |b_{(j,3)}^{(2)}| &= |\{v \in V | b^{(1)}(v) = j, b^{(2)}(v) = 3\} \\ &= |b_j^{(1)}| - |b_{(j,i)}^{(2)}| - |b_{(j,i)}^{(2)}| \end{aligned}$$

This particular parameterization is chosen to produce different motifs that stress our HSGN framework by producing motifs that are very similar, and motifs where the block structure is extremely subtle (i.e., close to flat), and cases where both these problems happen, as it draws from a distribution in which it is not uncommon that the motifs turn out to be extremely similar and/or very flat.

The block structure was inferred by clustering via Gaussian mixture modeling after embedding, utilizing the **R** package **Mclust** [97] to cluster the graph we are nominating from

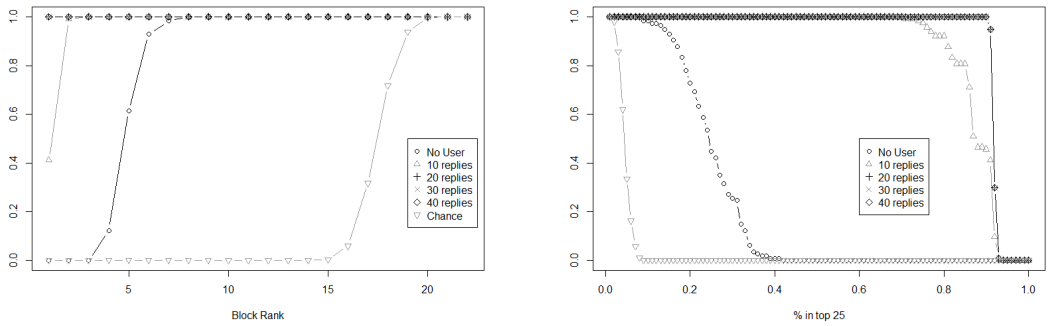


Figure 4.7: With perfect knowledge of the block structure of the brain (the division into 71 regions provided by the data) and using Method 1, (left) we plot on the y-axis the proportion of subjects (averaged across all blocks considered as the block-of-interest in the left hemisphere) that ranked the correct block in the right hemisphere the n-th place (x-axis); (right) we plot on the y-axis the the proportion of subjects (averaged across all blocks considered as the block-of-interest in the left hemisphere) for which we find the total proportion of the block of interest in the top 25 vertices nominated (x-axis).

into 8 sub-graphs. After that, we use two different methods to infer similarity. The first approximates Δ via the value of the non-parametric test statistic of [107] computed between re-embeddings of each inferred community (as in [66]); simulation results are shown in Figures 4.1 and 4.5. We will call this *Method 1* in the sequel. The second approximates Δ between communities i and j via the scaled graph matching pseudo-distance [38],

$$1 - \frac{\min_{P \in \Pi(n)} \|A_i - PA_jP^T\|_F^2}{\frac{1}{n!}\sum_{P \in \Pi(n)} \|A_i - PA_jP^T\|_F^2}$$

where either A_i (the induced subgraph of community i) or A_j has been appropriately padded as in [36]); results are shown in Figures 4.4 and 4.6. We will call this *Method 2* in the sequel.

The user replies are those of an oracle. However, as discussed previously since we do perfectly recover the true network partition (rather than using an estimate of the true network partition), our user could still end up being errorful. In particular, the user could be asked to rank vertices that are not in the subgraph of interest. This exemplifies the behavior characterized by *iii* of Theorem 4.1, albeit with a different q for each of the blocks. Another

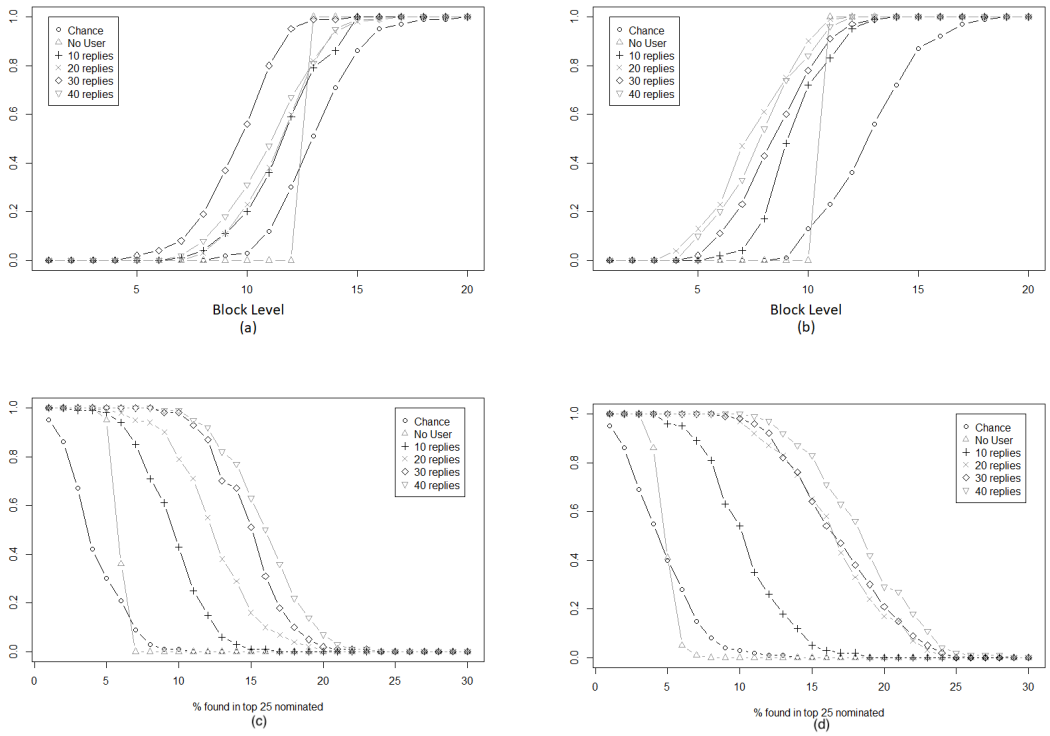


Figure 4.8: Clustering the graph using Gaussian Mixture Modeling on the embedded network, in the left two panels, we use Method 1 to estimate Δ, while the right two panels we use Method 2. In the top row, on the y-axis we plot the proportion of subjects (averaged across all blocks considered as the block-of-interest in the left hemisphere) that ranked the correct block in the right hemisphere the at worst x-th place (x-axis). In the bottom row, on the y-axis we plot the the proportion of subjects for which we find the total proportion of the block of interest in the top 25 vertices nominated (x-axis).

significant difference in the scheme of the simulation is that the nomination process takes the first affirmative reply as the correct community and returns the others in their order as is.

Results & Discussion

In this section we will go over the results of the experiments we described. The aim is to bolster the claims in the theoretical results with experimental results, as such we will have a special focus on the features that are iconic from the theory. First we start with

Method 1, the experiment that uses the test statistic form [107] to estimate the dissimilarity function.

In Figure 4.1 we can see that the user training provides an improvement over the original algorithm ("No user"), as suggested by Theorem 4.1, with more training often yielding superior performance. Moreover, especially for small x-values, the the algorithm outperforms a chance ranking of the obtained blocks. Interestingly, we see that more user-supervision is not always better. Indeed, as mentioned previously, in our regime of lossy block recovery additional supervision can be deleterious. Figure 4.5 elucidates this finding further from the perspective of the distribution density of the correct block in each position. We notice that as user-supervision increases the distribution are shifted towards putting the correct block in the first position. We also notice the trend predicted by Figure 4.3, as we increase h (the nomination error level) the ideal value of t (user-provided replies) is decreasing, and less supervision is preferred. We see this, as the curves with less supervision overtake those of higher supervision as we plot $1 - L_n(\phi, B^*)$. However we note that here the value of q is not constant and hence the minimizers of the loss function do not not adhere to the figure exactly. In Figures 4.4 and 4.6 we see similar trends using Method 2. Here the main contribution to the error of the methods is the misclustering of the vertices in the initial GMM step, and both dissimilarity estimates provide good estimates of the the latent Δ. In the real data example considered below, there is a more pronounced differentiation across methods.

Subgraph nomination in BNU1 connectomes

Data acquisition resources and limitations can be eased through automation. For example, in the study of human connectomes, one of the more labour intensive steps, is classifying different parts of the brain. One such classification task is finding corresponding regions of interest across hemispheres in a human connectome. This task is time and

human resource intensive when done manually, and automation is essential for achieving a high throughput [45]. To illustrate our HSN methodology further, we will consider then the task of nominating brain regions across hemispheres in the DT-MRI derived brain networks in the BNU1 database [122]. The spatial image of the brain includes information about its structure; in other words, the neural fibers that run through different areas of the brain. A graph of the subject's brain is then created with the following definitions: nodes or vertices correspond to different spatial units (usually the brain is divided into cubic sections called volumetric pixels, or voxels), edges are weighted based on the amount of neural fibers that run through two regions. One may hypothesize that there is a sense of structural symmetry between paired regions across hemispheres, and discovery of this pairing is the inference task for this HSN application.

Here, we apply the user aided subgraph nomination scheme to a connectomic dataset. The data we apply this to is a set of 114 neuronal networks—two repeated scans for each of 57 human subjects—derived from the BNU1 connectome dataset [122]. Each brain scan sections the brain into one thousand voxels—or volumetric pixels—with weighted edges between them indicating the amount of neuronal batches that are shared, indicating connectedness in a structural but not necessarily functional sense. Additionally each voxels contains information on it detailing its membership to the left or right hemispheres, membership to one of the 71 neuronal regions [32], whether it is grey or white matter, and its XYZ coordinates in the partially registered scan. The relative size of the networks is then ≈ 1000 vertices, with pre-defined neuronal regions varying in size between 2 and 100 vertices. As in the simulation, although the distributional properties in paired regions across hemispheres is often similar (i.e. similar motifs), there are no guarantees that the numbers of vertices that make up each region are equal. Errors from this (and misclustering) are magnified in smaller subgraphs, especially since there is less information about the motif in the small subgraph setting. The aforementioned sources of errors are mitigated by

only considering paired regions that have at least 10 vertices in each hemisphere and using similarity metrics (Methods 1 and 2) that do not depend on the size of a region.

Our procedure can then be described as follows, considering each pair of sufficiently large matched regions (≥ 10 vertices in each hemisphere) separately, first we use the information given by spectrally embedding the adjacency matrix of the connectome (and the collected spatial coordinate data in Figure 4.9) to infer the subgraph structure in the right hemisphere via clustering by GMM [97]; using the region-of-interest in the left hemisphere as our training, next we use Methods 1 and 2 to rank above to provide different estimates of Δ. We then supply a user-in-the-loop with a vertex (or a few vertices) from each of the top k communities to re-rank the nomination list. We plot the performance of our algorithms with perfect knowledge of the block structure of the brain (the division into 71 regions provided by the data) and using Method 1 in Figure 4.7. We plot:

- (Left) on the y-axis the proportion of subjects (averaged across all blocks considered as the block-of-interest in the left hemisphere) that ranked the correct block in the right hemisphere in the at worst x-th place (x-axis).

- (Right) on the y-axis the the proportion of subjects for which we find the total proportion of the block of interest in the top 25 vertices nominated (x-axis).

Note that an artifact of the metric in panel (b) is that we may never find a complete block that is of a size larger than 25 this is why we see a drop near 90%. As ideal performance in the left and right panels corresponds to the function $f(x) \equiv 1$, this figure enforces the intuition that, given high fidelity clusters, our ranking procedure is significantly better than chance and benefits strongly from user-in-the-loop supervision.

In Figure 4.8, in the setting of potentially errorful clusters (obtained via `Mclust` applied to the adjacency spectral embeddings of the brain networks, we plot (similar to in Figure 4.7), in the top panels the proportion whose match is found in the top x (similar to the left

panel of Figure 4.8) and in the bottom panels the percent who had the proportion of the top 25 vertices come from the block of interest (similar to the right panel of Figure 4.8). The left two panels use Method 1 to estimate Δ, while the right two panels use Method 2. In light of perfect performance corresponding the the constant 1 function in all cases, we see here that Method 2 achieves better performance (especially for large values on the x-axis in each figure) than Method 1 both when making use of the user-supervision and in the no-user setting. Here, we see the general trend that the methods perform poorly without a user-in-the-loop (due to the error in the clustering; see Figure 4.9), but that the method can efficiently make use of the user to achieve better performance. As was the case previously, the clustering is errorful, and on average this can have a deleterious effect on the user-supervision (top row). However, in the bottom row we see that *sample-wise* the supervision is monotonically beneficial.

We repeat the above experiment using Method 2 and incorporating the XYZ coordinates of each voxel into the clustering (by appending the features onto the spectral graph embeddings), and plot the results in Figure 4.9. The increased clustering fidelity achieved by incorporating the vertex features manifests itself here by the performance gain achieved in the no-user setting when compared with Figure 4.8. As a result of this, better performance is achieved across all settings here (again compared to Figure 4.8). While the clustering here is still errorful, and on average this can have a deleterious effect on the user-supervision, the right panel again shows that sample-wise the supervision is monotonically beneficial. This means that a user can successfully aid in finding larger portions of the block of interest. Therefore, we can use these vertices in the input to further investigate and find the rest of the block, by searching in the k-nearest neighbors of these vertices.

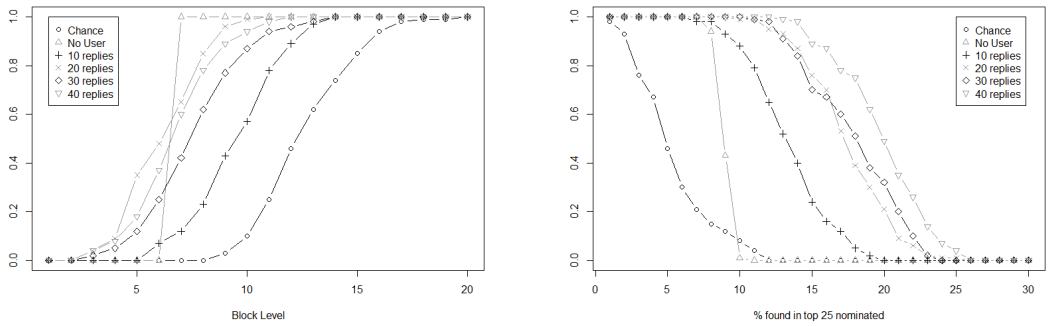


Figure 4.9: Clustering the embedded graphs using Gaussian Mixture Modeling incorporating the XYZ coordinate data and using Method 2 to estimate Δ. In the left panel, on the y-axis we plot the proportion of subjects (averaged across all blocks considered as the block-of-interest in the left hemisphere) that ranked the correct block in the right hemisphere the at worst x-th place (x-axis). In the right panel, on the y-axis we plot the the proportion of subjects (averaged across all blocks considered as the block-of-interest in the left hemisphere) for which we find the total proportion of the block of interest in the top 25 vertices nominated (x-axis).

Subgraph nomination on the Reddit social network

Moderation of social networks is another area where the biggest limitation is the human resources required to analyze the prohibitively large amount of data. One problem that appears in Reddit, is finding subreddits for which the name has changed. This can be needed to track a previously banned subreddit whose users create new accounts and a new subreddit. Further, since this experiment depends only on structure, which is informed by user interaction patterns, it could be used for recommender systems, or finding suspicious behavior in other subreddits. The first is under the assumptions that people who communicate in the same way may enjoy similar content. The second can be done by giving examples of subreddits that show suspicious behavior and looking at other subreddits that are embedded close to those subreddits. The idea behind the automation here is to distill the data so an expert moderator can take a look at the smaller data.

Data were collected by [14]. The data includes information about subreddits (S), their threads (T), comments (C) on these threads, users (U) that posted the thread or comment,

and flairs (F) associated with the subreddits. These form our node types; the focus here is on structure, so we do not maintain node types. We also dispose of the users, as they will form a leaf coming out of each comment and thread. Each of these comes with a multitude of features, notably parent comment/thread, and containing subreddit. All features are discarded for a structure based approach, however we use the relations above to create the graph. Our edges represent $S \leftrightarrow T$, $T \leftrightarrow C$ if T is not parent of C, else $T \leftarrow C$, $C_1 \leftarrow C_2$ if C_1 is parent of C_2, and $S \leftrightarrow F$. A graph is constructed for the dataset, then the subgraph for each subreddit is retrieved. Using the post-time date data each month can be subset to 4 week time periods. The data is subset to subreddits of size 1000 to 2000 nodes.

The graph matching network of [61], a graph neural network approach for comparison at the graph level, is used as the graph similarity (or dissimilarity) function. The neural network embeds graphs into a space where the closer two graphs are to one another, the more similar they are, the normalized embeddings provide a similarity metric. At this point, usually the top recommendations h are passed on to an expert to analyze. The expert would analyze the subreddits to make a decision. Here, we can insert a user in the loop, and the user would consider a node (or small collection of nodes) in the graph instead of the entire graph. Here, where we are looking to recover the subreddits in time, the node-level user defines a correct match if the node is in the subreddit of interest. Training is done on the first 2 weeks of the reddit data, and the remaining two weeks are used to test the neural network's performance and implement the user. Note that the user input can be used as a node feature and implemented during the training as well; we do not pursue that here.

After the training is complete and the tests are performed, the top 8 subgraphs are nominated, and we simulate VN users-in-the-loop with different values for θ and γ. The result is displayed in Figure 4.10, where the x-axis is the number of GMN training steps and the y-axis represents the retrieval accuracy. Accuracy here is the percentage of subgraphs for which the correct community is nominated in the top ℓ graphs. Denoting the user U_8

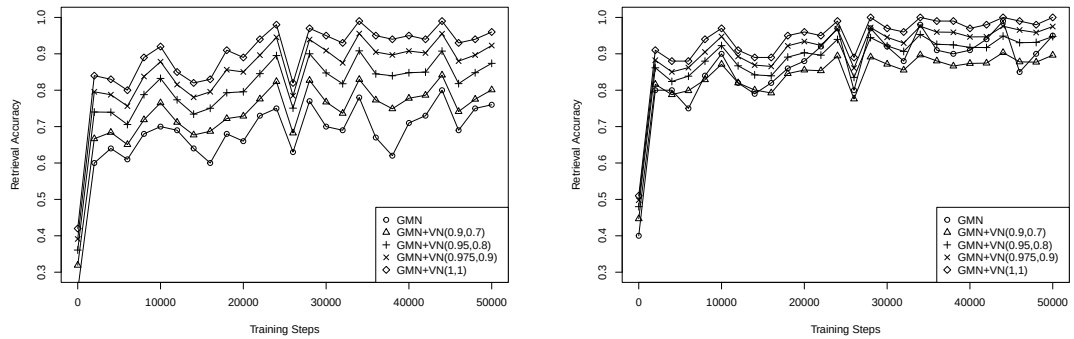


Figure 4.10: Clustering the graphs using their subreddit membership and using GMN to estimate Δ. The y-axis we plot the proportion of subreddits (averaged across 100 runs of the VN user) that ranked the correct subreddit in the second time period x-axis is the training time . In the left panel, we look at the top 2 nominations. In the right panel, we look at the top 8 nominations.

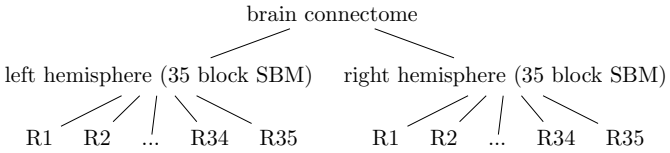
with parameters (θ, γ) with $VN(\theta, 1-\gamma)$ (recalling that θ and $1-\gamma$ are the probabilities of correct user input for true positive and true negative nodes resp.), the left plot is the accuracy for $\ell = 2$ and on the right is for $\ell = 8$.

Notice in Figure 4.10 the addition of the user-in-the-loop helps achieve better results especially at the lower level $\ell = 2$. Also, we can achieve desirable results with less training, even with a faulty VN user. The performance at $\ell = 2$ suggests the user can lower the costs of verification, as an expert user needs to look at less subgraphs to determine whether the nomination is successful. Moreover, this suggests that we can automate the VN user-in-the-loop by using node features to make decisions, and train the GMN for a shorter amount of time to reduce the computational cost. This is critical since the GMN computational cost grows quadratically with the total number of nodes in the graphs.

4.2 Testing for Repeated Motifs and Hierarchical Structure in Stochastic Blockmodels

One of the principal advantages of the repeated motif HSBM framework is that, given the structure of the motifs, the number of parameters for the HSBM model are greatly reduced. This is because all substructures equivalent to a given motif share the same block probability parameters and block membership parameters; these need only be estimated once per represented motif instead of separately over each substructure. See Figure 2.3 for example.

As discussed above, our definition of the RMHSBM is motivated by the hemisphere-level similarity in connectomes. We can model this structure via a two-level, one-motif RMHSBM. The tree structure for this RMHSBM is displayed below:



The tree is two levels, with the first level representing the hemisphere structure of the brain, and the second level the 35 regions of interest (R1, R2, …, R35) within each hemisphere, for a total of 70 leaves in the tree. The meta-communities represented by these regions of interest are modeled as simple Erdős-Rényi graphs. The hypothesized motif structural model posits that each hemisphere is an instantiation of a single, twice-repeated, 35 block SBM motif. We will be comparing *RMHSBM*s with *SBM*s, and to do this sensibly, we need to ensure that the two models are compatible; for that purpose, we give the following definition:

Definition 4.2.1 ($\tau - \mathscr{T}$ Compatibility)**.** *A node assignment function $\tau : [n] \to [K]$, a traversal function $\mathscr{T}$ for a tree T with K leaves are compatible when $\mathscr{T}(i) = \mathscr{T}(j)$ if and only if*

$\tau_i = \tau_j$.

In the case of the brain connectome, for example, this means that $\tau : [n] \to [70]$ and for all nodes $v \in Rk$ on the left hemisphere according to $\mathscr{T}$, then $\tau(v) = k$, and for all nodes $v \in Rk$ on the right hemisphere according to $\mathscr{T}$, then $\tau(v) = 35 + k$.

4.2.1 Main Results

Consider the problem of testing whether a graph G comes from a K^* block SBM (with block probability matrix $B^{(1)}$ and known block membership function τ) or from the model

$$\text{RMHSBM}(n, K, B, \{S_i\}_{i=1}^K, T, \mathscr{T}, \tilde{L}, \mathfrak{m}, \{\mathscr{M}_i\}_{i=1}^M),$$

where T has K^* leaves, and the traversal $\mathscr{T}$ is compatibe with τA. From this tree, traversal and motif structure, the RMHSBM model has a (potentially) reduced set of parameters versus the fully general SBM. For example, all structures in the HSBM corresponding to a single motif share a common set of parameters, and all inter-block SBM connectivity parameters across a layer in the RMHSBM are merged in the RMHSBM as dictated by the flat connectivity across layers in the HSBM. Let the parameter set given by the tree, traversal and motif of the model be denoted via $\Gamma = \Gamma_{\text{RMHSBM}(T, \mathscr{T}, \{\mathscr{M}_i\}_{i=1}^M)}$, and for each parameter $\gamma \in \Gamma$, let γ_B denote the set of structures in the SBM that are stochastically identical in the RMHSBM and correspond to a single parameter in RMHSBM model space to form γ. We remind the reader that the structures merged in γ could correspond to a merging of inter-block SBM connectivity structure dictated by the flat connectivity across layers in the HSBM or the merging of structures that are part of a common motif.

We will consider two data settings in which to formulate and test our hypotheses. In both settings, we have a population of $N \geq 1$ graphs that serve as our data. We are essentially testing the goodness-of-fit of the RMHSBM model; tests for SBM goodness-of-fit

have been proposed in the literature, see for example [51, 53, 56]. In the first setting, we assume that there is no variation between block parameters in the population, and all elements of the population either follow the K^* block SBM (with fixed τ and fixed communities $\{\mathscr{B}_i\}$) or the same repeated motif HSBM

$$\text{RMHSBM}(n, K, B, \{S_i\}_{i=1}^K, T, \mathscr{T}, \tilde{L}, \mathfrak{m}, \{\mathscr{M}_i\}_{i=1}^M)$$

as outlined above. Testing between the two models can be written as follows, (where $B_\gamma^{(0)}$ denotes the probability parameter for the structures merged in γ, and A is the adjacency matrix of a graph in our (assumed i.i.d.) population):

$$\begin{aligned} H_0 &: \text{For all } \gamma \in \Gamma, \text{ for all } (\ell,k) \in \gamma_B, \\ &\quad \text{and } (v,v') \in \mathscr{B}_\ell \times \mathscr{B}_k,\ A_{v,v'} \overset{ind.}{\sim} \text{Bern}(B_\gamma^{(0)}) \\ H_1 &: \text{For all } (\ell,k) \in [k]^2 \text{ and } (v,v') \in \mathscr{B}_\ell \times \mathscr{B}_k, A_{v,v'} \overset{ind.}{\sim} \text{Bern}(B_{\ell k}^{(1)}). \end{aligned} \tag{4.3}$$

For every $\gamma \in \Gamma$, let $n_\gamma = \sum_{(l,k)\in\gamma_B} n_{lk}$ denote the effective sample size used to estimate $B_\gamma^{(0)}$, and (where $\mathscr{B}_{\ell,k} = \mathscr{B}_\ell \times \mathscr{B}_k$ if $\ell \neq k$ and $\mathscr{B}_{\ell,\ell}\binom{\mathscr{B}_\ell}{2}$)

$$\widehat{B}_\gamma^{(0)} = \frac{1}{n_\gamma} \sum_{(\ell,k)\in\gamma_B} \sum_{(v,v')\in\mathscr{B}_{\ell,k}} A_{v,v'}, \quad \text{and} \quad \widehat{B}_{lk}^{(1)} = \frac{1}{n_{\ell k}} \sum_{(v,v')\in\mathscr{B}_{\ell,k}} A_{v,v'}.$$

The log likelihood ratio for the global test of H_1 versus H_0 in the setting without variation between blocks is then given by

$$-2\lambda_T = 2 \sum_{\gamma\in\Gamma_T} \sum_{(\ell,k)\in\gamma_B} n_{\ell k} \log\left(\frac{1-B_{\ell k}^{(1)}}{1-B_\gamma^{(0)}}\right) + n_{\ell k}\hat{B}_{\ell k}^{(1)} \log\left(\frac{B_{\ell k}^{(1)}(1-B_\gamma^{(0)})}{B_\gamma^{(0)}(1-B_{\ell k}^{(1)})}\right), \tag{4.4}$$

with maximum likelihood estimator as specified in the following lemma, which is proven in Appendix A.3.

Lemma 4.1. *Under the hypotheses in Equation* (4.3)*, the maximum likelihood estimator for* $B^{(1)}_{\ell k}$ *is* $\widehat{B}^{(1)}_{\ell k}$ *and for* $B^{(0)}_{\gamma}$ *is* $\widehat{B}^{(0)}_{\gamma}$.

In this case, a consistent (and extremely powerful; see Section 4.2.2) test for the above hypotheses can be constructed using the MLEs and Wilk's theorem. This test, even at proper level α, is too sensitive in practice; note that this (over) sensitivity of global network testing has been noted before, see, for example, the two-sample mesoscale testing work of [68]. Indeed, it (rightly!) asymptotically rejects the null for any small fixed deviation across any merged blocks, and is unable to identify cases where there is significant repeated structure within the model.

The global nature of the test also renders it inappropriate in the case where the SBM structures merged in the RMHSBM have small variations in their parameters that are centered around a common mean. Indeed, often perfect HSBM structure is not present practically in observed networks, as there will be small deviations within pieces of the hierarchy. For such a graph, with tree structure given by Γ, we will define the signal-to-noise ratio of the hierarchical structure via

$$\mathfrak{s}_{\Gamma} := \max_{\gamma \in \Gamma} \frac{\frac{1}{n_\gamma} \sum_{\{\ell,k\} \in \gamma_B} \mathscr{E}(A, \mathscr{B}_\ell, \mathscr{B}_k)}{\sum_{\{\ell,k\} \in \gamma_B} \sum_{(v,v') \in \mathscr{B}_l \times \mathscr{B}_k} \mathbb{E}\left(A_{v,v'} - \frac{1}{n_\gamma} \sum_{\{\ell,k\} \in \gamma_B} \mathscr{E}(A, \mathscr{B}_\ell, \mathscr{B}_k)\right)^2},$$

where for a matrix A we have $\mathscr{E}(A, \mathscr{B}_\ell, \mathscr{B}_k) = \sum_{(v,v') \in \mathscr{B}_l \times \mathscr{B}_k} \mathbb{E}(A_{v,v'})$. Note that, as we are assuming block structure, it is implicit that for each $\{\ell,k\}$, $\mathbb{E}(A_{v,v'})$ is constant for all $(v,v') \in \mathscr{B}_l \times \mathscr{B}_k$. For each $\varepsilon > 0$, we can then test if the HSBM signal strength exceeds a threshold of $\varepsilon > 0$ by testing the following hypotheses.

$$\begin{aligned} H_0^{(\varepsilon)} &: \mathfrak{s}_{\Gamma}^{-1} < \varepsilon; \\ H_1 &: \text{ For all } (l,k) \in [k]^2 \text{ and } (v,v') \in \mathscr{B}_l \times \mathscr{B}_k, E(v,v') \sim \text{Bern}(B^{(1)}_{lk}). \end{aligned} \tag{4.5}$$

Practically, we can realize a model satisfying the signal-to-noise condition $\mathfrak{s}_\Gamma^{-1} < \varepsilon$ as follows. We can have that for all $\gamma \in \Gamma$, for all $\{\ell,k\} \in \gamma_B$, and $(v,v') \in \mathscr{B}_l \times \mathscr{B}_k$, we have $A_{v,v'} \sim \text{Bern}(B_\gamma^{(0)} + s_{\ell,k})$, where the $s_{\ell,k}$'s are suitably small.

Testing in the setting with variations between blocks is useful in our motivating neuroscience application, as it can test whether there is symmetry in the distribution of connectivity within the brain while encompassing individual differences and variations between left and right hemispheres due to environmental effects rather than parametric differences.

The log-likelihood ratio in the setting with variations between blocks from (4.5) takes the form

$$\begin{aligned} -2\lambda_T = 2 \sum_{\gamma \in \Gamma_T} \sum_{\{l,k\} \in \gamma_B} \Bigg[& n_{lk} \log \frac{1 - B_{lk}^{(1)}}{1 - B_\gamma^{(0)} - s_{lk}} \\ & + n_{lk} \widehat{B}_{lk}^{(1)} \log \frac{B_{lk}^{(1)} (1 - B_\gamma^{(0)} - s_{lk})}{(B_\gamma^{(0)} + s_{lk})(1 - B_{lk}^{(1)})} \Bigg], \end{aligned} \tag{4.6}$$

The minimizer of this likelihood ratio function is not unique, as there is interdependence between the estimators of $\widehat{B}_\gamma^{(0)}$ and $\{\hat{s}_{lk}\}_{l,k \in \gamma_b}$ for each $\gamma \in \Gamma_T$. One may use numerical methods with an added condition to find a solution; e.g. L_p regularization.

Likelihood Ratio Testing

When the data come from the setting with variations between blocks, a naïve use of the likelihood ratio test based on the modeling assumptions in Equation (4.3) can be shown to be problematic. This is summarized in the following theorem. A detailed proof can be found in Appendix A.4):

Theorem 4.2. *Let $\mathbb{P}_{0,S}$ indicate the probability under the null hypothesis in Equation* (4.5)*, where these $\{s_{ij}\}$ further satisfy*

$$\varepsilon_S = \min_{\gamma \in \Gamma} \min_{(i,j) \in \gamma_B} \left| s_{ij} - \frac{1}{|\gamma_B|} \sum_{(\ell,h) \in \gamma_B} s_{\ell h} \right| > 0. \tag{4.7}$$

Let

$$\widehat{\lambda}_\gamma = -\sum_{(l,k)\in\gamma_B} n_{lk}\log\left(\frac{1-\widehat{B}^{(1)}_{lk}}{1-\widehat{B}^{(0)}_{\gamma}}\right) + n_{lk}\hat{B}^{(1)}_{lk}\log\left(\frac{\widehat{B}^{(1)}_{lk}(1-\widehat{B}^{(0)}_{\gamma})}{\widehat{B}^{(0)}_{\gamma}(1-\widehat{B}^{(1)}_{lk})}\right)$$

We then have that $\mathbb{P}_{0,S}\left(\frac{\varepsilon_S^2 n_\gamma}{9} \le -2\widehat{\lambda}_\gamma \le \frac{8n_\gamma}{\delta}\right) \ge 1 - C(\varepsilon_S, \{n_{lk}\}, \delta)$*, where*

$$C(\varepsilon_S, \{n_{lk}\}, \delta) = 2e^{-2(\min(\varepsilon_S/3, \delta/2))^2 n_\gamma} + 2\sum_{(l,k)\in\gamma_B} e^{-2(\min(\varepsilon_S/3,\delta/2))^2 n_{lk}}$$

Note that the assumption on ε_S holds almost surely, as the s_{ij} are assumed continuous, and the number of blocks in the SBM model K^* is assumed fixed.

From this theorem, we see that the approximation of likelihood ratio testing based on Wilk's theorem is inappropriate. Wilk's theorem posits that the testing criteria are based on a fixed value that depends only on the difference in the number of parameters or the degrees of freedom. Theorem 4.2 shows that the LLR statistic here grows as the number of nodes grow.

Alternatively, if $N > 1$, there is hope that averaging over the population will mitigate the effect of the $s_{\ell k}$ terms, and render classical LLR asymptotics again applicable. As the $s_{\ell k}$ are mean zero, we expect that an average of N such s's to be of order $O(\sqrt{\log(N)/N})$ with high probability, and hence ε_S would be at most of this order as well. Note however that $N/log(N)$ would need to be of order n_γ for the probability lower bound in Theorem 4.2 to not converge to 1, and thus $-2\hat{\lambda}_T$ does not grow with n_γ with high probability. However, $n_\gamma = \Theta(n^2)$ for a network of size n. This makes this requirement prohibitive in practice, as it would require as many networks as there are nodes.

Penalized Model Selection and Comparison

In this section, we will consider the problem of choosing between a collection of nested hierarchical SBMs using BIC model selection criteria [25, 95]. BIC and its variants have proven to be effective tools for model selection in the SBM setting (see, for example, [48, 113, 114]). By nested models, we mean that the hierarchical structure of the one model is a refinement of the hierarchical structure of another. As an example, see Figure 2.3, in which case the middle 2-level HSBM model is a refinement of the right 3-level HSBM model, and the left SBM model a refinement of the middle model; in essence, model $\mathscr{M}_1$ being a refinement of model $\mathscr{M}_2$ implies that

i. The partition of vertices into blocks provided by $\mathscr{M}_1$ is a refinement for the partition provided by $\mathscr{M}_2$;

ii. The set of motifs of $\mathscr{M}_2$ is a subset of the set of motifs in $\mathscr{M}_1$

iii. The metablock mapping functions $\mathfrak{m}_1$ and $\mathfrak{m}_2$ are such that if $\mathfrak{m}_1$ maps a metablock to a motif that is present in both models, $\mathfrak{m}_2$ must also map the same metablock to that motif; this is the case with the center panel being a refinement of the right panel in Figure 2.3.

This nestedness assumption leads to a similar merging structure as considered in [113], and our theoretical results below are of a similar nature. We can use the BIC formulation below to compare any pair of these three network models, but for ease of exposition, we provide theory for comparing the SBM on the left to one of the HSBM models.

A key assumption below is that we are in the setting of one of the the hypotheses in Equation (4.3), so that under the RMHSBM model assumption (i.e., under H_0), we observe data from

$$\text{RMHSBM}(n, K, B, \{S_i\}_{i=1}^K, T, \mathscr{T}, \tilde{L}, \mathfrak{m}, \{\mathscr{M}_i\}_{i=1}^M)$$

that assumes that the tree structure, traversal, and motif assignments are known a priori. Under the SBM model assumption (i.e., under H_1), we are assuming that the block membership function τ is known a priori and is compatible with $T, \mathscr{T}$, and $\mathfrak{m}$. This avoids the combinatorial unpleasantness (and intractability) that arises when we consider all possible RMHSBM structures possible for a given network.

We now proceed to show the consistency of the BIC penalized likelihood under these assumptions. With notation as in Section 4.2.1, the log-likelihood ratio test statistic (with likelihood maximizing plug-in estimators) for the unknown block probabilities can then be written as

$$-2\widehat{\lambda}_T = 2\sum_{\gamma\in\Gamma}\sum_{\{\ell,k\}\in\gamma_B} n_{\ell,k}\left[\widehat{B}^{(1)}_{\ell,k}\log\left(\frac{\widehat{B}^{(1)}_{\ell,k}(1-\widehat{B}^{(0)}_{\gamma})}{\widehat{B}^{(0)}_{\gamma}(1-\widehat{B}^{(1)}_{\ell,k})}\right)+\log\left(\frac{1-\widehat{B}^{(1)}_{\ell,k}}{1-\widehat{B}^{(0)}_{\gamma}}\right)\right] \tag{4.8}$$

where $\hat{P}^{(1)} = \hat{P}^{(1)}_{\ell,k}$ denotes a $\mathrm{Bern}(\widehat{B}^{(1)}_{\ell,k})$ measure and $\hat{P}^{(0)} = \hat{P}^{(0)}_{\gamma}$ denotes a $\mathrm{Bern}(\widehat{B}^{(0)}_{\gamma})$ measure. Note that $-2\widehat{\lambda}_T = D_{\mathrm{KL}}(\hat{P}^{(1)}_{\ell,k}\|\hat{P}^{(0)}_{\gamma})$ and hence is nonnegative. In this conditional setting (conditioning on the the tree structure, traversal, and motif assignments being given a priori), the standard BIC penalties applied block-wise to the estimated likelihood ratio statistic is defined by, letting $\hat{L}_{0,T}$ be the maximum likelihood under H_0 and $\hat{L}_{1,\tau}$ under H_1,

$$\widehat{\mathrm{BIC}}_{0,T} = -2\log(\hat{L}_{0,T}) + |\Gamma|\binom{n}{2} \quad \text{and} \quad \widehat{\mathrm{BIC}}_{1,\tau} = -2\log(\hat{L}_{1,\tau}) + \sum_{\gamma\in\Gamma}|\gamma_B|\binom{n}{2}$$

and hence

$$\widehat{\Delta}_{T,\mathrm{BIC}} = \widehat{\mathrm{BIC}}_{0,T} - \widehat{\mathrm{BIC}}_{1,\tau} = -2\widehat{\lambda}_T - \left(\sum_{\gamma\in\Gamma}(|\gamma_B|-1)\right)\log\binom{n}{2}. \tag{4.9}$$

Below we state a pair of theorems that delineate the asymptotic behavior of $\widehat{\Delta}_{T,\mathrm{BIC}}$ under H_0 and H_1. In the theorems below, we make use of the following assumption on the

growth rate of $B^{(1)}$:

Assumption 1. *We assume that there exists an $n_0 \in \mathbb{Z} > 0$ and constant $c_1 \in (0, 1/2)$ such that for all $n > n_0$, we have that the following holds for all $\{\ell, k\}$ pairs:*

$$B^{(1)}_{\ell,k} > \frac{\log n_{\ell,k}}{n_{\ell,k}}, \quad \textit{and} \quad B^{(1)}_{\ell,k} < 1/2 - c_1. \tag{4.10}$$

Our first result states that under H_0, the penalized BIC difference is negative and the true RMHSBM model is preferred. A proof can be found in Section A.5 of the Appendix.

Theorem 4.3. *With notation as above, given Assumption 1, let*

$$c_2 \leq \frac{1}{16}\left(1 - \frac{2|\Gamma|}{(K^*)^2 + K^*}\right)$$

be a constant. Then we have that for all n sufficiently large,

$$\mathbb{P}_0\left[\widehat{\Delta}_{T,\mathrm{BIC}} < 0\right] \geq 1 - O\left(\exp\left\{-((K^*)^2 + K^*) - \frac{c_2 \log n_{**}}{2 + 2\sqrt{c_2}/3}\right\}\right)$$

where $n_{**} = \min_{\ell,k} n_{\ell,k}$.

Let us next consider the case in which the RMHSBM is misspecified. The next definition quantifies the level of model incompatibility. Recall that under H_1, the model is an SBM$(K^*, B^{(1)}, \tau)$ (where τ is assumed compatible with $\mathscr{T}$).

Definition 4.2.2. *Under H_1, we say that the RMHSBM model with block structure given by Γ is (M, η)-incompatible with the hypothesized SBM (from H_1) model if $|\mathbb{E}\widehat{B}^{(0)}_{\gamma} - B^{(1)}_{\ell,k}| > \eta$ for at least M triplets $\{\ell, k, \gamma\}$ such that $\{\ell, k\} \in \gamma_B$.*

Our next result states that under H_1 (when the SBM is the true model and not the HSBM), the penalized BIC difference is positive and the true SBM model is preferred; we then have

Theorem 4.4. *With notation as above and given Assumption 1, assume further that the RMHSBM is (M,η)-incompatible with the true SBM, where*

$$\eta^2 M = \omega((K^*)^2 n_{**}^{-1} \log n.$$

Then for any $c_3 > 0$, we have that for all n sufficiently large, it holds that

$$\mathbb{P}_1(\widehat{\Delta}_{T,\mathrm{BIC}} > 0) \geq 1 - 4(K^*)^2 \exp\left\{-\frac{c_3 \log n_{**}}{2 + 2\sqrt{c_3}/3}\right\},$$

where $n_{**} = \min_{\ell,k} n_{\ell,k}$.

Note that a proof of Theorem 4.4 is given in Section A.6 of the Appendix.

4.2.2 Experiments

At first, we consider a global log likelihood ratio test. As we have showed above, if one of the parameters we are testing for similarity fails, we would reject the hypothesis that there is similarity between the communities for sufficiently large n. This holds since the log likelihood ratio grows with a rate of $O(n_{**}^2)$ where the lead coefficient depends on the parameters. In light of this, we propose using conducting multiple hypotheses at the motif level, using the Benjamini-Hochberg correction [15] to control False Discovery Rate. In both cases, we will consider three methods for retrieving the null probabilities: classic likelihood ration testing (implemented via Wilk's theorem), Friedman signed rank for a nonparametric alternative, and ANOVA for testing population versus individual variability. We next proceed to expound upon these methods.

These different methods consider different levels of granularity in the null hypotheses as well. In the case of Wilk's χ^2, we can consider two levels of null granularity. More precisely, suppose we have S graphs, $\{A_s\}_{s=1}^S$ with set parameter matrices $\{B_s = [B_{\ell,k;s}]\}_{s=1}^S$;

we then consider

i. Individual level testing: the null hypotheses can be written (where $n_{\ell,k;s}$ denotes the number of possible edges between community ℓ and k in graph s),

$$H_0 : \begin{cases} \text{For all } \gamma \in \Gamma \\ \text{and all } (\ell,k),(\ell',k') \in \gamma_B,\, B_{\ell,k;s} = B_{\ell',k';s} \end{cases} \quad \text{(global test)}$$

$$\left\{ H_0^{(\gamma)} :\ \text{For all } (\ell,k),(\ell',k') \in \gamma_B,\, B_{\ell,k;s} = B_{\ell',k';s} \right\}_{\gamma\in\Gamma} \quad \text{(local tests)}$$

These hypotheses test within each graph at the individual level of the S graphs;

ii. Aggregated testing: let $\tilde{N}_{\ell,k} = \sum_{s=1}^{S} n_{\ell,k;s}$ and $\tilde{B}_{\ell,k} = \sum_{s=1}^{S} \frac{n_{l,k;s}}{\tilde{N}_{\ell,k}} B_{\ell,k;s}$, then the null hypotheses can be written,

$$H_0 :\ \text{For all } \gamma \in \Gamma \text{ and all } (\ell,k),(\ell',k') \in \gamma_B,\, \tilde{B}_{\ell,k} = \tilde{B}_{\ell',k'} \quad \text{(global test)}$$

$$\left\{ H_0^{(\gamma)} :\ \text{For all } (\ell,k),(\ell',k') \in \gamma_B,\, \tilde{B}_{\ell,k} = \tilde{B}_{\ell',k'} \right\}_{\gamma\in\Gamma} \quad \text{(local tests)}$$

Note that this means that the parameters can be different across individuals in the population as long as these weighted sums are not statistically significantly different.

Meanwhile, the ANOVA test puts an additional condition that the parameters across individuals are the same, replacing the set of parameter matrices above with a single matrix B. The nulls can then be written as (note for ANOVA and Friedman, we only test locally; the global tests exhibited the same trend as the χ^2)

$$\left\{ \begin{array}{l} B_{\ell,k;s} = B_{\ell,k;s'} \text{ for all } s,s' \in [S], \\ H_0^{(\gamma)} :\ B_{\ell,k;s} = B_{\ell',k';s} \text{ for all } (\ell,k),(\ell',k') \in \gamma_B, \\ \mathrm{Var} B_{\ell,k;s} = \sigma_\gamma^2 \text{ for all } s \in [S] \text{ and all } (\ell,k),(\ell',k') \in \gamma_B. \end{array} \right\}_{\gamma\in\Gamma} \quad \text{(local tests)}$$

The Friedman test meanwhile, as the data are Bernoulli, tests the same null hypotheses as the ANOVA case with the caveat that independence of the entries of B_s is no longer assumed.

Simulations

We do the simulations over two models, one to simulate the brain data from [42, 122] (obtained via `https://fcon_1000.projects.nitrc.org/indi/CoRR/html/bnu_1.html` through `https://neurodata.io/mri/`), the other a 3-motif RMHSBM. Additional details about these models is in Table 4.1. In each simulation the parameters are drawn independently from a Dirichlet distribution. In each we corrupt the model by changing some of the true parameters to be different from each other at random (injecting different levels of error into the alternative models); these changed parameters are drawn from the same Dirichlet distribution independently of the remainder of the parameters; an example model matrix for this in the BNU1-inspired simulation setting can be found in Figure 4.12 and for the 3-motif, 7-block model in Figure 4.13. We repeat this twice, first, without any individual differences (as in Eq. 4.3), then by adding a deviations to all parameters to represent individual differences (as in Eq. 4.5). The small deviations are normally distributed and centered about the true parameters, with a variance dependent on the magnitude of the true parameters, on average these differences have a magnitude of around 1% of the true parameters. We draw the parameters once for all samples. In all cases the parameters had a lower bound cut-off of 0.01 and an upper bound cut-off of 0.99 to avoid boundary unpleasantries.

In Figure 4.11, the right axes along with the colored line plots show the average rejection rate over varying numbers of changed parameters for the global χ^2 test of the BNU1 and the 3-motif, 7-communities error-less simulations, respectively; the left axes along with the ribbon plots show the penalized likelihood ratio from equation 4.9. The results

Characteristic	BNU1 Simulation	3 motif RMHSBM
Nodes per block	200	200
Number of blocks	70	70
motifs	1 (rep. twice)	3 (1 rep. 3 times, 2 & 3 rep. twice)
parameters	631	195

Table 4.1: Model specifications for the two RMHSBM models we test.

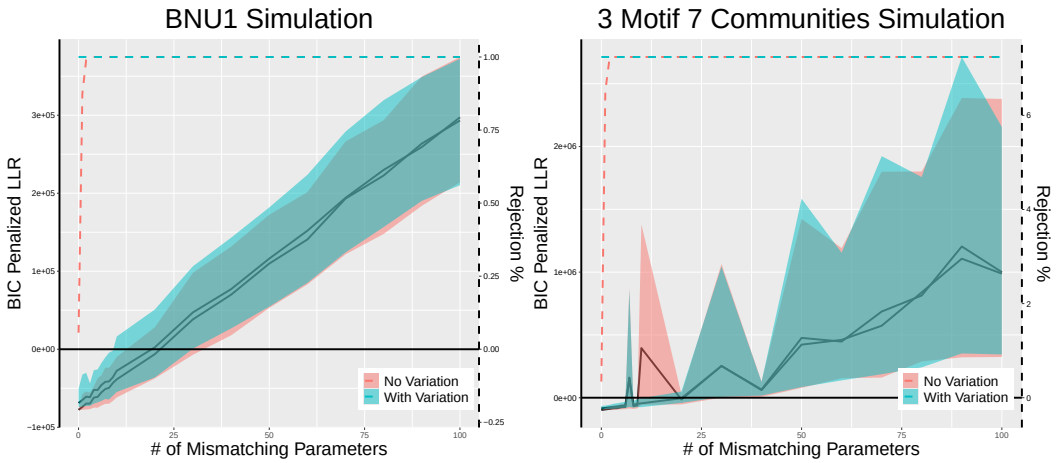


Figure 4.11: In each panel: the left axis (and the plots with error bands) shows the penalized log likelihood ratio based on equation 4.9 while the right axis (and the dashed line plots) show the rejection rate of the global log likelihood ratio test. The right panel shows the BNU1-based simulation, the left panel the 3 motif-7 communities simulation.

are averaged over 200 simulations (20 for each of the 200 different parameterizations). For each, we plot both the no variation (red) and small variation (blue) settings. In both panels, we see that in the no variation case this test (correctly) rejects the null hypothesis of repeated structure if we change 10 of the parameters. In the setting with small variations, the LLR-based test is inconsistent and rejects the null (based upon Wilk's Theorem) even in the case where no parameters are mismatched, as the application of the χ^2 critical value here—as is used in Wilk's theorem—is not appropriate; see Theorem 4.2. Meanwhile, we see that in both panels, the penalized LLR results indicate that the penalized model still selects for repeated structure even when we have around 5% mismatch in the parameters; and that the likelihood ratio test fails to detect similarities in the repeated motif structure even in the case where these few parameters are mismatched. This shows the weakness of global testing based on the likelihood ratio, aligning with the theory in which we have shown that these small errors are scaled by a factor of n^2 in the test, rendering this test very sensitive to small changes.

The next approach is to test individual motifs separately, with a Benjamini-Hochberg correction, using the methods discussed above. In Figure 4.14, we see the results of the averaged χ^2 individual testing, the aggregated χ^2 test, ANOVA, and Friedman's signed rank test from left to right, respectively. Note that this figure considers the model that does not account for variations in the parameters across individuals. In the first row are the rejection matrices showing correctly rejected blocks in black, incorrectly rejected blocks in blue, and incorrectly accepted blocks in red; note that the aggregated methods are more resilient to incorrectly accepting blocks, as indicated by the absence of red in this example. Meanwhile, individual testing is more susceptible to these errors. Moreover, the aggregated tests are susceptible to incorrect rejection of blocks, as indicated by the blue squares. The averaged individual testing, on the other hand, is not free of these errors, but with the correct threshold one may circumvent these errors; the method for selecting this threshold

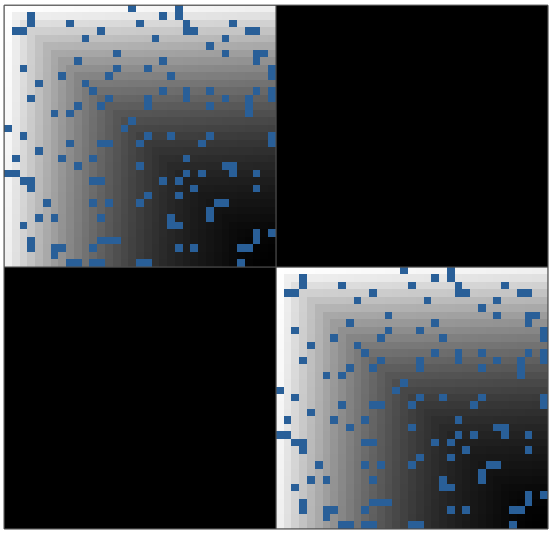


Figure 4.12: Example model matrices for the BNU1 simulation, 70 blocks, where the first 35 are similar to the second 35 with one cross community connection parameter between them. Blue squares indicate the parameters we set to be different from their supposed pair, while the remaining shades of grey indicate similarity.

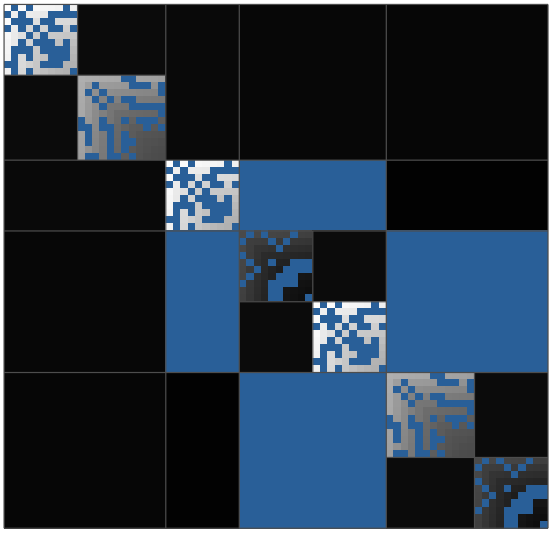


Figure 4.13: Example model matrix for the 3-motif, 7-communities simulation, 70 blocks, where the block groups $(1,3,6),(2,6),(4,7)$ share a motif. Blue squares indicate the parameters we set to be different from their supposed pair, while the remaining shades of grey indicate similarity.

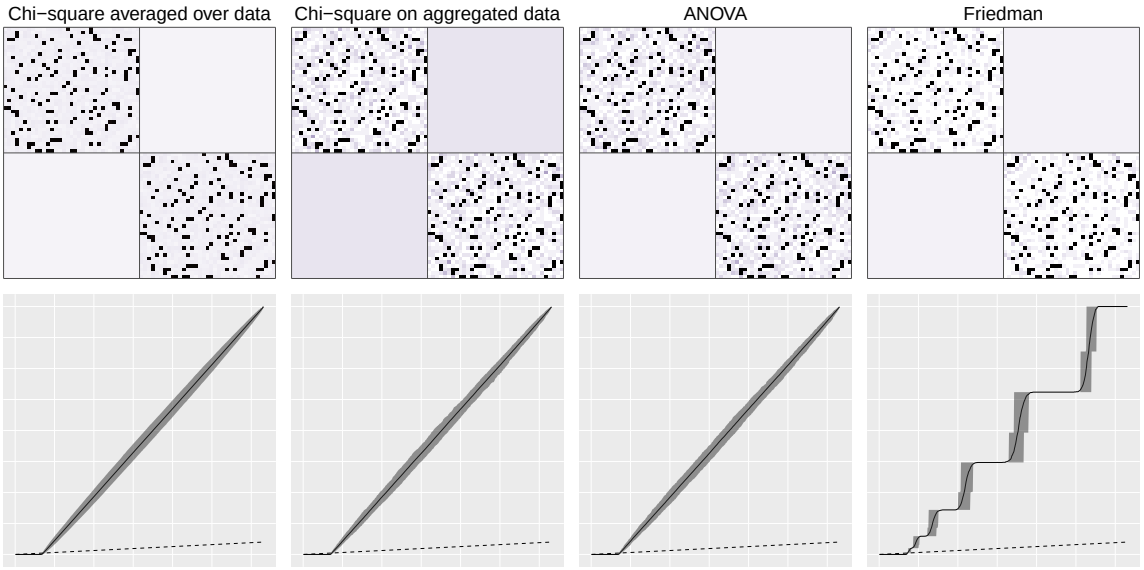


Figure 4.14: On the top row, are the rejection matrices for (from left to right) the averaged individual χ^2 tests, the aggregated χ^2 test, ANOVA test, and Friedman's signed rank test on the error-less BNU1 simulation.In black are the correctly rejected blocks, in blue are the incorrectly rejected blocks (Type II error), in white are the blocks that correctly failed to reject, and in red are the blocks are incorrectly failed to reject (Type I error). On the bottom row are the P-value profile for each method respectively, the solid line represents the P values of each block in increasing order, the dashed line represents the Benjamini-Hochberg rejection line. Blue colors are exaggerated to make the comparison clear, all models similarly well.

is not clear however. Keep in mind that the darkness of the blue squares is not linearly indicative of the rejection rate; instead it remapped for better visibility. On the second row, we see the profile of the p-values in increasing order; we refer to this as the p-profile going forward. The p-profiles are all linear, indicating a good fit for these tests. In Figure 4.15, we see similar results when testing for the 3-motif, 7-communities model.

Moving on to modeling small variations in the sample, we see a different picture. In Figures 4.16 and 4.17, we see the results of adding small variations to the BNU1 and 3-motif, 7-communities models, respectively. In the presence of small variations, the flaws in both χ^2 tests become apparent. False rejections become very common as indicated by the large amount of blue squares, compared to the ANOVA and Friedman's signed rank

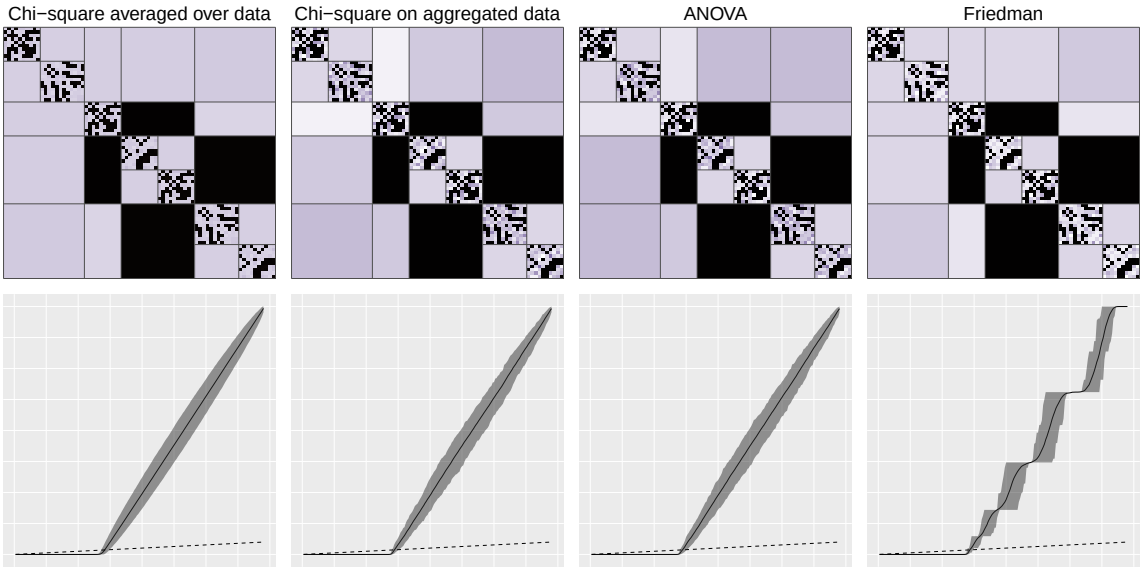


Figure 4.15: On the top row, are the rejection matrices for (from left to right) the averaged individual χ^2 tests, the aggregated χ^2 test, ANOVA test, and Friedman's signed rank test on the error-less 3-motif, 7-communities simulation. In black are the correctly rejected blocks, in blue are the incorrectly rejected blocks (Type II error), in white are the blocks that correctly failed to reject, and in red are the blocks are incorrectly failed to reject (Type I error). On the bottom row are the P-value profile for each method respectively, the solid line represents the P values of each block in increasing order, the dashed line represents the Benjamini-Hochberg rejection line. Blue colors are exaggerated to make the comparison clear, all models similarly well.

tests, respectively. The p-profiles of the χ^2 are not linear indicating that the modeling assumptions are not a good fit in this case. ANOVA and Friedman signed rank tests have a linear p-profile indicating the resilience of their modeling assumptions to small variations. These small variations simulate individual differences across the population.

Application: Testing for Community Structure in The Brain.

Identifying the regions of the brain that share a distributional similarity can help identify regions that have similar functionality or those that are independent. This analysis is performed on the BNU1 data set [42, 122], consisting of 57 fMRI scans of the human brain of adults at rest. After dividing the brain into spatial regions commonly referred to

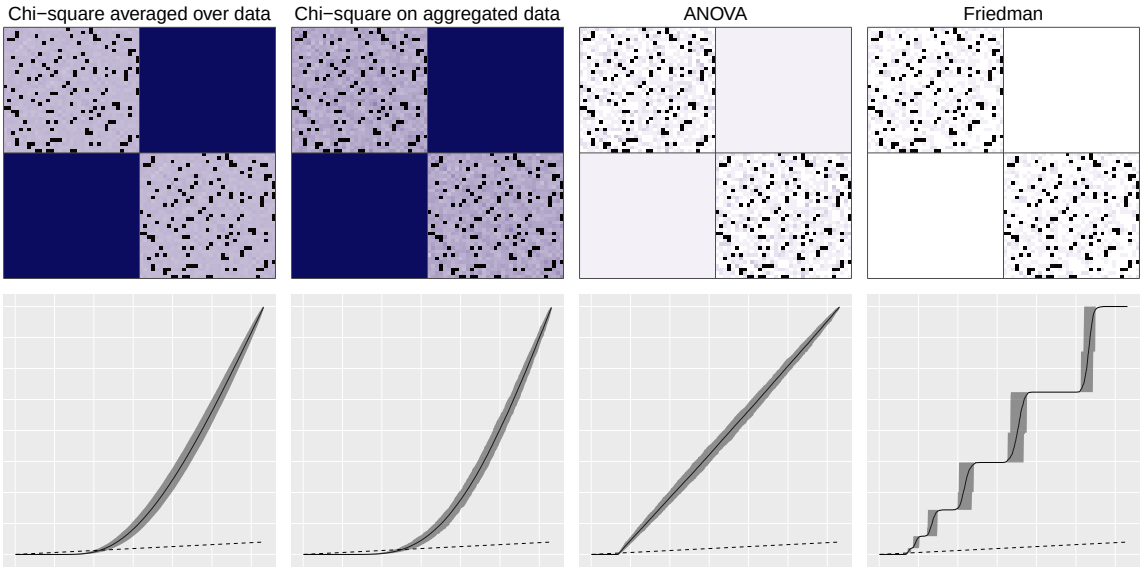


Figure 4.16: On the top row, are the rejection matrices for (from left to right) the averaged individual χ^2 tests, the aggregated χ^2 test, ANOVA test, and Friedman's signed rank test on the error-full BNU1 simulation.In black are the correctly rejected blocks, in blue are the incorrectly rejected blocks (Type II error), in white are the blocks that correctly failed to reject, and in red are the blocks are incorrectly failed to reject (Type I error). On the bottom row are the P-value profile for each method respectively, the solid line represents the P values of each block in increasing order, the dashed line represents the Benjamini-Hochberg rejection line.

as voxels, fMRI scans measure changes in blood flows. These changes are interpreted as connections between the voxels that serve as the nodes, producing a graph. The data classifies these nodes into regions that represent larger spatial portions of the brain, forming a natural block structure. Additionally, the voxels are classified based on which half of the brain they are located. Combining the classifications, regions and halves gives rise to a natural hierarchical structure for the brain. Table 4.2 describes the details of these graphs.

Using the posited hierarchical structure, we perform different analysis techniques to describe the possibility of repeated structure across hemispheres and to understand the potential reduction in the number of parameters needed to describe the distribution of these networks. More precisely, we attempt to answer the questions of whether there is symmetry in the edge distribution across the two halves of the brain. In Figure 4.18 we plot the result

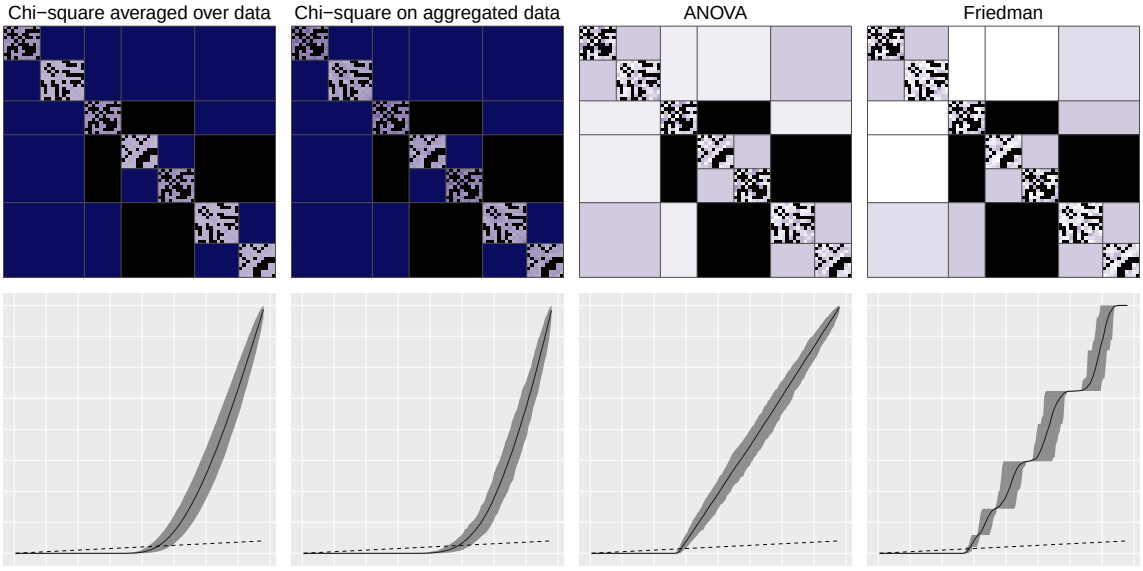


Figure 4.17: On the top row, are the rejection matrices for (from left to right) the averaged individual χ^2 tests, the aggregated χ^2 test, ANOVA test, and Friedman's signed rank test on the error-full 3-motif, 7-communities simulation. In black are the correctly rejected blocks, in blue are the incorrectly rejected blocks (Type II error), in white are the blocks that correctly failed to reject, and in red are the blocks are incorrectly failed to reject (Type I error). On the bottom row are the P-value profile for each method respectively, the solid line represents the P values of each block in increasing order, the dashed line represents the Benjamini-Hochberg rejection line.

Characteristic	Range
Number of vertices	9312-15631
Number of connections	204022-539259
Average degree	43-74
Number of blocks	70
Average block size	149-211

Table 4.2: The range of number of vertices, connections, average degree, and block sizes of the 57 individuals from the BNU1 dataset.

of our 4 procedures, in black are the rejected blocks, in white are the blocks that failed to reject, and in red are the blocks that are identically zero in both hemispheres and which trivially fail to reject. In the figure, in black are the rejected blocks, in white are the blocks that failed to reject, and in red are the blocks that are identically zero in both hemispheres which always fail to reject. On the left, the results for Wilk's χ^2-test with Benjamini-Hochberg simultaneous testing are displayed. We average over all the individuals since each individual is tested on their own. We note that many of the similarities we fail to reject are instances where both the left and right block probabilities are identically zero on both hemispheres, as indicated by the red colored blocks. From the simulations, we learned that it is difficult to draw any conclusions about the partially rejected tests.

Next, we aggregated the samples and then performed the chi-square test. The results of these tests are displayed in Figure 4.18 in the second column from the left. Once again, a large part of the tests we fail to reject correspond to parameters that are identically 0. In this case, we reject most of the ambiguous blocks from the averaged individual tests. This is in stark contrast with the ANOVA test, as we see in Figure 4.18 in the second to last column. Here, the model suggests that many of the parameters have a common mean with differences that are normally distributed.

Finally, we pair the parameters on the left and right hemispheres for each individual, then perform a Friedman signed-rank test. This is tested against a mean of zero, and the rejection of the null hypothesis would suggest that the pairs are not exchangeable. This

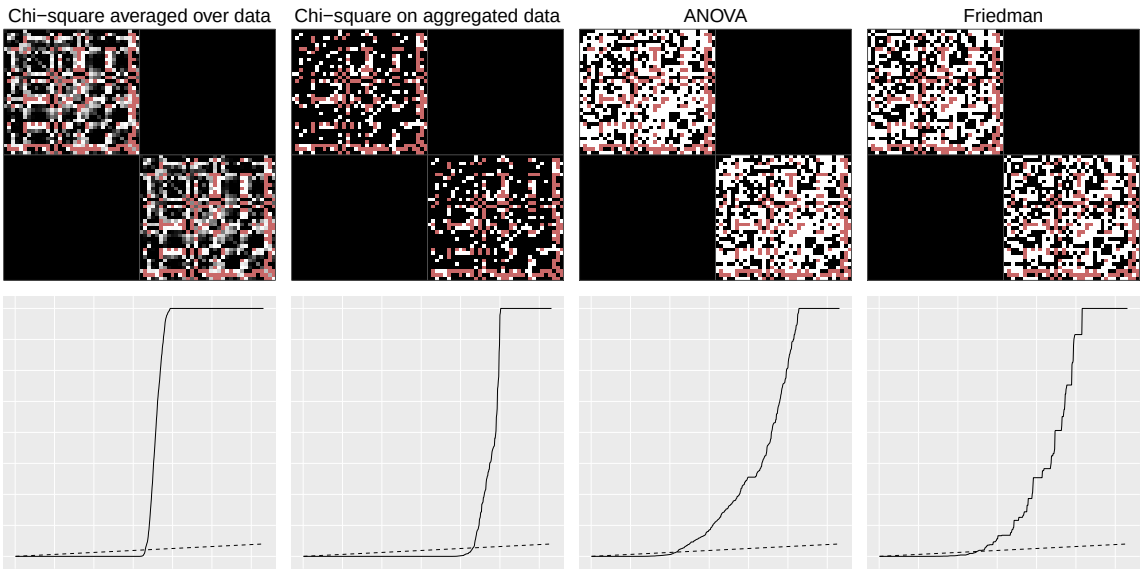


Figure 4.18: On the top row, are the rejection matrices for (from left to right) the averaged individual χ^2 tests, the aggregated χ^2 test, the ANOVA test, and Friedman's signed rank test on the BNU1 dataset. In black are the rejected blocks, in white are the blocks that failed to reject, and in red are the blocks that are identically zero in both hemispheres which always fail to reject. On the bottom row are the P-value profile for each method respectively, the solid line represents the P values of each block in increasing order, the dashed line represents the Benjamini-Hochberg rejection line.

implies (for instance) that the parameters on the left and right hemispheres do not have the same distribution. The results of this test are shown in figure 4.18 in the last column. These results largely agree with ANOVA, indicating statistically significant similarity between the left and right hemispheres with small variability across the population, with results similar to those in the simulation setting of Figure 4.16.

Chapter 5: Conclusions and open problems

We have introduced a formal and versatile framework for the subgraph nomination inference task, with special emphasis paid to the utility of users-in-the-loop in the context of subgraph nomination. This was done in the context of the RMHSBM, a model that captures hierarchical structures with similarity between the subgraphs defined by this hierarchy. Subgraph nomination is an important tool for structural queries within and across networks, and we demonstrated its utility in both real and synthetic data examples. In addition, the relatively simple users-in-the-loop formulation outlined herein can easily be lifted to more complex cases, including supervision done over several iterations, and can be simply extended to the case of multiple users. The theory and experiments included herein highlight the important role that user supervision can play in effective information retrieval. Meanwhile, structural similarity in graphs can be observed in many disciplines; the RMHSBM provides a general framework for describing graphs with similarity defined in a hierarchy. Such models can drastically reduce the number of parameters required to describe these networks, which is especially important for large networks with many small communities. Such simplification can help us better understand and estimate models for these networks. In addition, we used theory and simulations to show the difficulty of global testing in situations where there is partial similarity in the hierarchy. In this case, the sensitivity of likelihood ratio tests to small differences overwhelms the signal in a potentially repeated structure (as shown in the penalized LLR setting). Instead, we propose locally more robust methods that work to identify a repeated structure at the motif level.

Section 4.1.1 shows the importance of analyzing the original algorithm and the validity of user input before including them. For example, if the inferred clustering is incorrect in a hierarchical subgraph nomination framework, in the sense that the correct clusters' vertices are evenly spread among the inferred blocks, then our resources should be focused on collecting relevant data that would improve the inferred blocking structure, as even oracle user supervision in this case could be detrimental to the performance. In our approach to subgraph nomination, inferring high-fidelity clusters is essential, and we have demonstrated that clustering can be improved by incorporating informative vertex features. Exploring this further, understanding the information-theoretic gain of features on our nomination performance as in [59], is an open area of research that we are actively pursuing.

The validity of the user input itself is also important to consider when thinking about adding a user-in-the-loop. While we have theoretically demonstrated the possibility of increased performance even with an errorful user, in practice the situation is significantly more nuanced. The user errors, rather than being uniformly random, may be systematic or deterministic. For example an adversarial attacks could manipulate the results of the search using a user bot attack. More nuanced users-in-the-loop demand more nuanced theory to understand their broad effects. As an example, in our nomination regime an adversary can target the obtained cluster labels for contamination. This would immediately mitigate the benefit of a user-in-the-loop, as the benefit of the user decays as the cluster fidelity worsens. More robust users would be, most likely, more costly and a cost-benefit optimization analysis would be needed in these more nuanced setting to tease out the positive (or negative) impact of incorporating the user.

Another avenue that has not been investigated explicitly but for which the framework is still sufficient is the situation with multiple blocks of interest. For that we need to adjust the definition of the VN-user to accommodate for different values of p for each block of interest. Then, after some necessary adjustment to the loss function, one may obtain results

generalized to the case of multiple blocks of interest in the simple user-in-the-loop setting. On the other hand, one may want to extend the results of Theorem 4.1 to more nuanced users.

For implementing the core subgraph nomination routine, the computational burden is in graph partitioning and subgraph comparison, both of which can be implemented efficiently; e.g., any number of scalable clustering routines for graph division, and fast graph similarity algorithms (e.g., [12]). For the VN user-in-the-loop it will greatly depend on the setting, essentially we randomly retrieve an element from each of the top h communities. We then apply the user, if the user is an expert then the cost is not only computational (if the expert needs to perform some analysis on the node's feature for example) but also in human resources, which is beyond the scope of this paper, but intuitively scalability is an issue. If an automated user is used (e.g., one who is trained on the node features alongside the algorithm) then the cost is that of training and h times the cost of retrieving the model's fit on the selected nodes, here scalability is less of an issue.

In Section 4.2.2, we compared global and local methods in a simulated setting and applied local testing methods to the BNU1 data set, where we found some structural similarity between the left and right hemispheres in human brains. For naturally developing networks, confounding environmental effects can cause small perturbations that challenge the underlying symmetry. By employing a hierarchical structure, we can study these networks at multiple levels of resolution to uncover the underlying symmetry. However, even with hierarchical thinking, global testing suffers on two fronts: perturbations may lead to false rejections and failure to discover symmetries when some but not all sub-communities admit it.

We give theoretical support that likelihood ratio testing in the presence of small perturbations (that grow at a rate $\omega\left(\frac{1}{\sqrt{n}}\right)$) rejects any similarity under classical LLR testing assumptions. From simulations, we see that both ANOVA and Friedman's signed rank

tests are resilient to these perturbations, discovering the underlying symmetry in a setting where likelihood-ratio testing fails. In light of these discoveries, we apply these methods to test for symmetries between the hemispheres of human brains. We discover that when we account for these perturbations by using ANOVA or Friedman's test, there are some symmetries at the mesoscale. Also, during our work with brain imaging data, we discovered that functional regions of the brain do not admit the associative structure defined in 7 on which most clustering algorithms rely.

We used the natural clustering given by the data set, but a method that can reliably retrieve these structures and find further subdivisions of them to retrieve a hierarchical structure would allow for this testing to be performed on multiple levels to explore the degree to which there is symmetry. An extension of this framework to mixed memberships can be considered here; if the attachment probability in a community is lower than another and a node has mixed membership, it will naturally lead to the break from associativity conditions. In addition to symmetry, we discussed how repeated structure is related to subgraph matching. In this work, the matches were already known, and we tested whether they represent a repeated motif. Further study of the extent to which a repeated motif can lead to the discovery of matches would be another direction to extend this work.

Appendix A: Detailed Proofs

A.1 Proof of Theorem 4.1

For $i \leq j \leq c$, let the event $A_{[i,j]}$ be the event that the unknown subgraph of interest in g_2 has rank in $[i, j]$. Let E_h be the event that after user-in-the-loop supervision, the subgraph of interest has rank strictly greater than h.

Part I: With an oracle user-in-the-loop, we have that

$$\begin{aligned}
\mathbb{P}(E_h) &= \underbrace{\mathbb{P}(E_h|A_{[1,h]})}_{=0}\mathbb{P}(A_{[1,h]}) \\
&\quad + \underbrace{\mathbb{P}(E_h|A_{[h+1,h+t]})}_{=0}\mathbb{P}(A_{[h+1,h+t]}) \\
&\quad + \underbrace{\mathbb{P}(E_h|A_{[h+t+1,m_k]})}_{=1}\mathbb{P}(A_{[h+t+1,m_k]}) \\
&= \mathbb{P}(A_{[h+t+1,m_k]})
\end{aligned}$$

If $h+t \leq c$, then $\mathbb{P}(A_{[h+t+1,k]}) = 1 - \frac{h+t}{c}$, else it is 0. Hence, $\mathbb{P}(E_h) = \max(0, 1 - \frac{h+t}{c})$ as desired.

Part II: When the user has probability p of misidentifying the vertex from the subgraph of

interest as non-interesting, then when $h+t \leq c$,

$$\begin{aligned}
\mathbb{P}(E_h) &= \underbrace{\mathbb{P}(E_h|A_{[1,t]})}_{=p}\mathbb{P}(A_{[1,t]}) \\
&\quad + \underbrace{\mathbb{P}(E_h|A_{[t+1,h+t]})}_{=0}\mathbb{P}(A_{[t+1,h+t]}) \\
&\quad + \underbrace{\mathbb{P}(E_h|A_{[h+t+1,m_k]})}_{=1}\mathbb{P}(A_{[h+t+1,m_k]}) \\
&= p\mathbb{P}(A_{[1,t]}) + \mathbb{P}(A_{[h+t+1,m_k]}) \\
&= \frac{pt}{c} + 1 - \frac{h+t}{c} = 1 - (1-p)\frac{t}{c} - \frac{h}{c}.
\end{aligned}$$

When $m_k > h+t > c$, the above probability is simply $\frac{pt}{c}$.

Part III: When the user has probability p of misidentifying the vertex from the subgraph of interest as non-interesting and probability q of misidentifying a non-interesting vertex as

interesting, then when $h+t \leq c$ and $t \leq h$,

$$
\begin{aligned}
\mathbb{P}(E_h) = & \underbrace{\mathbb{P}(E_h|A_{[1,t]})}_{=p}\mathbb{P}(A_{[1,t]}) \\
& + \underbrace{\mathbb{P}(E_h|A_{[t+1,h]})}_{=0}\mathbb{P}(A_{[t+1,h]}) \\
& + \sum_{i=1}^{t}\left[\mathbb{P}(E_h|A_{[h+i,h+i]})\underbrace{\mathbb{P}(A_{[h+i,h+i]})}_{=\frac{1}{c}}\right. \\
& \left. + \underbrace{\mathbb{P}(E_h|A_{[h+t+1,m_k]})}_{=1}\mathbb{P}(A_{[h+t+1,m_k]})\right] \\
= & \frac{pt}{c} + \sum_{i=1}^{t}\frac{1}{c}\sum_{j=0}^{i-1}\binom{t}{j}(1-q)^j q^{t-j} \\
& + 1 - \frac{h+t}{c} \\
= & \frac{pt}{c} + \sum_{j=0}^{t-1}\frac{t-j}{c}\binom{t}{j}(1-q)^j q^{t-j} \\
& + 1 - \frac{h+t}{c} \\
= & \frac{pt}{c} + \sum_{j=0}^{t}\frac{t-j}{c}\binom{t}{j}(1-q)^j q^{t-j} \\
& + 1 - \frac{h+t}{c} \\
= & \frac{pt}{c} + \frac{qt}{c} + 1 - \frac{h+t}{c}.
\end{aligned}
$$

When $h+t > c$ and $t \leq h$,

$$\begin{aligned}
\mathbb{P}(E_h) = & \underbrace{\mathbb{P}(E_h|A_{[1,t]})}_{=p}\mathbb{P}(A_{[1,t]}) \\
& + \underbrace{\mathbb{P}(E_h|A_{[t+1,h]})}_{=0}\mathbb{P}(A_{[t+1,h]}) \\
& + \sum_{i=1}^{c-h}\mathbb{P}(E_h|A_{[h+i,h+i]})\underbrace{\mathbb{P}(A_{[h+i,h+i]})}_{=\frac{1}{c}} \\
& + \mathbb{P}(E_h|A_{[c+1,m_k]})\underbrace{\mathbb{P}(A_{[c+1,m_k]})}_{=0} \\
= & \frac{pt}{c} + \frac{1}{c}\sum_{i=1}^{c-h}F(i-1;t,1-q)
\end{aligned}$$

When $t > h$, and $h+t \leq c$, then

$$
\begin{aligned}
\mathbb{P}(E_h) = & \underbrace{\mathbb{P}(E_h|A_{[1,h]})}_{=p}\mathbb{P}(A_{[1,h]}) \\
& + \sum_{i=1}^{t-h} \mathbb{P}(E_h|A_{[h+i,h+i]}) \underbrace{\mathbb{P}(A_{[h+i,h+i]})}_{=\frac{1}{c}} \\
& + \sum_{i=t-h+1}^{t} \mathbb{P}(E_h|A_{[h+i,h+i]}) \underbrace{\mathbb{P}(A_{[h+i,h+i]})}_{=\frac{1}{c}} \\
& + \underbrace{\mathbb{P}(E_h|A_{[h+t+1,m_k]})}_{=1}\mathbb{P}(A_{[h+t+1,m_k]}) \\
= & \frac{ph}{c}\sum_{i=1}^{t-h}\frac{1}{c}\Bigg(p \\
& \quad +(1-p)\sum_{j=0}^{i-1}\binom{h+i-1}{j}(1-q)^j q^{h+i-1-j}\Bigg) \\
& + \sum_{i=t-h+1}^{t}\frac{1}{c}\sum_{j=0}^{i-1}\binom{t}{j}(1-q)^j q^{t-j} + 1 - \frac{h+t}{c} \\
= & 1 - \frac{h}{c} - \frac{(1-p)t}{c} \\
& + \frac{1-p}{c}\sum_{i=1}^{t-h} F(i-1; h+i-1, 1-q) \\
& + \frac{1}{c}\sum_{i=t-h+1}^{t} F(i-1; t, 1-q)
\end{aligned}
$$

When $t > h$, and $h+t > c$, then (assuming $c \geq t, h$)

$$\begin{aligned}
\mathbb{P}(E_h) &= \underbrace{\mathbb{P}(E_h|A_{[1,h]})}_{=p}\mathbb{P}(A_{[1,h]}) \\
&\quad + \sum_{i=1}^{t-h} \mathbb{P}(E_h|A_{[h+i,h+i]}) \underbrace{\mathbb{P}(A_{[h+i,h+i]})}_{=\frac{1}{c}} \\
&\quad + \sum_{i=t-h+1}^{c-h} \mathbb{P}(E_h|A_{[h+i,h+i]}) \underbrace{\mathbb{P}(A_{[h+i,h+i]})}_{=\frac{1}{c}} \\
&\quad + \mathbb{P}(E_h|A_{[c+1,m_k]}) \underbrace{\mathbb{P}(A_{[c+1,m_k]})}_{=0} \\
&= \frac{pt}{c} + \frac{1-p}{c}\sum_{i=1}^{t-h} F(i-1; h+i-1, 1-q) \\
&\quad + \frac{1}{c}\sum_{i=t-h+1}^{c-h} F(i-1; t, 1-q)
\end{aligned}$$

A.2 Supporting results

We will make use here of the following theorem, adapted here for our present purposes. (Theorem 3.3 in [24])

Theorem A.1. *Let $X \sim \text{Binom}(n,p)$, and let $t > 0$. We then have*

$$\begin{aligned}
\mathbb{P}\left(\frac{X}{n} - p \leq -t\right) &\leq \exp\left\{-\frac{t^2 n}{2p}\right\} \\
\mathbb{P}\left(\frac{X}{n} - p \geq t\right) &\leq \exp\left\{-\frac{t^2 n}{2p + 2t/3}\right\}.
\end{aligned}$$

We will also make use of the following two Pinsker-like inequalities that relate the Kullback-Liebler divergence to the total variation distance. To wit, let P denote a Bernoulli distribution with probability p and Q a Bernoulli distribution with probability q. Without loss of

generality, we let $q < 1/2$. Let $D_{\mathrm{KL}}(P\|Q)$ denote the Kullback-Leibler divergence between P and Q and $d_{\mathrm{TV}}(P,Q)$ the total variation distance between the measures. Then

$$2d_{TV}(P,Q)^2 = 2(p-q)^2 \leq D_{KL}(P\|Q) \leq \frac{2}{\min\{q,1-q\}} d_{TV}(P,Q)^2 = \frac{2}{q}(p-q)^2. \quad \text{(A.1)}$$

See Lemma 2.5 in [110] for the lower-bound, and Lemma 4.1 in [43] for the upper bound.

A.3 Proof of Lemma 4.1

Proof of Lemma 4.1. Under the null:

$$L(A;B^0|\tau) = \prod_{i<j} (B^0_{\tau_i\tau_j})^{A_{ij}} (1 - B^0_{\tau_i\tau_j})^{1-A_{ij}}$$

Defining the following function

$$\delta_\gamma(i,j) = \begin{cases} 1 & \text{if } (i,j) \in \mathscr{B}_l \times \mathscr{B}_k \text{ for } (l,k) \in \gamma_B \\ 0 & \text{otherwise} \end{cases}$$

and defining $n_\gamma = \sum_{i<j} \delta_\gamma(i,j)$, we can write,

$$\begin{aligned} \ell(B^0) &= \log\left(L(A;B^0|\tau)\right) \\ &= \sum_{i<j} A_{ij} \log(B^0_{\tau_i\tau_j}) + (1-A_{ij}) \log(1 - B^0_{\tau_i\tau_j}) \\ &= \sum_{\gamma\in\Gamma} \sum_{i<j} A_{ij} \log(B^0_\gamma) + (1-A_{ij}) \log(1 - B^0_\gamma) \\ &= \sum_{\gamma\in\Gamma} n_\gamma \hat{B}^{(0)}_\gamma \log(B^0_\gamma) + (1 - \hat{B}^{(0)}_\gamma) n_\gamma \log(1 - B^0_\gamma) \end{aligned}$$

Where $\hat{B}^0_\gamma = \frac{1}{n_\gamma}\sum_{i<j} A_{ij}\delta_\gamma(i,j)$. Since, log is a strictly increasing function on the support of $L(A;B^0|\tau)$, we get

$$\underset{B^0\in(0,1)^{K\times K}}{\operatorname{argmax}}\{\ell(B^0)\} = \underset{B^0\in(0,1)^{K\times K}}{\operatorname{argmax}}\{L(A;B^0|\tau)\}$$

We note that since $B^0_\gamma \in (0,1)$, we have

$$\frac{\partial \ell(B^0)}{\partial B^0_\gamma} = 0$$

if and only if

$$B^0_\gamma = \hat{B}^{(0)}_\gamma.$$

Writing

$$\ell(B^0) = \sum_{\gamma\in\Gamma_T} n_\gamma \hat{B}^{(0)}_\gamma \log(B^0_\gamma) + (1-\hat{B}^{(0)}_\gamma) n_\gamma \log(1-B^0_\gamma),$$

we have

$$\begin{aligned}
&\max_{B^0\in(0,1)^{K\times K}} \ell(B^0) \\
&\quad = \max_{B^0}\left\{\sum_{\gamma\in\Gamma_T} n_\gamma \hat{B}^{(0)}_\gamma \log(B^0_\gamma) + (1-\hat{B}^{(0)}_\gamma) n_\gamma \log(1-B^0_\gamma)\right\} \\
&\quad = \sum_{\gamma\in\Gamma_T} \max_{B^0_{lk}\in(0,1)}\left\{n_\gamma \hat{B}^{(0)}_\gamma \log(B^0_\gamma) + (1-\hat{B}^{(0)}_\gamma) n_\gamma \log(1-B^0_\gamma)\right\}.
\end{aligned}$$

We then conclude that the MLE under the null hypothesis of a hierarchical HSBM takes the form $\hat{B}^0 = [\hat{B}^0_{lk}]$, where, if $(l,k)\in\gamma$ then

$$\hat{B}^0_{lk} = \hat{B}^0_\gamma = \frac{1}{n_\gamma}\sum_{i<j} A_{ij}\delta_\gamma(i,j)$$

Under the alternative, the model is a traditional SBM, and the form of MLE is shown in

(for example) [16]. □

A.4 Proof of Theorem 4.2

Before proving Theorem 4.2, we first restate the Theorem for ease of reference.

Theorem 4.2: *Let $\mathbb{P}_{0,S}$ indicate the probability conditioned on the null hypothesis in 4.5 being the true distribution, where these $\{s_{ij}\}$ further satisfy*

$$\varepsilon_S = \min_{\gamma\in\Gamma} \min_{(i,j)\in\gamma_B} \left| s_{ij} - \frac{1}{|\gamma_B|} \sum_{(\ell,h)\in\gamma_B} s_{\ell h} \right| > 0. \tag{A.2}$$

Let

$$-2\hat{\lambda}_\gamma = \sum_{(l,k)\in\gamma_B} n_{lk} \log\left(\frac{1-\hat{B}^{(1)}_{lk}}{1-\hat{B}^{(0)}_{\gamma}}\right) + n_{lk}\hat{B}^{(1)}_{lk} \log\left(\frac{\hat{B}^{(1)}_{lk}(1-\hat{B}^{(0)}_{\gamma})}{\hat{B}^{(0)}_{\gamma}(1-\hat{B}^{(1)}_{lk})}\right)$$

We then have that $\mathbb{P}_{0,S}\left(\frac{\varepsilon_S^2 n_\gamma}{9} \le -2\widehat{\lambda}_\gamma \le \frac{8n_\gamma}{\delta}\right) \ge 1 - C(\varepsilon_S, \{n_{lk}\}, \delta)$, where

$$C(\varepsilon_S, \{n_{lk}\}, \delta) = 2e^{-2(\min(\varepsilon_S/3, \delta/2))^2 n_\gamma} + 2\sum_{(l,k)\in\gamma_B} e^{-2(\min(\varepsilon_S/3,\delta/2))^2 n_{lk}}$$

Proof. Let $\mathbb{E}_{0,S}[\cdot]$ (resp., $\mathbb{P}_{0,S}[\cdot]$) be the expectation (resp., probability) conditioned on the null from the hypotheses in Equation (4.5) and the collection of $\{s_{ij}\}_{(i,j)\in\gamma_B}$ being observed for each $\gamma\in\Gamma$ and that these s_{ij}'s satisfy Eq. 4.7. For a single collection of blocks $\gamma\in\Gamma$ and $(l,k)\in\gamma$, we have:

$$\begin{aligned}
\mathbb{E}_{0,S}[\hat{B}^{(1)}_{lk}] &= B^{(0)}_{\gamma} + s_{lk} \\
\mathbb{E}_{0,S}[\hat{B}^{(0)}_{\gamma}] &= B^{(0)}_{\gamma} + S_{\gamma}
\end{aligned}$$

Where $S_\gamma = \frac{1}{|\gamma_B|}\sum_{(l,k)\in\gamma_B} s_{lk}$. From Hoeffding's inequality applied to $\hat{B}_\gamma^{(0)}$ and $\hat{B}_{lk}^{(1)}$, we have that for any $t > 0$,

$$\begin{aligned}
&\mathbb{P}_{0,S}\left(\left|\hat{B}_\gamma^{(0)} - B_\gamma^{(0)} - S_\gamma\right| \geq t\right) + \sum_{(l,k)\in\gamma_B} \mathbb{P}_{0,S}\left(\left|\hat{B}_{lk}^{(1)} - B_\gamma^{(0)} - s_{lk}\right| \geq t\right) \\
&\leq 2e^{-2t^2 n_\gamma} + 2\sum_{(l,k)\in\gamma_B} e^{-2t^2 n_{lk}}
\end{aligned}$$

Let $t = \min(\varepsilon_S/3, \delta/2)$, and note that

$$\left|\hat{B}_{lk}^{(1)} - B_\gamma^{(0)} - s_{lk}\right| \leq t, \quad \text{and} \quad \left|\hat{B}_\gamma^{(0)} - B_\gamma^{(0)} - S_\gamma\right| \leq t, \tag{A.3}$$

holds with probability at least

$$1 - 2e^{-2n_\gamma t^2} - 2\sum_{(l,k)\in\gamma_B} e^{-2n_{lk}t^2}.$$

Given the event in Eq. A.3,

$$\begin{aligned}
\hat{B}_{lk}^{(1)} - B_\gamma^{(0)} - s_{lk} \geq -t; \quad -\hat{B}_\gamma^{(0)} + B_\gamma^{(0)} + S_\gamma \geq -t; \\
\hat{B}_{lk}^{(1)} - B_\gamma^{(0)} - s_{lk} \leq t; \quad -\hat{B}_\gamma^{(0)} + B_\gamma^{(0)} + S_\gamma \leq t.
\end{aligned}$$

Combining these inequalities, we see that

$$\begin{aligned}
&-2\varepsilon_S/3 - S_\gamma + s_{lk} \leq \hat{B}_{lk}^{(1)} - \hat{B}_\gamma^{(0)} \leq 2\varepsilon_S/3 - S_\gamma + s_{lk} \\
&\Rightarrow \begin{cases} \hat{B}_{lk}^{(1)} - \hat{B}_\gamma^{(0)} \leq -\varepsilon_S/3 & \text{if } S_\gamma > s_{lk} \\ \hat{B}_{lk}^{(1)} - \hat{B}_\gamma^{(0)} \geq \varepsilon_S/3 & \text{if } S_\gamma < s_{lk} \end{cases}
\end{aligned}$$

and hence $|\hat{B}^{(1)}_{lk} - \hat{B}^{(0)}_{\gamma}| \geq \varepsilon_S/3$. Applying A.1 with $P = \hat{B}^{(1)}_{lk}, Q = \hat{B}^{(0)}_{\gamma}$,

$$-2\hat{\lambda}_\gamma = \sum_{(l,k)\in\gamma_B} n_{lk} D_{\mathrm{KL}}(P||Q) \geq \sum_{(l,k)\in\gamma_B} n_{lk} \left(\hat{B}^{(1)}_{lk} - \hat{B}^{(0)}_{\gamma}\right)^2 \geq n_\gamma \varepsilon_S^2/9$$

Moreover, given the event in Eq. A.3, we have that $\hat{B}^{(0)}_{\gamma} \in (\delta/2, 1-\delta/2)$ so that Eq. A.1 also provides

$$-2\hat{\lambda}_\gamma \leq \sum_{(l,k)\in\gamma_B} \frac{4}{\delta} n_{lk} \left(\hat{B}^{(1)}_{lk} - \hat{B}^{(0)}_{\gamma}\right)^2 \leq \frac{8}{\delta} n_\gamma$$

as desired. □

A.5 Proof of Theorem 4.3

Consider the case where the graph under consideration comes from an RMHSBM distribution; note that below we will assume that the assumption in Eq. 4.10 holds. In this setting, an application of Theorem A.1 yields that for $n > n_0$ (letting $t = \sqrt{c_2 B^{(1)}_{\ell,k} \frac{\log n_{\ell,k}}{n_{\ell,k}}}$ for a constant $c_2 > 0$ to be set shortly)

$$\mathbb{P}(\hat{B}^{(1)}_{\ell,k} - B^{(1)}_{\ell,k} \leq -t) \leq e^{-\frac{c_2 \log n_{\ell,k}}{2}} \tag{A.4}$$

$$\begin{aligned}\mathbb{P}(\hat{B}^{(1)}_{\ell,k} - B^{(1)}_{\ell,k} \geq t) &\leq \exp\left\{-\frac{c_2 B^{(1)}_{\ell,k} \log n_{\ell,k}}{2B^{(1)}_{\ell,k} + \frac{2}{3}\sqrt{c_2 B^{(1)}_{\ell,k} \frac{\log n_{\ell,k}}{n_{\ell,k}}}}\right\} \\ &\leq \exp\left\{-\frac{c_2 \log n_{\ell,k}}{2 + \frac{2}{3}\sqrt{c_2}}\right\}\end{aligned} \tag{A.5}$$

Note that, if

$$|\hat{B}^{(1)}_{\ell,k} - B^{(1)}_{\ell,k}| \leq \sqrt{c_2 B^{(1)}_{\ell,k} \frac{\log n_{\ell,k}}{n_{\ell,k}}} \text{ for all } \{\ell,k\} \in \gamma, \tag{A.6}$$

then it holds that

$$|\hat{B}^{(0)}_{\gamma} - B^{(1)}_{\ell,k}| \leq \max_{(\ell,k)\in\gamma_B} \sqrt{c_2 B^{(0)}_{\ell,k} \frac{\log n_{\ell,k}}{n_{\ell,k}}}$$

as well. Now, (where $n_{**} = \min_{\ell,k} n_{\ell,k}$)

$$\begin{aligned}
&\mathbb{P}\left(\bigcap_{\gamma\in\Gamma}\bigcap_{(\ell,k)\in\gamma_B}\left\{|\hat{B}^{(1)}_{\ell,k} - B^{(1)}_{\ell,k}| \leq \sqrt{c_2 B^{(1)}_{\ell,k}\frac{\log n_{\ell,k}}{n_{\ell,k}}}\right\}\right) \\
&= \prod_{\gamma\in\Gamma}\prod_{(\ell,k)\in\gamma_B}\mathbb{P}\left(|\hat{B}^{(1)}_{\ell,k} - B^{(1)}_{\ell,k}| \leq \sqrt{c_2 B^{(1)}_{\ell,k}\frac{\log n_{\ell,k}}{n_{\ell,k}}}\right) \\
&\geq \prod_{\gamma\in\Gamma}\prod_{(\ell,k)\in\gamma_B}\left(1 - 2\exp\left\{-\frac{c_2 B^{(1)}_{\ell,k}\log n_{\ell,k}}{2B^{(1)}_{\ell,k} + \frac{2}{3}\sqrt{c_2 B^{(1)}_{\ell,k}\frac{\log n_{\ell,k}}{n_{\ell,k}}}}\right\}\right) \qquad \text{(A.7)} \\
&= \Omega\left(\exp\left(-((K^*)^2 + K^*)\cdot\exp\left\{-\frac{c_2\log n_{**}}{2+\frac{2}{3}\sqrt{c_2}}\right\}\right)\right) \qquad \text{(A.8)}
\end{aligned}$$

We then have that with probability at least the value in Eq. A.7, it uniformly holds that (where $n_{*,\gamma} = \min_{\{\ell,k\}\in\gamma} n_{\ell,k}$)

$$\hat{B}^{(1)}_{\ell,k} \in B^{(1)}_{\ell,k} \pm \sqrt{c_2 B^{(1)}_{\ell,k}\frac{\log n_{\ell,k}}{n_{\ell,k}}} \qquad \text{(A.9)}$$

$$\hat{B}^{(0)}_{\gamma} \in B^{(1)}_{\ell,k} \pm \sqrt{c_2 B^{(1)}_{\ell,k}\frac{\log n_{*,\gamma}}{n_{*,\gamma}}} \qquad \text{(A.10)}$$

Note that Eq. A.10 implies that

$$\hat{B}^{(0)}_{\gamma} \geq B^{(1)}_{\ell,k} - \sqrt{c_2 B^{(1)}_{\ell,k}\frac{\log n_{*,\gamma}}{n_{*,\gamma}}}$$

and

$$1 - \hat{B}^{(0)}_{\gamma} \geq 1 - B^{(1)}_{\ell,k} - \sqrt{c_2 B^{(1)}_{\ell,k}\frac{\log n_{*,\gamma}}{n_{*,\gamma}}}$$

Assuming Eq. A.9–A.10 hold moving forward, applying the Pinsker upper bound in Eq. A.1 to Eq. (4.4), for $n > n_0$ we then arrive at the following which holds with probability at least $1 - 4K^2 e^{-c_4 \log n_*}$.

$$\begin{aligned}
-2\hat{\lambda}_T &\leq \sum_{\gamma \in \Gamma} \sum_{(\ell,k) \in \gamma_B} n_{\ell,k} \frac{2}{\min(\hat{B}_\gamma^{(0)}, 1 - \hat{B}_\gamma^{(0)})} (\hat{B}_\gamma^{(0)} - \hat{B}_{\ell,k}^{(1)})^2 \\
&\leq \sum_{\gamma \in \Gamma} \sum_{(\ell,k) \in \gamma_B} n_{\ell,k} \frac{8 c_2 B_{\ell,k}^{(1)} \frac{\log n_{\ell,k}}{n_{\ell,k}}}{B_{\ell,k}^{(1)} - \sqrt{c_2 B_{\ell,k}^{(1)} \frac{\log n_{*,\gamma}}{n_{*,\gamma}}}} \quad \text{(applying Eqs. A.9 and A.10)} \\
&\leq \sum_{\gamma \in \Gamma} \sum_{(\ell,k) \in \gamma_B} \frac{8 c_2}{(1 - \sqrt{c_2})} \log n_{\ell,k}
\end{aligned}$$

We then have the BIC difference is equal to

$$\begin{aligned}
\hat{\Delta}_{T,BIC} &= -2\hat{\lambda}_T - \left(\sum_{\gamma \in \Gamma} (|\gamma_B| - 1) \right) \log \binom{n}{2} \\
&\leq \sum_{\gamma \in \Gamma} \left(|\gamma_B| \frac{8 c_2}{(1 - \sqrt{c_2})} - (|\gamma_B| - 1) \right) \log \binom{n}{2} \qquad \text{(A.11)} \\
&= \left[\left(\frac{8 c_2}{(1 - \sqrt{c_2})} - 1 \right) \frac{(K^*)^2 + K^*}{2} + |\Gamma| \right] \log \binom{n}{2}
\end{aligned}$$

Choosing $c_2 \leq \frac{1}{16} \left(1 - \frac{2|\Gamma|}{(K^*)^2 + K^*} \right)$ yields $\hat{\Delta}_{T,BIC} < 0$ as desired.

A.6 Proof of Theorem 4.4

Again, applying Theorem A.1, we have that with probability at least

$$1 - 4(K^*)^2 e^{-\frac{c_2 \log n_{**}}{2 + \frac{2}{3}\sqrt{c_3}}}$$

it uniformly holds that

$$\begin{aligned} \widehat{B}^{(1)}_{\ell,k} &\in B^{(1)}_{\ell,k} \pm \sqrt{c_3 B^{(1)}_{\ell,k} \frac{\log n_{\ell,k}}{n_{\ell,k}}} \text{ and} \\ \widehat{B}^{(1)}_{\gamma} &\in \mathbb{E}(\widehat{B}^{(1)}_{\gamma}) \pm \sqrt{c_3 \mathbb{E}(\widehat{B}^{(1)}_{\gamma}) \frac{\log n_{\gamma}}{n_{\gamma}}}, \end{aligned} \tag{A.12}$$

where $n_\gamma = \sum_{(\ell,k)\in\gamma_B} n_{\ell,k}$.

If the RMHSBM is (M,η)-incompatible with the true SBM, then for $n > n_0$, applying the lower Pinsker bound in Equation (A.1) to Equation (4.4) yields (we remind the reader that $C > 0$ is a constant which may change from line to line)

$$-2\widehat{\lambda}_T \geq \sum_{\gamma\in\Gamma} \sum_{(\ell,k)\in\gamma_B} n_{\ell,k} 4(\hat{B}^1_\gamma - \hat{B}^0_{\ell,k})^2 = \Omega\left(n_{**} M\eta^2 - CM\log n_{**}\right)$$

Hence, we have the BIC difference satisfies

$$\begin{aligned} \hat{\Delta}_{T,\mathrm{BIC}} &= -2\hat{\lambda}_T - \sum_{\gamma\in\Gamma} (|\gamma_B|-1)\log\binom{n}{2} \\ &= \Omega\left(n_{**}M\eta^2 - C\left(\binom{K^*}{2} + K^* + M\right)\log n\right) \end{aligned}$$

Hence, if $\eta^2 M = \omega((K^*)^2 \frac{\log n}{n_{**}})$, under the high-probability event in Equation (A.12), we have $\hat{\Delta}_{T,\mathrm{BIC}} > 0$, completing the proof.

Bibliography

[1] J. Agterberg, Y. Park, J. Larson, C. White, C. E. Priebe, and V. Lyzinski. Vertex nomination, consistent estimation, and adversarial modification. *Electronic Journal of Statistics*, 14(2):3230–3267, 2020.

[2] Al-Fahad M. Al-Qadhi, Carey E. Priebe, Hayden S. Helm, and Vince Lyzinski. Subgraph nomination: Query by example subgraph retrieval in networks, 2022.

[3] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza. Power to the people: The role of humans in interactive machine learning. *Ai Magazine*, 35(4):105–120, 2014.

[4] Arash Amini, Marina Paez, and Lizhen Lin. Hierarchical stochastic block model for community detection in multiplex networks. *Bayesian Analysis*, 19(1):319–345, 2024.

[5] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan Reutter, and Domagoj Vrgoč. Foundations of modern query languages for graph databases. *ACM Computing Surveys (CSUR)*, 50(5):1–40, 2017.

[6] Alex Arenas, Alberto Fernandez, and Sergio Gomez. Analysis of the structure of complex networks at different resolution levels. *New journal of physics*, 10(5):053039, 2008.

[7] Avanti Athreya, Donniell E Fishkind, Minh Tang, Carey E Priebe, Youngser Park, Joshua T Vogelstein, Keith Levin, Vince Lyzinski, Yichen Qin, and Daniel L Sussman. Statistical inference on random dot product graphs: a survey. *Journal of Machine Learning Research*, 18(226):1–92, 2018.

[8] Maria Axenovich. Repetitions in graphs and sequences. *Recent Trends in Combinatorics*, pages 63–81, 2016.

[9] László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 684–697, 2016.

[10] Davide Bacciu, Federico Errica, and Alessio Micheli. Contextual graph markov model: A deep and generative approach to graph processing. In *International conference on machine learning*, pages 294–303. PMLR, 2018.

[11] Sabine Bachl. Isomorphic subgraphs. In *Graph Drawing: 7th International Symposium, GD'99 Štiřín Castle, Czech Republic September 15–19, 1999 Proceedings 7*, pages 286–296. Springer, 1999.

[12] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 384–392, 2019.

[13] Sivaraman Balakrishnan, Min Xu, Akshay Krishnamurthy, and Aarti Singh. Noise thresholds for spectral clustering. *Advances in Neural Information Processing Systems*, 24, 2011.

[14] Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. The pushshift reddit dataset. In *Proceedings of the international AAAI conference on web and social media*, volume 14, pages 830–839, 2020.

[15] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.

[16] P. J. Bickel and A. Chen. A nonparametric view of network models and newman–girvan and other modularities. *Proceedings of the National Academy of Sciences*, 106(50):21068–21073, 2009.

[17] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[18] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca. Network analysis in the social sciences. *Science*, 323:892–895, 2009.

[19] Paul S Bradley and Olvi L Mangasarian. K-plane clustering. *Journal of Global optimization*, 16:23–32, 2000.

[20] L. J. N. Brent, J. Lehmann, and G. Ramos-Fernández. Social network analysis in the study of nonhuman primates: A historical perspective. *American Journal of Primatology*, 73(8):720–730, 2011.

[21] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[22] T. Caelli and S. Kosinov. An eigenspace projection clustering method for inexact graph matching. *IEEE transactions on pattern analysis and machine intelligence*, 26(4):515–519, 2004.

[23] S. Chatterjee. Matrix estimation by universal singular value thresholding. *The Annals of Statistics*, 43:177–214, 2014.

[24] Fan Chung and Linyuan Lu. Concentration inequalities and martingale inequalities: a survey. *Internet mathematics*, 3(1):79–127, 2006.

[25] Gerda Claeskens and Nils Lid Hjort. Model selection and model averaging. *Cambridge books*, 2008.

[26] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Structural inference of hierarchies in networks. In *ICML workshop on statistical network analysis*, pages 1–13. Springer, 2006.

[27] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[28] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 70(6):066111, 2004.

[29] G. Coppersmith. Vertex nomination. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(2):144–153, 2014.

[30] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[31] D. Delling, A. V. Goldberg, I. Razenshteyn, and R. F. Werneck. Graph partitioning with natural cuts. In *2011 IEEE International Parallel & Distributed Processing Symposium*, pages 1135–1146. IEEE, 2011.

[32] R. S. Desikan, F. Ségonne, B. Fischl, B. T. Quinn, B. C. Dickerson, D. Blacker, R. L. Buckner, A. M. Dale, R. P. Maguire, B. T. Hyman, et al. An automated labeling system for subdividing the human cerebral cortex on mri scans into gyral based regions of interest. *Neuroimage*, 31(3):968–980, 2006.

[33] Derek Doran. Triad-based role discovery for large social systems. In *International conference on social informatics*, pages 130–143. Springer, 2014.

[34] Sijia Fang and Karl Rohe. T-stochastic graphs. *arXiv preprint arXiv:2309.01301*, 2023.

[35] Xu Feng, JC Zhao, and Ke Xu. Link prediction in complex networks: a clustering perspective. *The European Physical Journal B*, 85:1–9, 2012.

[36] D. E. Fishkind, S. Adali, H. G. Patsolic, L. Meng, D. Singh, V. Lyzinski, and C. E. Priebe. Seeded graph matching. *Pattern recognition*, 87:203–215, 2019.

[37] D. E. Fishkind, V. Lyzinski, H. Pao, L. Chen, and C. E. Priebe. Vertex nomination schemes for membership prediction. *The Annals of Applied Statistics*, 9(3):1510–1532, 2015.

[38] D. E. Fishkind, L. Meng, A. Sun, C. E. Priebe, and V. Lyzinski. Alignment strength and correlation for graphs. *Pattern Recognition Letters*, 125:295–302, 2019.

[39] F. D. Forte. Network topology of the Argentine interbank money market. *Journal of Complex Networks*, 8(4), 2020.

[40] Ziqi Gao, Chenran Jiang, Jiawen Zhang, Xiaosen Jiang, Lanqing Li, Peilin Zhao, Huanming Yang, Yong Huang, and Jia Li. Hierarchical graph learning for protein–protein interaction. *Nature Communications*, 14(1):1093, 2023.

[41] Edgar N Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.

[42] Krzysztof J Gorgolewski, Natacha Mendes, Domenica Wilfling, Elisabeth Wladimirow, Claudine J Gauthier, Tyler Bonnen, Florence JM Ruby, Robert Trampel, Pierre-Louis Bazin, Roberto Cozatl, et al. A high resolution 7-tesla resting-state fmri test-retest dataset with cognitive and physiological measures. *Scientific data*, 2(1):1–13, 2015.

[43] Friedrich Götze, Holger Sambale, and Arthur Sinulis. Higher order concentration for functions of weakly dependent random variables. *Electronic Journal of Probability*, 24(85), 2019.

[44] M. S. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6), 1973.

[45] W. R. Gray, J. A. Bogovic, J. T. Vogelstein, B. A. Landman, J. L. Prince, and R. J. Vogelstein. Magnetic resonance connectome automated pipeline: an overview. *Pulse, IEEE*, 3(2):42–48, 2012.

[46] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

[47] Liang Hong, Lei Zou, Xiang Lian, and S Yu Philip. Subgraph matching with set similarity in a large graph database. *Ieee transactions on knowledge and data engineering*, 27(9):2507–2521, 2015.

[48] Jianwei Hu, Hong Qin, Ting Yan, and Yunpeng Zhao. Corrected bayesian information criterion for stochastic block models. *Journal of the American Statistical Association*, 115(532):1771–1783, 2020.

[49] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2:193–218, 1985.

[50] H. Jin, X. He, Y. Wang, H. Li, and A. L. Bertozzi. Noisy subgraph isomorphisms on multiplex networks. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4899–4905. IEEE, 2019.

[51] Jiashun Jin, Zheng Tracy Ke, Jiajun Tang, and Jingming Wang. Network goodness-of-fit for the block-model family. *arXiv preprint arXiv:2502.08609*, 2025.

[52] S. Johnson, V. Domínguez-García, L. Donetti, and M. A. Muñoz. Trophic coherence determines food-web stability. *Proceedings of the National Academy of Sciences*, 111(50):17923–17928, 2014.

[53] Vishesh Karwa, Debdeep Pati, Sonja Petrović, Liam Solus, Nikita Alexeev, Mateja Raič, Dane Wilburne, Robert Williams, and Bowei Yan. Monte carlo goodness-of-fit tests for degree corrected and related stochastic blockmodels. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86(1):90–121, 2024.

[54] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[55] V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts-a review. *IEEE transactions on pattern analysis and machine intelligence*, 29(7):1274–1279, 2007.

[56] Jing Lei. A goodness-of-fit test for stochastic block models. *Ann. Statist. 44(1)*, 2016.

[57] Jing Lei and Alessandro Rinaldo. Consistency of spectral clustering in stochastic block models. *The Annals of Statistics*, pages 215–237, 2015.

[58] Lihua Lei, Xiaodong Li, and Xingmei Lou. Consistency of spectral clustering on hierarchical stochastic block models. *arXiv preprint arXiv:2004.14531*, 2020.

[59] K. Levin, C. E. Priebe, and V. Lyzinski. On the role of features in vertex nomination: Content and context together are better (sometimes). *arXiv preprint arXiv:2005.02151*, 2020.

[60] Tianxi Li, Lihua Lei, Sharmodeep Bhattacharyya, Koen Van den Berge, Purnamrita Sarkar, Peter J Bickel, and Elizaveta Levina. Hierarchical community detection by recursive partitioning. *Journal of the American Statistical Association*, 117(538):951–968, 2022.

[61] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *International conference on machine learning*, pages 3835–3845. PMLR, 2019.

[62] M. Lissandrini, D. Mottin, T. Palpanas, and Y. Velegrakis. Graph-query suggestions for knowledge graph exploration. In *Proceedings of The Web Conference 2020*, pages 2549–2555, 2020.

[63] J. Lladós, E. Martí, and J. J. Villanueva. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1137–1143, 2001.

[64] V. Lyzinski, K. Levin, D. E. Fishkind, and C. E. Priebe. On the consistency of the likelihood maximization vertex nomination scheme: Bridging the gap between maximum likelihood estimation and graph matching. *Journal of Machine Learning Research*, 17(179):1–34, 2016.

[65] V. Lyzinski, K. Levin, and C. E. Priebe. On consistent vertex nomination schemes. *Journal of Machine Learning Research*, 20(69):1–39, 2019.

[66] V. Lyzinski, M. Tang, A. Athreya, Y. Park, and C. E. Priebe. Community detection and classification in hierarchical stochastic blockmodels. *IEEE Transactions on Network Science and Engineering*, 2017.

[67] Vince Lyzinski, Donniell E Fishkind, and Carey E Priebe. Seeded graph matching for correlated erdös-rényi graphs. *J. Mach. Learn. Res.*, 15(1):3513–3540, 2014.

[68] Peter W MacDonald, Elizaveta Levina, and Ji Zhu. Mesoscale two-sample testing for network data. *arXiv preprint arXiv:2410.17046*, 2024.

[69] Louis Mahon, Eleonora Giunchiglia, Bowen Li, and Thomas Lukasiewicz. Knowledge graph extraction from videos. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 25–32. IEEE, 2020.

[70] David Marchette, CE Priebe, and Glen Coppersmith. Vertex nomination via attributed random dot product graphs. In *Proceedings of the 57th ISI World Statistics Congress*, volume 6, page 16. Citeseer, 2011.

[71] Marina Meilă. Comparing clusterings—an information based distance. *Journal of multivariate analysis*, 98(5):873–895, 2007.

[72] Lingyao Meng, Mengqi Lou, Jianyu Lin, Vince Lyzinski, and Donniell E Fishkind. On seeded subgraph-to-subgraph matching: The sssgm algorithm and matchability information theory. *arXiv preprint arXiv:2306.04016*, 2023.

[73] Bruno T Messmer and Horst Bunke. Efficient subgraph isomorphism detection: a decomposition approach. *IEEE transactions on knowledge and data engineering*, 12(2):307–323, 2000.

[74] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[75] Jacob D. Moorman, Qinyi Chen, Thomas K. Tu, Zachary M. Boyd, and Andrea L. Bertozzi. Filtering methods for subgraph matching on multiplex networks. *2018 IEEE International Conference on Big Data (Big Data)*, 2018.

[76] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas. Exemplar queries: a new way of searching. *The VLDB Journal*, 25(6):741–765, 2016.

[77] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[78] Roberto Imbuzeiro Oliveira. Concentration of the adjacency matrix and of the laplacian in random graphs with independent edges. *arXiv preprint arXiv:0911.0600*, 2009.

[79] Yongjin Park, Cristopher Moore, and Joel S Bader. Dynamic networks from hierarchical bayesian graph clustering. *PloS one*, 5(1):e8118, 2010.

[80] H. G. Patsolic, Y. Park, V. Lyzinski, and C. E. Priebe. Vertex nomination via local neighborhood matching. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 13(3):229–244, 2020.

[81] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[82] Tiago P Peixoto. Hierarchical block structures and high-resolution model selection in large networks. *Physical Review X*, 4(1):011047, 2014.

[83] Christina Prell and John Skvoretz. Looking at social capital through triad structures. *Connections*, 28(2):4–16, 2008.

[84] P. Rastogi, A. Poliak, V. Lyzinski, and B. Van Durme. Neural variational entity set expansion for automatically populated knowledge graphs. *Information Retrieval Journal*, 22(3-4):232–255, 2019.

[85] Pushpendre Rastogi, Vince Lyzinski, and Benjamin Van Durme. Vertex nomination on the cold start knowledge graph. *Human Language Technology Center of Excellence: Technical report*, 2017.

[86] John W Raymond, Eleanor J Gardiner, and Peter Willett. Rascal: Calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal*, 45(6):631–644, 2002.

[87] Douglas A Reynolds et al. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663), 2009.

[88] Mehrdad Rostami, Mourad Oussalah, and Vahid Farrahi. A novel time-aware food recommender-system based on deep learning and graph clustering. *IEEE Access*, 10:52508–52524, 2022.

[89] Marta Sales-Pardo, Roger Guimera, 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.

[90] Karin Saltoun, Ralph Adolphs, Lynn K Paul, Vaibhav Sharma, Joern Diedrichsen, BT Thomas Yeo, and Danilo Bzdok. Dissociable brain structural asymmetry patterns reveal unique phenome-wide profiles. *Nature Human Behaviour*, 7(2):251–268, 2023.

[91] Kanigalpula Samanvi and Naveen Sivadasan. Subgraph similarity search in large graphs. *arXiv preprint arXiv:1512.05256*, 2015.

[92] Venu Satuluri and Srinivasan Parthasarathy. Symmetrizations for clustering directed graphs. In *Proceedings of the 14th international conference on extending database technology*, pages 343–354, 2011.

[93] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[94] Edward R Scheinerman and Kimberly Tucker. Modeling graphs using dot product representations. *Computational statistics*, 25:1–16, 2010.

[95] G. Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

[96] F. Schweitzer, G. Fagiolo, D. Sornette, F. Vega-Redondo, A. Vespignani, and D. R. White. Economic networks: The new challenges. *Science*, 325(5939):422–425, 2009.

[97] Luca Scrucca, Chris Fraley, T. Brendan Murphy, and Adrian E. Raftery. *Model-Based Clustering, Classification, and Density Estimation Using mclust in R*. Chapman and Hall/CRC, 2023.

[98] Haichuan Shang, Ke Zhu, Xuemin Lin, Ying Zhang, and Ryutaro Ichise. Similarity search on supergraph containment. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 637–648. IEEE, 2010.

[99] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

[100] Shashank Sheshar Singh, Samya Muhuri, Shivansh Mishra, Divya Srivastava, Harish Kumar Shakya, and Neeraj Kumar. Social network analysis: A survey on process, tools, and application. *ACM Computing Surveys*, 56(8):1–39, 2024.

[101] R. V. Sole and J. M. Montoya. Complexity and fragility in ecological networks. *Proceedings of the Royal Society of London B*, 268(1480):2039–2045, 2001.

[102] O. Sporns. *Discovering the Human Connectome*. The MIT Press, 2016.

[103] O. Sporns. The complex brain: connectivity, dynamics, information. *Trends in cognitive sciences*, 26(12):1066–1067, 2022.

[104] Daniel L Sussman, Youngser Park, Carey E Priebe, and Vince Lyzinski. Matched filters for noisy induced subgraph detection. *IEEE transactions on pattern analysis and machine intelligence*, 42(11):2887–2900, 2019.

[105] Daniel L Sussman, Minh Tang, Donniell E Fishkind, and Carey E Priebe. A consistent adjacency spectral embedding for stochastic blockmodel graphs. *Journal of the American Statistical Association*, 107(499):1119–1128, 2012.

[106] S. Suwan, D. S. Lee, and C. E. Priebe. Bayesian vertex nomination using content and context. *Wiley Interdisciplinary Reviews: Computational Statistics*, 7(6):400–416, 2015.

[107] M. Tang, A. Athreya, D. L. Sussman, V. Lyzinski, and C. E. Priebe. A nonparametric two-sample hypothesis testing problem for random dot product graphs. *Bernoulli*, 23(3):1599–1630, 2017.

[108] Minh Tang and Carey E Priebe. Limit theorems for eigenvectors of the normalized laplacian for random graphs. 2018.

[109] V. A. Traag, L. Waltman, and N. J. Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*, 9(1):1–12, 2019.

[110] Alexandre B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer New York, NY, 2009.

[111] Shahadat Uddin and Liaquat Hossain. Dyad and triad census analysis of crisis communication network. *Social Networking*, 2013.

[112] Julian R Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, 23(1):31–42, 1976.

[113] Y. X. R. Wang and P. J. Bickel. Likelihood-based model selection for stochastic block models. *The Annals of Statistics*, 45(2):500–528, 2017.

[114] Xiaoran Yan. Bayesian model selection of stochastic block models. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 323–328. IEEE, 2016.

[115] Congyuan Yang, Carey E Priebe, Youngser Park, and David J Marchette. Simultaneous dimensionality and complexity model selection for spectral graph clustering. *Journal of Computational and Graphical Statistics*, 30(2):422–441, 2020.

[116] Dominic Yang, Yurun Ge, Thien Nguyen, Denali Molitor, Jacob D Moorman, and Andrea L Bertozzi. Structural equivalence in subgraph matching. *IEEE Transactions on Network Science and Engineering*, 10(4):1846–1862, 2023.

[117] Junhan Yang, Zheng Liu, Shitao Xiao, Chaozhuo Li, Defu Lian, Sanjay Agrawal, Amit Singh, Guangzhong Sun, and Xing Xie. Graphformers: Gnn-nested transformers for representation learning on textual graph. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 28798–28810. Curran Associates, Inc., 2021.

[118] J. Yoder, L. Chen, H. Pao, E. Bridgeford, K. Levin, D. E. Fishkind, C. E. Priebe, and V. Lyzinski. Vertex nomination: The canonical sampling and the extended spectral nomination schemes. *Computational Statistics & Data Analysis*, 145:106916, 2020.

[119] Ye Yuan, Guoren Wang, Lei Chen, and Haixun Wang. Efficient subgraph similarity search on large probabilistic graph databases. *arXiv preprint arXiv:1205.6692*, 2012.

[120] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[121] M. Zhu and A. Ghodsi. Automatic dimensionality selection from the scree plot via the use of profile likelihood. *Computational Statistics and Data Analysis*, 51:918–930, 2006.

[122] X.-N. Zuo, J. S. Anderson, P. Bellec, R. M. Birn, B. B. Biswal, J. Blautzik, J. C. S. Breitner, R. L. Buckner, V. D. Calhoun, F. X. Castellanos, et al. An open science resource for establishing reliability and reproducibility in functional connectomics. *Scientific data*, 1(1):1–13, 2014.