Graph Inference and Graph Matching

by

Henry Pao

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

August, 2014

Abstract

Graphs are widely used in many fields of research, ranging from natural sciences to computer and mathematical sciences. Graph inference is an area of intense research. In this dissertation, we propose several methodologies in graph inference. We focus on statistical inference using graph invariants, vertex nomination, and a divide-and-conquer graph matching technique.

We present a comparative power analysis of various graph invariants for testing the hypothesis that the graph has a subgraph with higher edge probability. Given a graph drawn from a kidney-egg random graph model, the null hypothesis is that all edge probabilities are equal. The alternative hypothesis is that there exists a subset of vertices which are more likely to be adjacenct to each other than the rest of the graph. Using Monte Carlo simulations, we estimate the power of several graph invariants acting as test statistics. We discovered that for many choices of parameters in the random graph model, the scan statistic and clustering coefficient often dominate other graph invariants. However, our results indicates that none of the graph invariants considered is uniformly most powerful.

ABSTRACT

Given a graph drawn from a stochastic block model where one block is of particular interest, vertex nomination is the task of creating a list of vertices such that there are an abundance of vertices from the block of interest at the top of the list. Vertex nomination is useful in situations where only a limited number of vertices can be examined and have their block membership checked. We propose several vertex nomination schemes, derive theoretical results for performance, and compare the schemes on simulated and real data.

Given two graphs, graph matching is to create a mapping from one set of vertices to the other, such that the edge structure of the graphs is preserved as best as possible. We develop a new method for scaling graph matching algorithms, and prove performance guarantees. Any graph matching algorithm can be scaled using our divide-and-conquer technique. The performance of this technique is demonstrated on large simulated graphs and human brain graphs.

Primary Reader: Donniell Fishkind

Secondary Reader: Carey Priebe

Acknowledgments

I want to first thank my advisors Donniell Fishkind and Vince Lyzinski for all their help and support in writing this thesis. Their help is instrumental in completing my thesis.

I would also like to thank Carey Priebe for serving as my second reader, and insight on numerous graph inference subjects. I want to also thank Avanti Athreya and Glen Coppersmith for serving on my defense committee.

Finally, I want to thank my friends and family for their help and support.

Dedication

This thesis is dedicated to my fiancée Li Chen, whose continual love and support is irreplaceable.

Contents

List of Algorithms

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Graphs have been studied since the 1700s, when Leonhard Euler first contemplated traversals of the seven bridges of Königsberg [1]. Graphs naturally arise in biology, neuroscience [2,3], computer vision [4,5], social networks [6,7], internet communication [8], and many other fields. There has been a perpetual interest in graph inference.

In this dissertation, we examine hypothesis testing using graph invariants as test statistics, the problem of vertex nomination, and a divide-and-conquer graph matching technique. We present a comparative power analysis of various graph invariants for testing the hypothesis that the graph has a subgraph with higher edge probability. Our results indicate that none of the graph invariants considered is uniformly most

powerful. Consider a graph drawn from a stochastic block model, where one of the blocks is of particular interest. The task of vertex nomination is to create a "nomination list" of vertices such that vertices from the block of interest are concentrated near the top of the list. We propose several vertex nomination schemes, derive theoretical results for performance, and compare the schemes on simulated and real data. We also develop a new method for scaling graph matching algorithms to large graphs, and prove performance guarantees. The performance of this technique is demonstrated on large simulated graphs and human brain connectomes.

This dissertation starts with an introductory chapter containing a literature review, notations, definitions, and the underlying setting – in particular this includes random graph models. In this thesis, most of the methodologies are developed in the context of the stochastic block model or the random dot product model.

In Chapter 2, we discuss several foundational tools for random graph inference. We begin with graph invariants, which are graph properties that are invariant under graph isomorphisms. Next, we discuss spectral embedding methods for adjacency matrices. We describe the problem of graph matching and review several inexact graph matching algorithms. Of the graph matching algorithms, we focus on fast approximation to quadratic assignment programing [34].

In Chapter 3, we study hypothesis testing, when given a random graph distributed as a kidney-egg random graph, on whether the edge probability between a subset of vertices is higher than the rest of the graph. Specifically, the null hypothesis is that

all pairs of vertices have the same probability of being an edge, and the alternative hypothesis is that there is an "egg" of vertices with higher edge probability than in the rest of the graph. We compare the power of several graph invariants as test statistics via Monte Carlo simulations. Of the graph invariants we used, our results indicate that there is no uniformly most powerful graph invariant among the collection which we consider.

Following in Chapter 4, the problem of vertex nomination is introduced. Given a graph drawn from a stochastic block model where one block is of particular interest, vertex nomination is the task of creating a list of vertices such that vertices from the block of interest are in abundance at the top of the list. Vertex nomination is useful in situations where only a limited number of vertices can be examined to check their block membership. We propose several vertex nomination schemes, including a canonical vertex nomination scheme, Metropolis-Hastings sampling vertex nomination scheme, spectral embedding vertex nomination scheme, and graph matching vertex nomination scheme. We derive theoretical results for performance, and compare the schemes on simulated and real data.

Finally in Chapter 5, we present a method of scaling graph match algorithms. This is a divide-and-conquer approach to parallelize inexact graph matching algorithms. We compare the performance of many graph matching algorithms within this paradigm, and demonstrate the effectiveness of our proposed technique on human brain connectome graphs.

1.2 Brief Literature Review

In this dissertation, we consider the stochastic block model [11, 12] as the underlying framework. For an observed graph drawn from a stochastic block model, the task of estimating the block memberships of vertices has been widely studied. Some methodologies include Bayesian methods [13], likelihood maximization [14,15], graphical models [16,17], and spectral methods [18–20].

One major component of this dissertation is the graph matching problem. Given two graphs, graph matching is to create a bijection from one set of vertices to the other, such that the edge structure of the graphs is preserved as best possible. Graph matching can be separated into three categories: exact graph matching, optimal inexact graph matching, and approximate inexact graph matching. Given two isomorphic graphs, exact graph matching is to find an isomorphism – a bijection between the two graphs which preserves all edge connectivity. Of course, when the graphs are not isomorphic, exact graph matching is not possible. Optimal inexact graph matching is to find the global minimum of a matching objective, where there are several possible different metrics. We are interested in minimizing the number of edge disagreements between two graphs [60]. Approximate inexact graph matching finds a local minimum of the matching objective. Conte, Foggia, Sansone and Vento summarize recent graph matching efforts in [21].

One problem closely related to exact graph matching is the graph isomorphism problem, which is to determine whether two graphs are isomorphic. That is, one

intends to determine if there exists an isomorphism between the two graphs. This problem is notoriously of unknown complexity [22–25]. Algorithms for solving many related problems such as subgraph isomorphism, monomorphism, homomorphism, and maximum common subgraph are NP-complete [21]. The problem of finding a bijection to minimize the number of edge disagreements from one graph to another graph for loopy, weighted, and directed graphs is equivalent to the quadratic assignment problem, which is NP-hard.

Solving exact graph matching can be efficiently done for certain kinds of graphs. Polynomial time exact graph matching algorithms have been found for matching trees by Aho et al. [26], planar graphs by Hopcroft and Wong [27], and bounded valence graphs by Luks [28]. One popular technique devised by Ullmann [29] for exact graph matching on general graphs is tree search with backtracking. An algorithm called Nauty not based on tree search was developed by McKay [30].

Since many of the exact graph matching problems are NP-hard, seeking a good approximate solutions in reasonable time is still an active field of research. The algorithms developed in this area are called inexact graph matching algorithms. One category of these algorithms are derived from tree search, first developed by Tsai and Fu [31]. Another technique for inexact graph matching relies on spectral embedding by Umeyama [32]. A third category is based on continuous optimization methods, by relaxing a combinatorial objective. One of the first such methods is due to Fischler and Elschlager [33]. In this thesis, we are primarily concerned with inexact graph

matching, utilizing a continuous relaxation optimization method, developed by Vogelstein et al. [34]. Henceforth, the term "graph matching" refers to inexact graph matching.

1.3 Notation

This section defines the notation to be used through out this dissertation.

For any vector $v = [v_i]$, let v_i denote the i^{th} element of v. For any matrix $M = [m_{ij}]$, let m_{ij} denote the element in M from the i^{th} row and j^{th} column.

Let $\mathbb{R}$ be the set of real numbers, and $\mathbb{N} = \{1, 2, 3, \ldots\}$ be the set of natural numbers.

Let (a, b) denote the set of real numbers between a and b noninclusive, and $[a, b]$ denote the set of real numbers between a and b inclusive.

If $A \in \mathbb{R}^{k \times n}$ and $B \in \mathbb{R}^{k \times m}$ are matrices (or vectors) with the same number of rows, let $[A, B] \in \mathbb{R}^{k \times (n+m)}$ be the horizontal concatenation of A, B. Similarly, if $A \in \mathbb{R}^{n \times k}$ and $B \in \mathbb{R}^{m \times k}$ are matrices (or vectors) with the same number of columns, let $[A; B] \in \mathbb{R}^{(n+m) \times k}$ be the vertical concatenation of A, B.

A matrix A is symmetric if $A = A^T$. A matrix $A = [a_{ij}] \in \mathbb{R}^{n \times n}$ is hollow if, for all $i \in \{1, 2, 3, \ldots n\}$, $a_{ii} = 0$.

For all $x, y \in \mathbb{R}^n$, we define the "Euclidean" dot product $\langle \cdot, \cdot \rangle$ as $\langle x, y \rangle = x^T y \in \mathbb{R}$.

Let $\subseteq$ denote subset allowing equality and $\subset$ denote subset excluding equality. Let $\mathbb{1}[\cdot]$ denote the indicator function.

Let V be a set, then let $\binom{V}{2}$ denote all (unordered) pairs $\{v, v'\}$ such that $v, v' \in V$ and $v \neq v'$.

Let $\|\cdot\|_p$ be defined to be the ℓ_p vector norm - that is, for a vector $x = [x_i] \in \mathbb{R}^n$,

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}.$$

In particular, we denote the ℓ_2 norm as

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}.$$

Let $\|\cdot\|_F$ be the Frobenius norm - that is, for any matrix $A = [a_{ij}] \in \mathbb{R}^{m \times n}$

$$\|A\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^2}.$$

We define the induced matrix norm of ℓ_α and ℓ_β vector norms as

$$\|A\|_{\alpha,\beta} = \sup_{x \neq 0} \frac{\|Ax\|_\beta}{\|x\|_\alpha} = \sup_{\|x\|_\alpha = 1} \|Ax\|_\beta.$$

In particular, for $A \in \mathbb{R}^{m \times n}$

$$\|A\|_{2,\infty} = \sup_{\|x\|_2 = 1} \|Ax\|_\infty = \sup_{\|x\|_2 = 1} \max_i |A_i x| = \max_{i \in [m]} \|A_i\|_2,$$

where A_i is the i^{th} column of A.

For the following definitions, let f and g be functions defined on the set of real numbers or natural numbers. If there exists some $M > 0$ and $x_0 \in \mathbb{R}$ such that

$$|f(x)| \leq M|g(x)|, \ \forall x > x_0,$$

then we say

$$f(x) = O(g(x)).$$

If for all $\epsilon > 0$ there exists $x_0 \in \mathbb{R}$ such that

$$|f(x)| \geq \epsilon |g(x)|, \ \forall x > x_0,$$

then we say

$$f(x) = \Omega(g(x)).$$

For every $\epsilon > 0$, there exists a $x_0 \in \mathbb{R}$ such that

$$|f(x)| \leq \epsilon |g(x)|, \ \forall x > x_0,$$

then we say

$$f(x) = o(g(x)).$$

Definition Here, we present graph theory definitions and notation. For this dissertation we are mainly concerned with undirected graphs. When convenient, we provide details on how to apply methods to directed graphs.

- A **graph** $G = (V, E)$ is defined to be a vertex set V and an edge set E, where E consists of two element subsets of V. Again, we are primarily concerned with undirected graphs, but occasionally we consider directed graphs for completeness or convenience. In a directed graph the edge set E consists of ordered pairs of elements from V.

- A vertex $v \in V$ is **adjacent** to $v' \in V$ if $\{v, v'\} \in E$, we denote this as $v \sim v'$.

- A **subgraph** $G' = (V', E')$ of $G = (V, E)$ is a graph such that $V' \subseteq V$ and $E' \subseteq E$. This is denoted as $G' \subseteq G$.

- For any $V' \subseteq V$, the **subgraph of G induced by V'**, denoted $G[V']$, is the graph $G' = (V', E')$, such that for all $v, v' \in V', \{v, v'\} \in E'$ if and only if $\{v, v'\} \in E$.

- A **walk** is a list of vertices $w = (v_0, v_1, v_2, \dots v_l)$, such that $v_{i-1} \sim v_i$ for $i \in \{1, 2, 3, \dots l\}$. The **length** of a walk is the number of edges in the walk, equivalent to the number of vertices minus 1, i.e. the length of w is l. A **path** is a walk with no repeated vertices.

- The **distance** between two vertices in a graph is the length of a minimum length walk between the two vertices. Distance is denoted as $d(\cdot, \cdot)$. For all $v, v' \in V$, we have $d(v, v) := 0$, and if no path exists between v and v', then $d(v, v') := \infty$.

- The **k^{th}-order neighborhood** of a vertex v are all vertices u such that $d(u, v) \leq k$. We denote the k^{th} order neighborhood of vertex v in graph G as $N_k(v; G)$.

- Let $G = (V, E)$ be a graph and say $V = \{v_1, v_2, \dots v_{|V|}\}$. The **adjacency matrix** $A = [a_{ij}] \in \{0, 1\}^{|V| \times |V|}$ of G is a matrix, such that for all $i, j \in [|V|]$ $a_{ij} = \mathbb{1}[\{v_i, v_j\} \in E]$. Although the vertex set V may not have any natural ordering, to create an adjacency matrix, an ordering of the vertices is required. Any ordering is permitted. The adjacency matrix in our setting is symmetric and hollow. If the graph is directed, then the adjacency matrix is not necessarily symmetric but is still hollow.

Definition Consider a sequence of events $(E_1, E_2, E_3, \dots)$. We say the sequence

happens **almost always** to mean that

$$\mathbb{P}[\liminf_{n\to\infty} E_n] = \mathbb{P}\left[\cup_{i=1}^{\infty} \cap_{j=i}^{\infty} E_j\right] = 1.$$

1.4 Random Graph Models

In this section, we present several random graph models pertaining to our subsequent analysis. In all of the presented random graphs, the vertex set is fixed, while the edge set is random.

1.4.1 Erdős-Rényi Random Graph

Definition An **Erdős-Rényi random graph** $G \sim \kappa(n, p)$, has two parameters; the number of vertices n, and edge probability p. Let V be a set of vertices such that $n = |V|$. A graph is an **Erdős-Rényi random graph**, if for all $\{v, v'\} \in \binom{V}{2}$, $\mathbb{1}[\{v, v'\} \in E]$ is independent identically distributed (i.i.d.) Bernoulli(p). Let $G \sim \kappa(n, p)$ denote an Erdős-Rényi random graph.

The Erdős-Rényi random graph has been well studied and appears often in the literature. It was first introduced in 1959 by Béla Bollobás' [9]. We are interested in the Erdős-Rényi random graph, because it is one of the simplest random graph models with many known properties.

1.4.2 Dense Sub-block Random Graph

Definition A **kidney and egg random graph** $G = (V, E) \sim \kappa(b, p, q)$ has parameters; **block membership function** $b : V \mapsto \{1, 2\}$ and $p, q \in (0, 1)$. Independently, for all $\{v, v'\} \in \binom{V}{2}$,

$$\mathbb{P}[\{v, v'\} \in E] = \begin{cases} q, & \text{if } b(v) = b(v') = 1 \\ p, & \text{otherwise} \end{cases}.$$

When $q > p$, we refer to the graph as a **dense sub-block random graph**.

For $k \in \{1, 2\}$, let the pre-image of b be denoted by $V_k = \{v : f(v) = k\}$; we call these sets **blocks**.

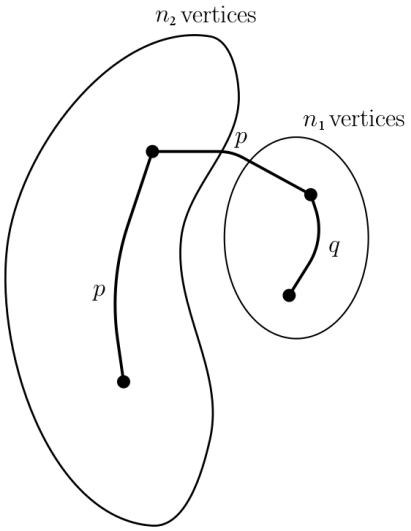

Figure 1.1: Depiction of a kidney and egg graph.

This is a more complex random graph model than the Erdős-Rényi random graph [35]. The "egg" refers to the induced subgraph of the vertices with inter-block edge

probability q, the "kidney" refers the induced subgraph of the remaining vertices with inter-block edge probability p, and the edges between the egg and kidney have probability p. A depiction of a kidney and egg random graph is shown in Figure 1.1.

Note that if $p = q$, then this model is not identifiable, because edges between vertices in the "egg" have the same distribution as vertices in the rest of the graph.

Often, the parameter b is unobserved. It is also useful to consider the following alternative definition of the kidney-egg random graph.

Definition A **kidney and egg random graph** $G = (V, E) \sim \kappa(n_1, n_2, p, q)$ has parameters $n_1, n_2 \in \mathbb{N}$ and $p, q \in (0, 1)$. Let the vertex set be V such that $|V| = n_1 + n_2$. Consider all possible **block membership functions** $b : V \mapsto \{1, 2\}$ such that the cardinality $|V_k| = |\{v : b(v) = k\}|$ of the pre-image of k is n_k. Let $\mathcal{B}$ denote the set of all possible $\binom{n_1+n_2}{n_1}$ block membership functions. Let b be uniformly distributed over $\mathcal{B}$. Given b, independently for all $\{v, v'\} \in \binom{V}{2}$

$$\mathbb{P}[\{v, v'\} \in E] = \begin{cases} q, & \text{if } b(v) = b(v') = 1 \\ p, & \text{otherwise} \end{cases}.$$

1.4.3 Stochastic Block Random Graph

The next model is a generalization of the kidney and egg random graph.

Definition A **stochastic block random graph** $G = (V, E) \sim (b, \Lambda)$ has two parameters; a matrix $\Lambda = [\lambda_{ij}] \in [0, 1]^{K \times K}$ and a **block membership function**

$b : V \mapsto K$. For all $\{v, v'\} \in \binom{V}{2}$, the edges are independent $\mathbb{1}[\{v, v'\} \in E] \sim$ Bernoulli($\lambda_{b(v)b(v')}$).

For $k \in \{1, 2, \ldots, K\}$, let V_k denote the pre-image $\{v \in V : b(v) = k\}$. We call $V_1, V_2, \ldots, V_K$ **blocks**.

Notice that the blocks $V_1, V_2, \ldots, V_K$ partition the vertex set. For this model to be identifiable, the edge probabilities of different blocks must be different from each other, i.e., all rows (or columns) of Λ are distinct. In the directed case for Λ to be identifiable, for all distinct $i, j \in \{1, 2, \ldots K\}$ either row i^{th} is not equal to the j^{th} row or the i^{th} column is not equal to the j^{th} column.

Often, the block membership function is not observed. As an alternative definition of the stochastic block random graph, we can define the model without b as a parameter, in the following manner.

Definition A **stochastic block random graph** $G = (V, E) \sim (N, \Lambda)$, has parameters $\Lambda \in [0, 1]^{K \times K}$ and $N = (n_1, n_2, \ldots, n_K)$, where n_k are positive integers. Let V be the set of n vertices, where $n = \sum_{k=1}^{K} n_k$. Let $\mathcal{B}$ be the set of all **block membership functions** $b : V \mapsto \{1, 2, \ldots K\}$ such that, all $k \in \{1, 2, \ldots, K\}$, the cardinality of the pre-image $|\{v \in V : b(v) = k\}| = n_k$. There are $\binom{n}{n_1, n_2, \ldots, n_K}$ such block membership functions. Let the block membership function b be uniformly distributed over $\mathcal{B}$. Then, conditioning on $b \in \mathcal{B}$, for all $\{v, v'\} \in \binom{V}{2}$, the edges are independent $\mathbb{1}[\{v, v'\} \in E] \sim$ Bernoulli($\lambda_{b(v)b(v')}$).

Often we let the vertex set V be denoted as $\{v_1, v_2, \ldots, v_n\}$. In these cases, we occasionally denote $b = [b_i] \in \{1, 2, \ldots, K\}^n$ as a vector with $b_i = b(v_i)$. When there is possible confusion, we denote the blocks $V_1^b, V_2^b, \ldots, V_K^b$.

For convenience, let us define an "expanded Λ" as $\overline{A} = [\overline{a}_{ij}] \in [0,1]^{n \times n}$ where $\overline{a}_{ij} = \Lambda_{b_i, b_j}$. When excluding the diagonal, $\overline{A}$ also may be viewed as the mean of the adjacency matrix A, i.e. $\mathbb{E}[A] = \overline{A}$.

In certain instances, the block membership function b is partially observed. In other words, its value is known for some vertices, say $U \subseteq V$, and not others, say $W \subseteq V$. The observed and unobserved vertices partition the vertex set $V = U \cup W$. We call these vertices in U **seed vertices**. Just as before, for all $k \in \{1, 2, 3, \ldots K\}$, define U_k, W_k be the inverse image of k under b for vertices in U and W respectively ($V_k = U_k \cup W_k$). Define $S = [s_k] \in \mathbb{N}^K$, $M = [m_k] \in \mathbb{N}^K$ be vectors of length K, such that for all $k \in [K]$, $s_k = |U_k|$ and $m_k = |W_k|$, and define $s = \sum_{k \in \{1,2,3,\ldots K\}} s_k$ and $m = \sum_{k \in \{1,2,3,\ldots K\}} m_k$.

Note that the dense sub-block model is a special case of a stochastic block random graph with 2-blocks where

$$\Lambda = \begin{bmatrix} q & p \\ p & p \end{bmatrix}.$$

1.4.4 Bernoulli Random Graph

Definition A **Bernoulli random graph** $G \sim \kappa(\overline{A})$ has the parameter matrix of edge probabilities $\overline{A} = [\overline{a}_{ij}] \in [0,1]^{n \times n}$. Let $V = \{v_1, v_2, \ldots, v_n\}$ be the set of ver-

tices. A graph $G = (V, E)$ is an **Bernoulli random graph** if, independently for all $\{v_i, v_j\} \in \binom{V}{2}$, $\mathbb{1}[\{v_i, v_j\} \in E] \sim$Bernoulli$(\overline{a}_{ij})$.

If the graph is undirected, then $\overline{A}$ is a symmetric and hollow. If the graph is directed, then $\overline{A}$ may not be symmetric. If we write $\kappa(\overline{A})$ where $\overline{A}$ is not hollow, then let it be understood that we treat $\overline{a}_{ii}$ as if it is zero for $i \in \{1, 2, \ldots, n\}$.

Consider a stochastic block random graph $\kappa(b, \Lambda)$, and its associated expanded lambda matrix $\overline{A}$. Notice that $\kappa(\overline{A})$ is precisely $\kappa(b, \Lambda)$. Thus the stochastic block random is a special case of a Bernoulli random graph.

1.4.5 Stochastic Block Model

Another variant of the stochastic block model which does not fix the block sizes is as follows.

Definition A **stochastic block model** G, denoted $\kappa(n, \pi, \Lambda)$, has parameters: number of vertices $n \in \mathbb{N}$, **block probability vector** $\pi \in [0, 1]^K$ such that $\sum_{k \in K} \pi_k = 1$, and a matrix of edge probabilities $\Lambda \in [0, 1]^{K \times K}$. For all $v \in V$, the block membership function value $b(v)$ is distributed as i.i.d. Multinomial(π). In other words, for all $v \in V$ and $k \in \{1, 2, 3, \ldots K\}$, $\mathbb{P}[b(v) = k] = \pi_k$. Then, given the block membership function b, the edge probabilities between any vertices $\{v, v'\} \in \binom{V}{2}$ are independent $\mathbb{1}[\{v, v'\} \in E] \sim$ Bernoulli$(\lambda_{b(v)b(v')})$, as before. Notice that this makes $N = (n_1, n_2, \ldots, n_K) = (|V_1|, |V_2|, \ldots, |V_K|)$ a random vector.

1.4.6 Random Dot Product Graph

A further generalization on this variant of the random stochastic block model is the random dot product graph (RDPG).

Definition A **random dot product graph** $\kappa(n, \mathcal{F})$ has parameters: number of vertices $n \in \mathbb{N}$ and a distribution $\mathcal{F}$ on $\mathbb{R}^d$ such that for all $x, y \sim \mathcal{F}$, the inner product $\langle x, y \rangle \in [0, 1]$. Let the vertex set be $V = (v_1, v_2, \dots v_n)$, then for all $v_i \in V$, the corresponding **latent positions** x_i are distributed i.i.d. $\mathcal{F}$. Conditioned on the latent positions $x_1, x_2, \dots, x_n$, independently for all vertices $\{v_i, v_j\} \in \binom{V}{2}$, $\mathbb{1}[\{v_i, v_j\} \in E] \sim$ Bernoulli$(\langle x_i, x_j \rangle)$. For convenience, define the matrix $X = [x_1, x_2, \dots, x_n]^T \in \mathcal{R}^{n \times d}$.

Often it is convenient to assume that $\text{rank}(X^T X) = \text{rank}(X) = d$. However, in this model no choice of parameters is sufficient for this assumption.

Consider a distribution $\mathcal{F}$ with finite support, say the cardinality of the support of $\mathcal{F}$ is K. If for all $k \in \{1, 2, \dots, K\}$ we denote the k^{th} element in the support of $\mathcal{F}$ as z_k, and the corresponding probability as π_k, $\mathbb{P}[z_k] = \pi_k$, then for any $k, k' \in \{1, 2, 3, \dots K\}$, we can say $\lambda_{k,k'} = \langle z_k, z_{k'} \rangle$ and this model is equivalent to $\kappa(n, \pi, \Lambda)$.

Conversely, consider $\kappa(n, \pi, \Lambda)$ and assume Λ is positive semi-definite, say $\text{rank}(\lambda) = d$. Let the eigenvalue decomposition of Λ be $\mathcal{U}\Sigma\mathcal{U}^T$, where $\mathcal{U} \in \mathbb{R}^{n \times d}$ has orthogonal columns and $\Sigma \in \mathbb{R}^{d \times d}$ is diagonal. Since Λ is positive semi-definite, all the eigenvalues are non-negative and have real square roots. Then define $X' = [x'_1, x'_2, \dots, x'_K]^T = \mathcal{U}\Sigma^{\frac{1}{2}} \in [0, 1]^{K \times d}$. Say the latent position for $v_i \in V_k$ is the k^{th} row of X', and $\mathcal{F}$ is

defined with support $\{x'_k\}$ for $k \in \{1, 2, 3, \dots K\}$ with $\mathbb{P}[x'_k] = \pi_k$. Then we have a random dot product graph $\kappa(n, \mathcal{F})$.

In the setting where Λ is not positive semi-definite or G is directed, consider a distribution $\mathcal{F}$ on $\mathbb{R}^{2d}$, where each vertex $v_i \in V$ has two latent positions. One "left" latent position $x_i \in \mathbb{R}^d$ and another "right" latent position $y_i \in \mathbb{R}^d$ such that for all $v_i \in V$, $[x_i; y_i]$ is i.i.d. $\mathcal{F}$. Given the latent positions of every vertex, the probability of an edge between any pair of vertices $v_i, v_j \in V$ is, independent for all $\{v_i, v_j\} \in \binom{V}{2}$, $\langle x_i, y_j \rangle$, i.e. $\mathbb{P}[\mathbb{1}[v_i \sim v_j]] = \langle x_i, y_j \rangle$. Again for convenience, we collect the latent positions into matrices $X = [x_1, x_2, \dots, x_n]^T \in \mathcal{R}^{n \times d}$ and $Y = [y_1, y_2, \dots y_n]^T \in \mathcal{R}^{n \times d}$. Unless otherwise stated, we do not use this as our RDPG. Proofs in Chapters 4 and 5 require the assumption that Λ is positive semi-definite.

1.4.7 ρ-correlated Graphs

Informally, these are a pair of random graphs such that there is a fixed correlation across the two graphs. Correlation refers to the Pearson's product-moment coefficient.

Definition The **correlation** of two random variables X and Y is defined as

$$\text{corr}(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{E[X - \mu_X]E[Y - \mu_Y]}{\sigma_X \sigma_Y} = \frac{E[XY] - E[X]E[Y]}{\sqrt{E[X^2] - E^2[X]}\sqrt{E[Y^2] - E^2[Y]}}.$$

Definition Let $G^A = (V^A, E^A)$ and $G^B = (V^B, E^B)$ be two Bernoulli random graphs, where edge probabilities in each graph are independent. Let $V^A = \{v_1^A, v_2^A, \dots v_n^A\}$, $V^B = \{v_1^B, v_2^B, \dots v_n^B\}$. The two graphs G^A and G^B are **ρ-correlated** if there exists

a bijection $\varphi : V^A \mapsto V^B$ such that for all $i, j \in \{1, 2, 3, \dots n\}$,

$$\text{corr}(\mathbb{1}[v_i^A \sim v_j^A], \mathbb{1}[f(v_i^B) \sim f(v_j^B)]) = \rho.$$

Furthermore, this is the only dependence between the edges E^A and E^B.

We present a convenient procedure to produce two ρ-correlated Bernoulli random graphs.

Consider a hollow edge probability matrix $\overline{A}$ for a Bernoulli graph. We first generate a graph $G^A = (V^A, E^A) \sim \kappa(\overline{A})$, then generate another random graph $G^B = (V^B, E^B)$ in a specific manner. Then we show that $G^B \sim \kappa(\overline{A})$ and the graphs G^A, G^B are ρ-correlated with $\varphi(v_i^A) = v_i^B$. Conditioning on any event $\{v_i^A, v_j^A\} \in E^A$, let the probability that $\{v_i^B, v_j^B\} \in E^B$ be

$$\mathbb{P}[\{v_i^B, v_j^B\} \in E^B | \{v_i^A, v_j^A\} \in E^A] = \rho + (1 - \rho)\overline{a}_{ij}.$$

Still conditioning on any $\{v_i^A, v_j^A\} \in E^A$, this implies the probability that the non-edge $\{v_i^B, v_j^B\} \notin E^B$ is

$$\mathbb{P}[\{v_i^B, v_j^B\} \notin E^B | \{v_i^A, v_j^A\} \in E^A] = (1 - \rho)(1 - \overline{a}_{ij}).$$

Conditioning on every non-edge $\{v_i^A, v_j^A\} \notin E^A$, the corresponding non-edge $\{v_i^B, v_j^B\} \notin E^B$ with probability

$$\mathbb{P}[\{v_i^B, v_j^B\} \notin E^B | \{v_i^A, v_j^A\} \notin E^A] = \rho + (1 - \rho)(1 - \overline{a}_{ij}) = (1 - \overline{a}_{ij}) + \rho\overline{a}_{ij},$$

making the probability that the corresponding edges exists $\{v_i^B, v_j^B\} \in E^B$ with probability

$$\mathbb{P}[\{v_i^B, v_j^B\} \in E^B | \{v_i^A, v_j^A\} \notin E^A] = (1-\rho)\overline{a}_{ij}.$$

This concludes our procedure to generate G^B. Now we show G^A and G^B are ρ-correlated and $G^B \sim \kappa(\overline{A})$. Now let us compute the correlation of $\mathbb{1}[v_i^A \sim v_j^A]$ and $\mathbb{1}[v_i^B \sim v_j^B]$. Let X be the event $\{v_i^A, v_j^A\} \in E^A$ and Y be the event $\{v_i^B, v_j^B\} \in E^B$, then

$$\begin{aligned}
\text{corr}(X,Y) &= \frac{\mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]}{\sqrt{\mathbb{E}[X^2] - \mathbb{E}^2[X]}\sqrt{\mathbb{E}[Y^2] - \mathbb{E}^2[Y]}} \\
&= \frac{\mathbb{P}[XY] - \mathbb{P}[X]\mathbb{P}[Y]}{\sqrt{\mathbb{P}[X^2] - \mathbb{P}^2[X]}\sqrt{\mathbb{P}[Y^2] - \mathbb{P}^2[Y]}} \\
&= \frac{\mathbb{P}[XY] - \overline{a}_{ij}^2}{\overline{a}_{ij} - \overline{a}_{ij}^2}.
\end{aligned}$$

Now the only probability we need to compute is $\mathbb{P}[XY]$, which is

$$\begin{aligned}
\mathbb{P}[XY] &= \mathbb{P}[Y|X]\mathbb{P}[X] = \mathbb{P}[\{v_i^B, v_j^B\} \in E^B | \{v_i^A, v_j^A\} \in E^A]\mathbb{P}[\{v_i^A, v_j^A\} \in E^A] \\
&= (\rho + (1-\rho)\overline{a}_{ij})\overline{a}_{ij} = (\rho + \overline{a}_{ij} - \rho\overline{a}_{ij})\overline{a}_{ij} = \rho(\overline{a}_{ij} - \overline{a}_{ij}^2) + \overline{a}_{ij}^2.
\end{aligned}$$

Substituting back into the correlation formula yields

$$\text{corr}(X,Y) = \frac{\rho(\overline{a}_{ij} - \overline{a}_{ij}^2) + \overline{a}_{ij}^2 - \overline{a}_{ij}^2}{\overline{a}_{ij} - \overline{a}_{ij}^2} = \frac{\rho(\overline{a}_{ij} - \overline{a}_{ij}^2)}{\overline{a}_{ij} - \overline{a}_{ij}^2} = \rho.$$

Finally, we need to verify that the this new G^B graph is $\kappa(\overline{A})$,

$$\begin{aligned}
\mathbb{P}[\{v_i^B, v_j^B\} \in E^B] &= \mathbb{P}[\{v_i^B, v_j^B\} \in E^B | \{v_i^A, v_j^A\} \in E^A]\mathbb{P}[\{v_i^A, v_j^A\} \in E^A] \\
&\quad + \mathbb{P}[\{v_i^B, v_j^B\} \in E^B | \{v_i^A, v_j^A\} \notin E^A]\mathbb{P}[\{v_i^A, v_j^A\} \notin E^A] \\
&= (\rho + (1-\rho)\overline{a}_{ij})\overline{a}_{ij} + (1-\rho)\overline{a}_{ij}(1-\overline{a}_{ij}) \\
&= \rho\overline{a}_{ij} + (1-\rho)\overline{a}_{ij}\overline{a}_{ij} + (1-\rho)\overline{a}_{ij}(1-\overline{a}_{ij}) \\
&= \rho\overline{a}_{ij} + (1-\rho)\overline{a}_{ij}(\overline{a}_{ij} + 1 - \overline{a}_{ij}) = \rho\overline{a}_{ij} + (1-\rho)\overline{a}_{ij} \\
&= \overline{a}_{ij}(\rho + 1 - \rho) = \overline{a}_{ij}.
\end{aligned}$$

Thus we have now verified that G^A and G^B are ρ-correlated and both are $\kappa(\overline{A})$.

The joint distribution for two ρ-correlated Bernoulli trials, is fully determined by the Bernoulli parameters and ρ. Therefore the joint distribution is unique. Thus this is the only possible joint distribution for two ρ-correlated Bernoulli random graphs distributed as $\kappa(\overline{A})$. These ρ-correlated graphs are especially important for studying graph matching (see Section 2.4 and Chapter 5). These models describe a well defined way of creating dependency between two graphs, and provide theoretical framework for our proposed methods.

Chapter 2

Tools for Studying Random Graphs

In this chapter, we describe three tools that we use in subsequent chapters for graph inference. Section 2.1 introduces graph invariants, which are used as statistics in Chapter 3, for hypothesis testing. Specifically, the statistics are used for random graphs drawn from the dense sub-block random graph $\kappa(n_1, n_2, p, q)$. We conduct hypothesis testing with the null hypothesis $p = q$ and alternative hypothesis $p < q$. Section 2.2 covers the spectral partitioning method, which is used later in Chapter 4, for the task of vertex nomination, and in Chapter 5 as a part of the LSGM algorithm. Section 2.4 describes graph matching, which is also featured in Chapters 4 and 5.

2.1 Graph Invariants

This section introduces various graph invariants for use as test statistics in Chapter 3. We estimate their power in hypothesis testing, where the null hypothesis is an Erdős-Rényi random graph and the alternative is a dense sub-block random graph. In the following definitions, let $G = (V, E)$ be an undirected graph.

Definition The **order** is the number of vertices in the graph, denoted as $\text{order}(G) = |V|$, and the **size** is the number of edges in the graph, denoted as $\text{Size}(G) = |E|$.

Definition The **degree** of a vertex $v \in V$, $d(v)$ is the number of edges containing v, $d(v) = |\{(v, u) : u \in V, \{v, u\} \in E\}|$. The **maximum degree** $\delta(G)$ is the maximum degree of all vertices in a graph

$$\delta(G) = \max_{v \in V} d(v).$$

The maximum degree is a simple local graph invariant.

Definition The **average degree** of G is given by

$$\bar{d}(G) = \frac{1}{|V|} \sum_{v \in V} d(v) = \frac{2|E|}{|V|}.$$

The **maximum average degree** of G is given by

$$\text{MAD}(G) = \max_{H \subseteq G} \bar{d}(H),$$

where the maximum is over all subgraphs H of G.

Notice that it suffices to consider only induced subgraphs since for any subset of vertices $V' \subseteq V$, subgraphs of G on V' with smaller edge sets will have lower average degree than the subgraph induced by V'.

Scan statistics are a local graph invariant studied by Priebe et al. [8] for inference on the Enron graph.

Definition Recall that the k^{th} order neighborhood of a vertex $v \in V$ is the set of vertices $\{u \in V : d(u, v) \leq k\}$. Let $S_k(G)$ denote the k^{th} **order scan statistic**, which is the maximum number of edges over all graphs induced by k^{th} order neighborhoods,

$$S_k(G) = \max_{v \in V} \text{Size}(N_k(v; G)).$$

The scan statistic is very good for detecting high local activity in a graph. The primary limitation is that it is computationally expensive to compute.

Definition A **triangle** is defined as a clique of three vertices. The total **number of triangles** $\tau(G)$ is another graph invariant.

Definition To define **clustering coefficient**, first we define an **angle** as an induced subgraph on three vertices that has exactly two edges. Let the total number of induced subgraphs, which are angles in G, be denoted by angles(G), then the global **clustering coefficient** is given by

$$\text{CC}(G) = \frac{\tau(G)}{\tau(G) + \text{angles}(G)}.$$

Consider when graph G is a social network, where the vertices represent people, and the edges represent pairs of people who are friends. Over all instances where any person v has two distinct friends u and u', the clustering coefficient is the fraction of the time that u and u' are also friends of each other.

Definition The **average path length** is

$$\mathrm{APL}(G) = \frac{\sum_{\{v,v'\}\in\binom{V}{2}} d(v,v')}{n(n-1)}$$

As currently defined, all disconnect graphs have an infinite average path length. To avoid an infinite average path length, if there exists $v_i, v_j \in V$ such that there is no v_i, v_j path, then for average path length computation we use the convention $d(v_i, v_j) := 2\ \max_{i',j'} d(v_{i'}, v_{j'})$ over all $v_{i'}, v_{j'}$ that are connected.

Average path length graph invariant is a way to quantify the connectivity of the graph. It gives a sense of how far apart all the vertices are from each other. This is related to the popular concept of " six degrees of separation" also known as the "small world" phenomenon [37, 38]. Graphs with small average path length exhibit this phenomenon. This occurs in many real word graphs.

2.2 Adjacency Spectral Embedding

Here, we are primarily concerned with spectral embedding on the adjacency matrix. Alternative matrices such as the Laplacian and normalized Laplacian have also been studied in [18, 19] (see Section 2.2.6).

We first discuss the adjacency spectral embedding of directed graphs, then consider the undirected case. Adjacency spectral embedding requires an embedding dimension $\widehat{d} \in \mathbb{N}$ as a parameter. To compute the adjacency spectral embedding of a graph, first one computes the singular value decomposition of the adjacency matrix $A = \mathcal{U}\Sigma\mathcal{V}^T$, where $\mathcal{U}, \mathcal{V}$ are real orthogonal matrices, $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$ is a diagonal matrix, and $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n$ are singular values of A. The adjacency spectral embedding uses the first d columns of $\mathcal{U}$, call this sub-matrix $\widehat{\mathcal{U}}$, the first $\widehat{d}$ columns of $\mathcal{V}$, call this $\widehat{\mathcal{V}}$, and the leading $\widehat{d} \times \widehat{d}$ principle sub-matrix of Σ, $\widehat{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \ldots \sigma_{\widehat{d}})$. The adjacency spectral embedding of A is $\widehat{X} = \widehat{\mathcal{U}}\widehat{\Sigma}^{\frac{1}{2}} \in \mathbb{R}^{n \times \widehat{d}}$ and $\widehat{Y} = \widehat{\mathcal{V}}\widehat{\Sigma}^{\frac{1}{2}} \in \mathbb{R}^{n \times \widehat{d}}$, where each pair of rows $\widehat{x}_i$ of $\widehat{X}$ and $\widehat{y}_i$ of $\widehat{Y}$, represent the embedding of vertex v_i.

Algorithm 2.1 Adjacency Spectral Embedding (for directed graphs)

$\mathcal{U}\Sigma\mathcal{V}^T \leftarrow A$ ▷ Compute singular value decomposition

$\widehat{\mathcal{U}} \leftarrow [\mathcal{U}_1, \mathcal{U}_2, \ldots \mathcal{U}_{\widehat{d}}]$ ▷ Low rank approximation

$\widehat{\mathcal{V}} \leftarrow [\mathcal{V}_1, \mathcal{V}_2, \ldots \mathcal{V}_{\widehat{d}}]$

$\widehat{\Sigma} \leftarrow \text{diag}(\sigma_1, \sigma_2, \ldots \sigma_{\widehat{d}})$

$\widehat{X} \leftarrow \widehat{\mathcal{U}}\widehat{\Sigma}^{\frac{1}{2}}, \ \widehat{Y} \leftarrow \widehat{\mathcal{V}}\widehat{\Sigma}^{\frac{1}{2}}$

$\widehat{Z} \leftarrow [\widehat{X}, \widehat{Y}]$ ▷ Embedded Vertices

In the following lemma, if $G \sim \kappa(n, \mathcal{F})$, then the adjacency spectral embedding approximates the latent positions up to rotation [19].

Lemma 2.2.1 *Sussman et al. [19]*

Let $G \sim \kappa(n, \mathcal{F})$ such that $\mathcal{F}$ has finite support, i.e. $G \sim \kappa(n, \pi, \Lambda)$. Let the singular

value decomposition of Λ *be* $\mathcal{U}\Sigma\mathcal{V}^T$ *and call* $\mu = \mathcal{U}\Sigma^{\frac{1}{2}}$, $\nu = \mathcal{V}\Sigma^{\frac{1}{2}}$. *Truncating matrices* μ *and* ν *give us the adjacency spectral embedding of* A, $\widehat{Z} = [\widehat{X}, \widehat{Y}] \in \mathbb{R}^{n \times 2d}$, *where* $\widehat{X} = \widehat{\mathcal{U}}\widehat{\Sigma}^{\frac{1}{2}} \in \mathbb{R}^{n \times d}$ *and* $\widehat{Y} = \widehat{\mathcal{V}}\widehat{\Sigma}^{\frac{1}{2}} \in \mathbb{R}^{n \times d}$. *Also let* X, Y *be the left and right latent positions of the model respectively and define* $Z = [X, Y] \in \mathbb{R}^{n \times 2d}$. *Then for any sequence of graphs* $G_n \sim \kappa(n, \pi, \Lambda)$, $n = 1, 2, \ldots$, *there almost always exists an orthogonal matrix* $Q \in \mathbb{R}^{2d \times 2d}$ *such that*

$$\|\widehat{Z}Q - Z\|_F \leq \sqrt{2}\frac{\sqrt{6}}{\alpha^2\gamma^2}\sqrt{\frac{\log n}{n}} = O\left(\sqrt{\frac{\log n}{n}}\right),$$

where α, β, γ *are positive real numbers such that*

- *all eigenvalues of* Y^TY *and* X^TX *are greater than* α.
- *for all* $i, j \in \{1, 2, 3, \ldots n\}$, *call* μ_i *and* ν_i *the* i^{th} *column of* μ *and* ν *respectively, then* $\beta < \|\mu_i - \mu_j\|$, $\beta < \|\nu_i - \nu_j\|$.
- *for all* $i \in \{1, 2, 3, \ldots K\}$, $\gamma < \pi_i$.

This lemma states that the adjacency spectral embedding $\widehat{X}$ approaches the true latent positions X up to an orthogonal transformation "as n goes to infinity," and is the crux of the parallelization of seeded graph matching to make the large seeded graph matching algorithm (see Chapter 5).

2.2.1 Undirected Graphs

Now consider the case that G is undirected. Let the singular value decomposition of the adjacency matrix be $A = \mathcal{U}\Sigma\mathcal{V}^T$. Since A is symmetric, using the eigenvalue

decomposition of A, we have

$$A = W\Lambda W^T = \sum_{i=1}^{n} w_i \lambda_i w_i^T = \sum_{i=1}^{n} w_i |\lambda_i| \text{sign}(\lambda_i) w_i^T.$$

Thus the rows of $\widehat{X}$ and $\widehat{Y}$ differ at most by a sign. In many cases, we only need to consider $\widehat{X} = [\widehat{\mathcal{U}}\widehat{\Sigma}^{\frac{1}{2}}]$. For example, spectral partitioning improves by a factor of two in misclassification rate when only using $\widehat{X}$ [19].

Algorithm 2.2 Adjacency Spectral Embedding (for undirected graphs)

$\mathcal{U}\Sigma\mathcal{V}^T \leftarrow A$ ▷ Compute spectral decomposition

$\widehat{\mathcal{U}} \leftarrow [\mathcal{U}_1, \mathcal{U}_2, \ldots \mathcal{U}_{\widehat{d}}]$ ▷ Low rank approximation

$\widehat{\mathcal{V}} \leftarrow [\mathcal{V}_1, \mathcal{V}_2, \ldots \mathcal{V}_{\widehat{d}}]$

$\widehat{\Sigma} \leftarrow \text{diag}(\sigma_1, \sigma_2, \ldots \sigma_{\widehat{d}})$

$\widehat{X} \leftarrow [\widehat{\mathcal{U}}\widehat{\Sigma}^{\frac{1}{2}}]$ ▷ Embedded Vertices

2.2.2 Weighted Graphs

Another common variant of graphs are weighted graphs. Weighted graphs are graphs where the edges can be positive real valued instead of binary valued, i.e. $A \in \mathbb{R}_+^{n \times n}$ instead of $A \in \{0,1\}^{n \times n}$. Adjacency spectral embedding can be applied to weighted edges. It is also applicable to directed weighted graphs.

2.2.3 Dimension Selection

Let $G \sim \kappa(n, \pi, \Lambda)$ be a stochastic block model, or any other model where Λ is known. In order to perform adjacency spectral embedding, we need to pick an embedding dimension $\widehat{d}$. The embedding dimension $\widehat{d}$ we choose is rank(Λ), because we can construct a latent position with dimension rank(Λ). Say the singular value decomposition of Λ is $\mathcal{X}\Sigma\mathcal{Y}^T$, where $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{K \times \text{rank}(\Lambda)}$ and $\Sigma \in \mathbb{R}^{\text{rank}(\Lambda) \times \text{rank}(\Lambda)}$. Call the k^{th} column of $\mathcal{X}$ as $\mathcal{X}_k$, notice that $z_{b_i}^T = [\mathcal{X}_{b_i}\Sigma^{\frac{1}{2}}, \mathcal{Y}_{b_i}\Sigma^{\frac{1}{2}}]$ is a latent position for vertex v_i.

In practice, Λ is typically not known, and dimension selection is a hard problem. If the graph is drawn from a stochastic block model, and an upper bound of rank(Λ) is known, then embedding to a dimension equal to the upper bound is sufficient to estimate the latent position well (see Theorem 2.3.1) [20]. Choosing the embedding dimension is a hard problem that has been studied as in topics of dimension selection and model selection [39, 40]. We can also avoid selecting a dimension by using sparse representation techniques [41].

2.2.4 Unscaled Embedding

Another alternative is to use $\widehat{X} = \widehat{\mathcal{U}}$ directly without scaling by $\Sigma^{\frac{1}{2}}$. The consistency results of Lemma 2.2.1 still applies when using $\widehat{X} = \widehat{\mathcal{U}}$. Thus subsequent theorems based on this lemma also hold with the unscaled embedding. However, it

should be noted that Theorem 2.3.1 does not hold for unscaled embeddings [20]. The embeddings displays noticeable performance differences in finite samples. There exist scenarios where $\widehat{\mathcal{U}}$ performs better than $\widehat{\mathcal{U}}\Sigma^{\frac{1}{2}}$ and there exist other scenarios where $\widehat{\mathcal{U}}\Sigma^{\frac{1}{2}}$ performs better than $\widehat{\mathcal{U}}$.

2.2.5 Projection onto the Sphere

Projection onto the sphere is normalization following adjacency spectral embedding. Let $\widehat{X}$ be the embedding of a graph G and let $\widehat{X} = [x'_1, x'_2, \ldots, x'_d]^T$, where x_j denotes the j^{th} column of $\widehat{X}$. For all $j \in \{1, 2, 3, \ldots d\}$, define $\widetilde{x}_j = x'_j/\|x'_j\|$. The projection of $\widehat{X}$ onto the sphere is $\widetilde{X} = [\widetilde{x}_1, \widetilde{x}_2, \ldots, \widetilde{x}_d]$.

2.2.6 Laplacian

Another version of spectral embedding uses the Laplacian matrix instead of the adjacency matrix. The Laplacian of a graph is defined as $D - A$, where D is a diagonal matrix with diagonals d_{ii} being the degree of vertex v_i. An important and popular variant is the normalized Laplacian defined as

$$\mathcal{L} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}AD^{-1/2}.$$

For the stochastic block model, theorems analogous to Lemma 2.2.1 and Theorems 2.3.1, 2.3.2, and 2.3.3, have been proven using the Laplacian. Clustering after adjacency spectral embedding or Laplacian spectral embedding provides a consistent

estimation of the block membership function b. For finite samples, one embedding may outperform the other depending on the graph structure [19].

2.3 Spectral Partitioning

The next natural progression is to analyze the embedding and make inferences on the observed graph. Spectral partitioning is performing adjacency spectral embedding then clustering the embedded vertices. The clusters identify similar vertices in the graph. For a stochastic block random graph $\kappa(b, \Lambda)$, we are interested in identifying the block membership function b. The spectral embedding step requires an embedding dimension $\widehat{d}$. Using an embedding dimension of $d = \text{rank}(\Lambda)$ provides consistent estimation of the blocks, see Theorem 2.3.3 [47]. If the exact value of d is not known, results by Fishkind et al. [20] show that knowing an upper bound on d is sufficient for achieving good performance (see theorem 2.3.1).

Consider a graph G drawn from a stochastic block model. Let $X : \mathcal{U}\Sigma^{\frac{1}{2}}$ be the adjacency spectral embedding of the graph G. We cluster the embeddings using mean squared error as the objective

$$\min_{C \in \mathbb{R}^{n \times d}} \|X - C\|_F,$$

where there are at most K distinct-valued rows in C. Let C_i denote the rows of C and let $c'_1, c'_2, \ldots, c'_K$ denote the distinct-valued rows of C. The distinct-valued rows c'_k are called centroids and are the center of the K clusters. Since this objective

is not exactly solvable in practice, we use a clustering algorithm like the k-means algorithm [43] or the `R` package `mclust` [44–46] to approximate the solution. See [42] for a review of clustering methods.

The estimated block membership function for all $i \in \{1, 2, \ldots n\}$ is defined as $\widehat{b}_i = k$ if and only if $c'_k = c_i$.

Algorithm 2.3 Spectral Partitioning

$\mathcal{U}D\mathcal{V}^T \leftarrow A$ ▷ Compute singular value decomposition

$\widehat{\mathcal{U}} \leftarrow [\mathcal{U}_1, \mathcal{U}_2, \ldots \mathcal{U}_d]$ ▷ Low rank approximation

$\widehat{\mathcal{V}} \leftarrow [\mathcal{V}_1, \mathcal{V}_2, \ldots \mathcal{V}_d]$

$\widehat{\Sigma} \leftarrow \text{diag}(\sigma_1, \sigma_2, \ldots \sigma_d)$

$\widehat{X} \leftarrow [\widehat{\mathcal{U}}\widehat{\Sigma}^{\frac{1}{2}}, (\widehat{\Sigma}^{\frac{1}{2}}\widehat{\mathcal{V}})^T]$ ▷ Embedded Vertices

$C \leftarrow \min_C \|\widehat{X} - C\|_F$ ▷ Cluster Estimated Latent Positions

$c'_1, c'_2, \ldots, c'_K \leftarrow$ distinct rows of C ▷ Estimate Block Membership Function

for $n = 1, 2, \ldots, n$ **do**

$\widehat{b}_i \leftarrow \arg\min_k \|c'_k - x_i\|$

end for

Theorem 2.3.1 *Fishkind et al. [20]*

For $n = 1, 2, \ldots$, assume that the graphs G_n are random dot product random graphs distributed as $\kappa(n, \mathcal{F})$ whose $\mathcal{F}$ has finite support (i.e., a stochastic block model $\kappa(n, \pi, \Lambda)$). Further, assume all unknown parameters, and only assume that the embedding dimension $\hat{d} \geq$ rank(Λ). Let Ψ_K be the set of permutations on $\{1, 2, 3, \ldots K\}$.

For any fixed $\epsilon > \frac{3}{4}$*, for the sequence of graphs* $G_1, G_2, \ldots, G_n \sim \kappa(n, \pi, \Lambda)$*, the number of misalignments*

$$\min_{\varphi \in \Psi_K} |\{v_i \in V : b_i \neq \varphi(\hat{b}_i\}| < n^{\epsilon}$$

almost always.

Specifically, this theorem shows that the resulting clusters from spectral partitioning almost always have less than n^{ϵ} errors. The minimization over the set of permutations Ψ_K is used to permute the clusters to optimally match the true membership labels b.

Another consistency result proved by Sussman et al. [19] shows that the number of misalignments from spectral partitioning is only $O(\log n)$ vertices. This theorem requires the stronger assumption that the embedding dimension $\hat{d} = \text{rank}(\Lambda)$.

Theorem 2.3.2 *Sussman et al. [19]*

For $n = 1, 2, \ldots$*, assume that the graphs* G_n *are a stochastic block model* $\kappa(n, \pi, \Lambda)$ *or* $\kappa(n, \mathcal{F})$*, where* $\mathcal{F}$ *has finite support, and the embedding dimension is* $\hat{d} = rank(\Lambda)$*. For the sequence of graphs* G_n*,* $n = 1, 2, \ldots$*, where* $G_n \sim \kappa(n, \pi, \Lambda)$*, almost always holds that*

$$\min_{\varphi \in \Psi_K} |\{v_i \in V : b_i \neq \varphi(\hat{b}_i\})\| \leq \frac{2^3 3^2 6}{\alpha^5 \beta^2 \gamma^5} \log n$$

where, as in Lemma 2.2.1, α, β, γ *are positive real numbers such that*

- *all eigenvalues of* $Y^T Y$ *and* $X^T X$ *are greater than* α*.*
- *for all* $i, j \in \{1, 2, 3, \ldots n\}$*, call* μ_i *and* ν_i *the* i^{th} *column of* μ *and* ν *respectively, then* $\beta < \|\mu_i - \mu_j\|$*,* $\beta < \|\nu_i - \nu_j\|$*.*

- *for all* $i \in \{1, 2, 3, \dots K\}$, $\gamma < \pi_i$.

In the following theorem, Lyzinski et al. [47] has shown that spectral partitioning correctly identifies all block memberships, under some mild conditions.

Theorem 2.3.3 *Lyzinski et al. [47]*

Let $G \sim \kappa(n, \mathcal{F})$ *be a stochastic block model with* K *blocks and block membership vector* b. *Let the eigenvalues of* Λ *be* $\sigma_1, \sigma_2, \dots, \sigma_K$. *Let* $\eta \in (0, 1/2)$ *and*

$$\Delta = \max_{i \in [n]} \sum_{i \neq j} \overline{a}_{ij}, \qquad \gamma = \min_{i \in [d]} \frac{|\sigma_{i+1} - \sigma_i|}{n}$$
$$\beta = \frac{85 d \Delta^3 \log(n/\eta)}{(\gamma n)^{7/2}}.$$

Let $r > 0$ *be such that for all* $i, j \in \{1, 2, 3, \dots n\}$ *with* $x_i \neq x_j$, $\|x_i - x_j\|_2 > 4r$. *Let* $\widehat{X}$ *be the adjacency spectral embedding of* G *and* $\widehat{b} : \{1, 2, 3, \dots n\} \mapsto \{1, 2, 3, \dots K\}$ *be the optimal mean squared error clustering of the rows of* $\hat{X}$ *into* K *clusters. Let* Ψ_K *denote the set of permutations on* $\{1, 2, 3, \dots K\}$. *Finally, let* $n_{\min} = \min_{k \in \{1,2,3,\dots K\}} n_k$ *be the smallest block size. If* $r > \beta\sqrt{n/n_{\min}}$ *and* $\gamma n > 4\sqrt{\Delta \log(n/\eta)}$ *then with probability at least* $1 - 2\eta$,

$$\min_{\varphi \in \Psi_K} |\{i \in \{1, 2, 3, \dots n\} : b_i \neq \varphi(\hat{b}_i)\}| = 0.$$

Blocks are sometimes referred to as communities in the literature. Communities can have vertices which belong to multiple blocks. The problem of community detection is to identify which communities a vertex belongs to. Vertex partitioning is a procedure which can be used in community detection. See Fortunato [48], Now-

icki and Snijders [49], and Newman and Girvan [50] for a comprehensive survey on community detection.

2.4 Graph Matching

Definition Consider two graphs $G^A = (V^A, E^A)$ and $G^B = (V^B, E^B)$ such that $|V^A| = |V^B| = n$. For convenience, let $V^A = \{v_1^A, v_2^A, \ldots, v_n^A\}$, $V^B = \{v_1^B, v_2^B, \ldots, v_n^B\}$. The task of **graph matching** is finding a bijection $\varphi : V^A \mapsto V^B$ such that the number of edge disagreements is minimized after applying the bijection φ. An edge disagreement is defined as follows: either $\{v_i^A, v_j^A\} \in E^A$ and $\{\varphi(v_i^A, \varphi(v_j^A)\} \notin E^B$, or $\{v_i^A, v_j^A\} \notin E^A$ and $\{\varphi(v_i^A), \varphi(v_j^A)\} \in E^B$.

While numerous objective functions exist for matching graphs, we focus on the number of edge disagreements between E^A and E^B after applying the bijection φ. For possible alternative objective functions, see [4, 51–53].

We assume that there exists some underlying bijection φ^*. For example, let $G^A, G^B \sim \kappa(\overline{A})$, then the underlying bijection would be the function φ, where $\varphi(v_i^A) = v_i^B$ for all $i \in \{1, 2, 3, \ldots n\}$. In real data, suppose that we observed two social networks on the same collection of people (perhaps one is Facebook and the other is LinkedIn), then the alignment of the vertices corresponds to the same person. Note that the permutation which minimizes the number of edge disagreements is not necessarily the same as φ^* [61].

The simpler problem of graph isomorphism is the problem of determining if two graphs are isomorphic. The graph isomorphism problem is notoriously of unknown complexity [22–25]. In the case, where we allow $A, B \in \mathbb{R}^{n\times n}$ to be loopy, weighted, and directed, then graph matching is equivalent to the quadratic assignment problem (QAP), which is NP-hard [57].

Let A and B be adjacency matrices of two graphs $G^A,\ G^B$. Then some equivalent objective functions are

$$\underset{P\in\mathcal{P}_n}{\arg\min}\, \|AP - PB\|_F = \underset{P\in\mathcal{P}_n}{\arg\min}\, \|A - PBP^T\|_F, \tag{2.1}$$

where $\mathcal{P}_n$ is the set of all permutation matrices of size $n \times n$ and $\|\cdot\|_F$ is the Frobenius norm. The second objective function is equivalent to

$$\begin{aligned}
\|A - PBP^T\|_F^2 &= \mathrm{tr}((A - PBP^T)^T(A - PBP^T)) \\
&= \mathrm{tr}(A^TA - A^TPBP^T - PBP^TA + PBP^TBP^T) \\
&= \mathrm{tr}(A^TA) - \mathrm{tr}(A^TPBP^T) - \mathrm{tr}(PBP^TA) + \mathrm{tr}(PBP^TPBP^T) \\
&= \|A\|_F^2 - \mathrm{tr}(A^TPBP^T) - \mathrm{tr}((PBP^TA)^T) + \mathrm{tr}(P^TPBP^TPBP^TP) \\
&= \|A\|_F^2 - 2\mathrm{tr}(A^TPBP^T) + \|B\|_F^2.
\end{aligned}$$

Thus we have that

$$\underset{P\in\mathcal{P}_n}{\arg\min}\, \|A - PBP^T\|_F = \underset{P\in\mathcal{P}_n}{\arg\min}\, -\mathrm{tr}(A^TPBP^T). \tag{2.2}$$

These objective functions in Equations 2.2 and 2.1, are all the mathematically equivalent for P in the space of permutation matrices $\mathcal{P}_n$, but are different when this

domain is relaxed to the space of doubly stochastic matrices $\mathcal{D}$. In particular, we can optimally solve the relaxation $\arg\min_{P\in\mathcal{D}} \|AP - PB\|_F$ in polynomial time [54] (see Section 2.4.3), since it is a convex relaxation over linear constraints. This is in contrast to Equation 2.2, which is a non-convex relaxation. Unfortunately, this optimal relaxed solution to $\arg\min_{P\in\mathcal{D}} \|AP - PB\|_F$ is usually poorly related to the solution to $\arg\min_{P\in\mathcal{P}_n} \|AP - PB\|_F$ [55].

2.4.1 Frank-Wolfe Algorithm

Developed by Marguerite Frank and Philip Wolfe in 1956, the Frank-Wolfe algorithm [56] is a first-order optimization algorithm, which approximately solves

$$\min_{x\in\mathcal{X}} f(x),$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ is a polyhedron, and $f : \mathcal{D} \mapsto \mathbb{R}$ is continuously differentiable.

The first step is to pick an initial starting point in $x_0 \in \mathcal{X}$. For each iteration of the Frank-Wolfe algorithm, we first compute the gradient $\nabla f(x_t)$ to create a first order approximation. Next, we find the minimum s of the first order approximation to $f(\cdot)$ over the feasible region, $\mathcal{X}$. Then, we find the minimum on the line segment between s and the previous point x_t. The minimum on the line may be found by a line search, or alternatively it could be a shrinking predetermined step size. In our application we compute the optimal step size.

Algorithm 2.4 Frank-Wolfe Algorithm

$t \leftarrow 0$ ▷ Initialization

Select $x_0 \in \mathcal{X}$

while x_t has not converged **do**

$s_t \leftarrow \arg\min_{s \in \mathcal{X}} s^T \nabla f(x_t)$ ▷ Direction-Finding

$\alpha_t \leftarrow \arg\min_{0 \leq \alpha \leq 1} f(x_t + \alpha(s_t - x_t))$ ▷ Line Search

$x_{t+1} \leftarrow x_t + \alpha_t(s_t - x_t)$ ▷ Update

$t \leftarrow t + 1$

end while

2.4.2 Fast Approximate Quadratic Assignment Problem (FAQ)

The quadratic assignment problem is the graph matching objective, but for general matrices A, B, which is equivalent to minimizing

$$f(P) = -\text{tr}(A^T P B P^T). \tag{2.3}$$

The gradient of $f(P)$ is

$$\nabla f(P) = -APB^T - A^T PB, \tag{2.4}$$

and has Hessian

$$-B \otimes A - B^T \otimes A^T$$

where $\otimes$ means Kronecker product. Notice that the diagonals of the Hessian are zero. Thus the trace of the Hessian is also zero. The objective is not necessarily convex.

If A or B has a nonzero entry, i.e., at least one edge, then there exists one positive eigenvalue, and at least one negative eigenvalue.

A fast method to approximate the quadratic assignment problem (QAP) is proposed by Vogelstein et al. [34] using the Frank-Wolfe algorithm. The fast approximate QAP algorithm (FAQ) (see Algorithm 2.5) applies the Frank-Wolfe algorithm to the QAP problem. The QAP objective function is not convex; however, the fast QAP algorithm is still able to reach good solutions. The algorithm achieves better accuracy than existing state-of-the-art graph matching algorithms in most benchmarks [34].

Recall that in the Frank-Wolfe algorithm $\mathcal{X}$ is in $\mathbb{R}^n$ (a vector), but in our problem $\mathcal{X}$ is in $\mathbb{R}^{n\times n}$ (a square matrix). Our $\mathcal{X}$ can be thought of as a long vector in $\mathbb{R}^{n^2}$ that has been reshaped. The Frank-Wolfe process in FAQ is done as follows: The first step is an iteration of the Frank-Wolfe algorithm. In our case, is the direction-finding subproblem is formulated as

$$\min_{P'\in\mathcal{D}} \operatorname{tr}(P'^T \nabla f(P)) = \min_{P'\in\mathcal{D}} \operatorname{tr}(P'^T(-APB^T - A^T PB)). \tag{2.5}$$

Notice that Equation 2.5 is precisely the linear assignment problem (LAP). The solutions of the LAP for $P' \in \mathcal{P}_n$ and for the relaxed problems $P' \in \mathcal{D}$ are equivalent [57]. At the end of the Frank-Wolfe algorithm, $P \in \mathcal{D}$ and is not necessarily in $\mathcal{P}$. We project the final doubly stochastic matrix P into the space of permutations

$$\arg\min_{P'\in\mathcal{P}} P'^T P.$$

This is again the linear assignment problem and is efficiently solvable by the Hungar-

ian algorithm [58]. Modern variants achieve a computational complexity of $O(n^3)$ [59]. The Hungarian Algorithm is known for solving the minimum weight bipartite matching, which is equivalent to the linear assignment problem.

The optimal step size is obtained by solving

$$
\begin{aligned}
& \min_{\alpha\in[0,1]} f(\alpha P' + (1-\alpha)P) \\
= & \min_{\alpha\in[0,1]} -\mathrm{tr}(A^T(\alpha P' + (1-\alpha)P)B(\alpha P' + (1-\alpha)P)^T) \\
= & \min_{\alpha\in[0,1]} -\mathrm{tr}((\alpha A^T P' + (1-\alpha)A^T P)(\alpha BP' + (1-\alpha)BP)^T) \\
= & \min_{\alpha\in[0,1]} -\mathrm{tr}((\alpha A^T P' + (1-\alpha)A^T P)(\alpha P'^T B^T + (1-\alpha)P^T B^T)) \\
= & \min_{\alpha\in[0,1]} -\mathrm{tr}(\alpha^2 A^T P' P'^T B^T + \alpha(1-\alpha)(A^T PP'^T B^T + A^T P' P^T B^T) \\
& \qquad +(1-\alpha)^2 A^T PP^T B^T) \\
= & \min_{\alpha\in[0,1]} -\mathrm{tr}(\alpha^2 A^T P' P'^T B^T + (-\alpha^2+\alpha)(A^T PP'^T B^T + A^T P' P^T B^T) \\
& \qquad +(\alpha^2 - 2\alpha + 1)A^T PP^T B^T) \\
= & \min_{\alpha\in[0,1]} -\mathrm{tr}(\alpha^2(A^T P' P'^T B^T - A^T PP'^T B^T - A^T P' P^T B^T + A^T PP^T B^T) \\
& \qquad +\alpha(A^T PP'^T B^T + A^T P' P^T B^T - 2A^T PP^T B^T) + A^T PP^T B^T) \\
= & \min_{\alpha\in[0,1]} -\alpha^2\mathrm{tr}(A^T P' P'^T B^T - A^T PP'^T B^T - A^T P' P^T B^T + A^T PP^T B^T) \\
& \qquad -\alpha\mathrm{tr}(A^T PP'^T B^T + A^T P' P^T B^T - 2A^T PP^T B^T) \\
& \qquad +\mathrm{tr}(A^T PP^T B^T)
\end{aligned}
\tag{2.6}
$$

Equation 2.6 is quadratic in α and solvable using the quadratic formula. Then we update $P \leftarrow \alpha P' + (1-\alpha)P$ to complete one step of the Frank-Wolfe algorithm. We

repeat this until certain convergence criteria are achieved. Examples of convergence criteria include small step size, gradient value, a maximum number of iterations, etc. After the algorithm converges, the solution P may not necessarily be a permutation matrix. Thus, we find the closest permutation matrix P' to P,

$$\arg\min_{P' \in \mathcal{P}} \|P - P'\|_F = \arg\min_{P' \in \mathcal{P}} \operatorname{tr}(P'^T P). \tag{2.7}$$

Equation 2.7 can be solved by the Hungarian algorithm. The efficiency of the fast approximate QAP algorithm lies in identifying that the direction-finding subproblem is the LAP.

Algorithm 2.5 Fast Approximate QAP Algorithm

$t \leftarrow 0$ ▷ Initialization

$P_{ij}^{(0)} \leftarrow \frac{1}{n}$ for all $i, j \in \{1, 2, \ldots, n\}$

while $P^{(t)}$ has not converged **do**

$P'^{(t)} \leftarrow \arg\max_{P' \in \mathcal{P}_n} P'^T (A P^{(t)} B)$, via Hungarian Algorithm ▷ Direction-Finding

$\alpha \leftarrow \arg\min_{\alpha \in [0,1]} f(\alpha P'^{(t)} + (1-\alpha) P^{(t)})$ ▷ Line search

$P^{(t)} \leftarrow \alpha P'^{(t)} + (1-\alpha) P^{(t)}$ ▷ Update

end while

$P \leftarrow \arg\max_{P' \in \mathcal{P}_n} (P'^{(t)})^T P^{(t)}$ ▷ Project solution to permutation matrix

This algorithm is capable of matching graphs up to thousands of vertices. Scalability of graph matching is explored later in Chapter 5.

2.4.2.1 Seeded Graph Matching

One way to improve the performance of the graph matching algorithm is to incorporate knowledge of partial alignment between corresponding vertices. Recall that vertices with known alignments are called seed vertices (Section 1.4.3). Fishkind et al. [60] proposed incorporating seed vertices in fast approximate QAP (FAQ). This is the problem of seeded graph matching. Seed vertices do not significantly alter the FAQ algorithm, but they improving the matching accuracy and runtime.

Let $G^A = (U^A \cup W^A, E^A)$ and $G^B = (U^B \cup W^B, E^B)$ be two graphs with adjacency matrices A and B respectively, where U^A, U^B are the seed vertex sets and W^A, W^B are the non-seed vertex sets. We subdivide A and B as

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \; B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix},$$

where the matrices $A_{11}, B_{11} \in \mathbb{R}^{s \times s}$ are the adjacency matrices of $G^A[U^A], G^B[U^B]$ respectively, and the matrices $A_{22}, B_{22} \in \mathbb{R}^{m \times m}$ are the adjacency matrices of $G^A[W^A], G^B[W^B]$ respectively. The remaining matrices $A_{12}, B_{12} \in \mathbb{R}^{s \times m}$, $A_{21}, B_{21} \in \mathbb{R}^{m \times s}$ indicate the adjacencies between the seed and non-seed vertices.

We can incorporate the known mapping between the seed vertices into the graph matching objective. Without loss of generality, let the true alignment φ for the first s vertices be trivial, $\varphi(v_i^A) = v_i^B$ for all $i \in \{1, 2, 3, \ldots s\}$. For convenience let the seed vertices be vertices $v_1, v_2, \ldots v_s$. Thus the seeded graph matching objective is

$$\arg\min_P \|A - (I_{s \times s} \oplus P) B (I_{s \times s} \oplus P)^T\|_F,$$

over all $n \times n$ permutation matrices P, where $I_{s\times s}$ is the $s \times s$ identity matrix and $\oplus$ is the direct sum. Similar to the original objective in Equation 2.3, the new objective can be rewritten as

$$\begin{aligned} &= \arg\min_{P\in\mathcal{P}_m} \|A\|_F - 2\mathrm{tr}(A^T(I_{s\times s}\oplus P)B(I_{s\times s}\oplus P)^T) + \|B\|_F \\ &= \arg\min_{P\in\mathcal{P}_n} -\mathrm{tr}(A^T(I_{s\times s}\oplus P)B(I_{s\times s}\oplus P)^T). \end{aligned}$$

Let $f(P)$ be the objective function. We have

$$\begin{aligned} f(P) &= \mathrm{tr}\left(A^T(I_{s\times s}\oplus P)B(I_{s\times s}\oplus P)^T\right) \\ &= \mathrm{tr}\left(\begin{bmatrix} A_{11}^T & A_{21}^T \\ A_{12}^T & A_{22}^T \end{bmatrix}\begin{bmatrix} I_{s\times s} & 0_{s\times m} \\ 0_{s\times m} & P \end{bmatrix}\begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}\begin{bmatrix} I_{s\times s} & 0_{s\times m} \\ 0_{s\times m} & P^T \end{bmatrix}\right) \\ &= \mathrm{tr}\left(\begin{bmatrix} A_{11}^T & A_{21}^T \\ A_{12}^T & A_{22}^T \end{bmatrix}\begin{bmatrix} B_{11} & B_{12}P^T \\ PB_{21} & PB_{22}P^T \end{bmatrix}\right) \\ &= \mathrm{tr}A_{11}^TB_{11} + \mathrm{tr}A_{21}^TPB_{21} + \mathrm{tr}A_{12}^TB_{12}P^T + \mathrm{tr}A_{22}^TPB_{22}P^T. \end{aligned}$$

Thus,

$$f(P) = \mathrm{tr}A_{11}^TB_{11} + \mathrm{tr}P^TA_{21}B_{21}^T + \mathrm{tr}P^TA_{12}^TB_{12} + \mathrm{tr}A_{22}^TPB_{22}P^T, \tag{2.8}$$

which has gradient

$$\nabla f(P) = A_{21}B_{21}^T + A_{12}^TB_{12} + A_{22}PB_{22}^T + A_{22}^TPB_{22}.$$

Notice that the first two term of the, gradient is a constant. The remaining part is the FAQ gradient part, which is very similar to the original gradient, except on A_{22} and B_{22} rather than A and B.

2.4.3 Other Graph Matching Algorithms

There are numerous other graph matching approximation algorithms. Most of these algorithms currently do not have an immediate handling seed vertices. We will examine the performance of these algorithms in Chapter 5. In this section, we briefly explain each algorithm. All of these algorithms are implemented in the `graphm` package, written by M. Zaslavskiy, F. R. Bach, J.-p. P. Vert, [54].

2.4.3.1 U

The Umeyama algorithm, referred to as U here, uses eigenvalue decomposition to approximate the permutation of vertices [32]. Given the adjacency graphs of two graphs A and B, let the eigenvalue decompositions of A and B be $A = \mathcal{U}_A \Sigma_A \mathcal{U}_A^T$ and $B = \mathcal{U}_B \Sigma_B \mathcal{U}_B^T$, respectively. Then we compute $\mathcal{U}_A \mathcal{U}_B^T$, which gives us a unitary matrix. We subsequently apply the Hungarian algorithm to project this matrix to a permutation matrix.

2.4.3.2 rank

Rohit Singh, JinboXu, and Bonnie Berger developed the IsoRank algorithm [62], which is referred to as rank. IsoRank has many similarities to the Google PageRank algorithm. First we compute the scores between every vertex from graph G^A and every vertex G^B. Then, the best alignment is determined via an eigenvalue decomposition involving the scores.

2.4.3.3 QCV

The QCV algorithm [54] is similar to the FAQ algorithm, which is discussed in Section 2.4.2. The difference between the algorithm FAQ and QCV is that the QCV has objective $\|AP - PB\|_F$. When relaxed to doubly stochastic matrices, we can solve this via the Frank-Wolfe algorithm in polynomial time. Often the matrix P is not a permutation matrix. Hence, the Hungarian algorithm is used to find the closest permutation matrix, just as in FAQ. It is important to keep in mind that the true alignment is almost always not a solution to the QCV objective [55].

2.4.3.4 PATH

The PATH algorithm iteratively solves convex combination of two objectives [54]. One is the same as the QCV convex objective, say F_0. The other is a concave objective derived from the same objective, say F_1. The objective that they solve for is $\min_P F_\lambda(P) = (1-\lambda)F_0(P) + \lambda F_1(P)$. Starting with $\lambda = 0$, the objective is minimized with any quadratic programing algorithm. For sufficiently small $d\lambda$, one uses Frank-Wolfe to solve $\min_P F_{\lambda+d\lambda}(P)$, with previous solution $\min F_\lambda$ as the initial starting point. The final solution returned is when $\min_P F_1(P)$.

2.4.3.5 GLAG

The GLAG is an approximate graph matching algorithm based on sparsity techniques [63]. The optimization is solved by augmented Lagrangian techniques. GLAG

is a fairly involved algorithm. The details of the algorithm can be found in [63].

2.5 Mixed Membership Stochastic Block-model

In this section, we present the mixed membership stochastic block model (MMB) [16]. This model is useful in identifying vertices belonging to a mixture of blocks, and is a modification of latent Dirichlet allocation (LDA) [64]. LDA is widely used in topic modeling in the field of natural language processing.

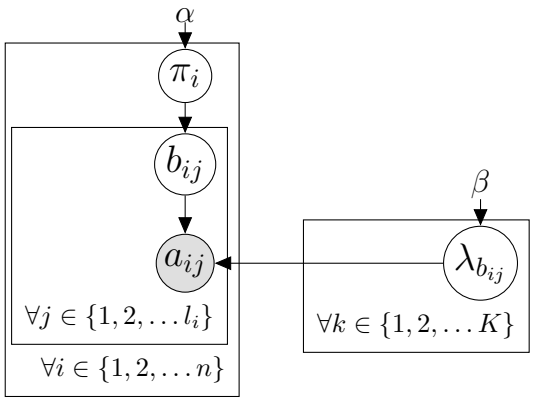

Figure 2.1: Plate notation of Latent Dirichlet Allocation.

Here, we first present the LDA model in the standard topic modeling setting. Suppose there are n documents and K topics. For each document $i \in \{1, 2, 3, \dots n\}$, suppose document i have l_i words. Each document can contain words from multiple topics, and each word is related to only one topic. Let π_i be the distribution of

topics in document i, let a_{ij} be the j^{th} word in document i, let b_{ij} be the topic of the j^{th} word in document i, and let $\lambda_k \in [0,1]^d$ be the distribution of words for topic $k \in \{1, 2, 3, \dots K\}$, where d is the number of words in the set of all words (dictionary). There are two hyper-parameters of LDA, say α and β. The generative procedure to obtain the n documents with k topics is

- For each topic $k \in \{1, 2, \dots K\}$:
  - Draw a word distribution $\lambda_k \sim$Dirichlet(β).
- For each document $i \in \{1, 2, \dots n\}$:
  - Draw a topic distribution $\pi_i \sim$Dirichlet(α).
  - For each word $j \in [l_i]$:
    - Sample a topic $b_{ij} \sim$Multinomial(π_i).
    - Sample a word $a_{ij} \sim$Multinomial($\lambda_{b_{ij}}$).

The parameters of LDA can be trained with variational Bayes [64]. Thomas Griffiths, and Mark Steyvers [65] performed inference of the parameters via Gibbs sampling. Thomas Minka and John Lafferty [66] performed inference via expectation-propagation.

Now we present the mixed membership stochastic block-model [16], a modification of LDA. There are two main changes to the model. First, for all $i \in \{1, 2, 3, \dots n\}$ the number of words in each document, $l_i = n$. Second, there is the hyper-parameter

α and an edge probability matrix $\Lambda \in [0,1]^{K\times K}$. Let $G = (V, E)$ be a graph with $|V| = n$. The generative procedure for MMB is

- For each $i \in \{1, 2, 3, \dots n\}$:
  - Draw a mixed membership distribution for vertex v_i, π_i ~Dirichlet(α).
  - For each $j \in \{1, 2, 3, \dots n\}$:
    - Sample vertex v_i's block membership for edge (i, j), b_{ij} ~Multinomial(π_i).
    - Sample vertex v_j's block membership for edge (i, j) b_{ji} ~Multinomial(π_j).
    - Sample edge $\mathbb{1}[i \sim j]$ ~Bernoulli($\lambda_{b_{ij},b_{ji}}$).

The parameters of this model are trained using a nested variational inference algorithm.

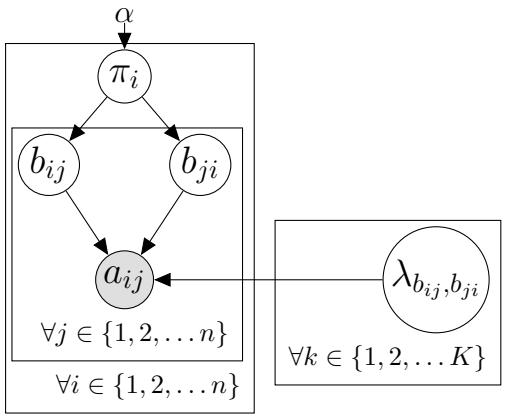

Figure 2.2: A plate notation of mixed membership stochastic block model.

Chapter 3

Statistical Inference for Dense Sub-community Detection

In this section, we present a comparative power analysis of various graph invariants for testing the hypothesis that a kidney-egg random graph has a subgraph with higher edge probability [67]. Given a graph drawn from kidney-egg model, the null hypothesis is that all edge probabilities are equal. The alternative hypothesis is that there exists subset of vertices with higher edge probability than other edges in the graph. Using Monte Carlo simulations, we estimate the power of the tests using graph invariants as test statistics. We discovered that for many choices of parameters in the random graph model, the scan statistic and clustering coefficient often dominate our other graph invariants. However, our results indicated that none of the graph invariants considered is uniformly most powerful. Graphs in this section are assumed

to be simple and distributed as $\kappa(n_1, n_2, p, q)$. For convenience, let $n = n_1 + n_2$.

3.1 Hypothesis Test for Dense Sub-community Detection

A hypothesis test is a statistical method to determine if there is enough evidence for rejecting the null hypothesis (the de-facto belief) in favor of the alternative hypothesis. For the purpose of our statistical inference, we design a hypothesis test to determine whether the given graph is Erdős-Rényi ($p = q$) or dense sub-block ($p < q$) random graph.

3.1.1 Null Hypothesis, $\kappa(n_1 + n_2, p)$

The null hypothesis H_0 is $p = q$ for an $\kappa(n_1, n_2, p, q)$ graph. This is an Erdős-Rényi $\kappa(n_1 + n_2, p)$ random graph with $n = n_1 + n_2$ vertices and edge probability p.

3.1.2 Alternative Hypothesis, $\kappa(n_1, n_2, p, q)$

The alternative hypotheses H_A is a kidney egg graph $\kappa(n_1, n_2, p, q)$ such that $q > p$, i.e. a dense sub-block graph model. The edges in the egg have higher edge probability q, and the remaining graph has edge probability p. The comparative power study's goal is to quantify the ability of graph invariants to distinguish between a homoge-

neous Erdős-Rényi random graph and a graph with higher local communication.

3.1.2.1 Type I Error

To determine the ability of a statistic to distinguish between the null and alternative hypothesis, we have to pick an acceptable type I error, denoted α. The type I error is the probability of rejecting the null hypothesis under the null hypothesis. In a hypothesis test, we set α to a specific value, such as 0.05 or 0.01. This α is referred to as a level of significance of the test.

3.1.2.2 Power

The power of a statistic is the probability of rejecting the null hypothesis under the alternative hypothesis, denoted β. Here, we estimate the power of various statistics using Monte-Carlo simulation.

If $T(G)$ is a statistic calculated from observed graph G, we compute the power of a statistic with approximate level of significance α from R Monte Carlo samples as follows. First we generate R independent, identically distributed (i.i.d.) graphs $G_1, G_2, \ldots G_R$ under H_0, and generate R i.i.d. graphs $G_1^A, G_2^A, \ldots G_R^A$ under H_A. Next, we compute $T_r = T(G_r)$, for each $r \in \{1, 2, 3, \ldots R\}$ and compute the order statistics $T_{(1)} \leq T_{(2)} \leq \ldots \leq T_{(R)}$. The rejection region of this test of hypothesis is $T(G) >$

$T_{(R(1-\alpha))}$ and the empirical power is

$$\widehat{\beta} = \frac{1}{R}\sum_{r=1}^{R} \mathbb{1}[T(G_r^A) > T_{(R(1-\alpha))}].$$

Further details on estimating the power can be found in [68].

3.2 Synthetic Experiments

In synthetic experiments, we examine the power of graph invariants introduced in section 2.1. We focus on one particular experiment.

3.2.1 Experiment Design

This section presents the Monte Carlo power results for seven graph invariants: size, max degree, maximum average degree, scan statistic, number of triangles, and average path length. The null hypothesis is $H_0 : \kappa(1000, 0.1)$, and the alternative hypothesis is $H_A : \kappa(950, 50, 0.1, 0.5)$. We proceed to design and execute a Monte Carlo experiment to generate comparative power results.

3.2.2 Monte Carlo Simulations

For each graph invariant from Section 2.1, we present Monte-Carlo histograms of the null and alternative hypotheses and plot the critical region and the actual and asymptotic distribution (if known).

3.2.2.1 Size

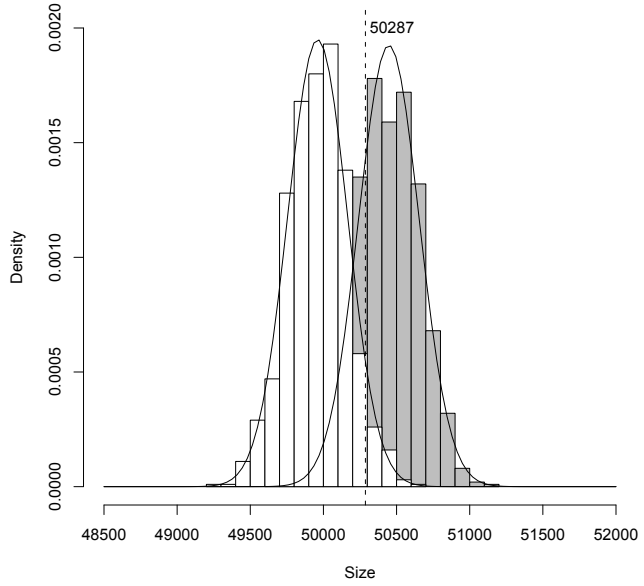

Figure 3.1: Hypothesis test for number of edges when the null hypothesis is H_0 : $\kappa(1000, 0.1)$, and the alternative hypothesis is $H_A : \kappa(950, 50, 0.1, 0.5)$.

Size is a simple graph invariant measuring global graph activity, so we do not expect size to perform well when n_2 is small. For an Erdős-Rényi random graph $G \sim \kappa(n, p)$, the size of a graph is distributed as a binomial random variable

$$P[\text{size}(\kappa(n,p)) = k] = \binom{\binom{n}{2}}{k} p^k (1-p)^{\binom{n}{2}-k}.$$

In a kidney egg random graph $\kappa(n - m, m, p, q)$, the size of a graph is distributed as the sum of 2 binomial random variables. The egg edges have probability Binomial($\binom{m}{2}, q$) and the remaining kidney edges have probability Binomial($\binom{n}{2}$ –

$\binom{m}{2}, p)$,

$$P[\text{size}(\kappa(n-m, m, p, q)) = k] = \sum_{i=0}^{k} \left[\binom{\binom{m}{2}}{k-i} q^{k-i} (1-q)^{\binom{m}{2}-k+i} \binom{\binom{n}{2} - \binom{m}{2}}{i} p^{i} (1-p)^{\binom{n}{2}-\binom{m}{2}-i} \right].$$

3.2.2.2 Max Degree

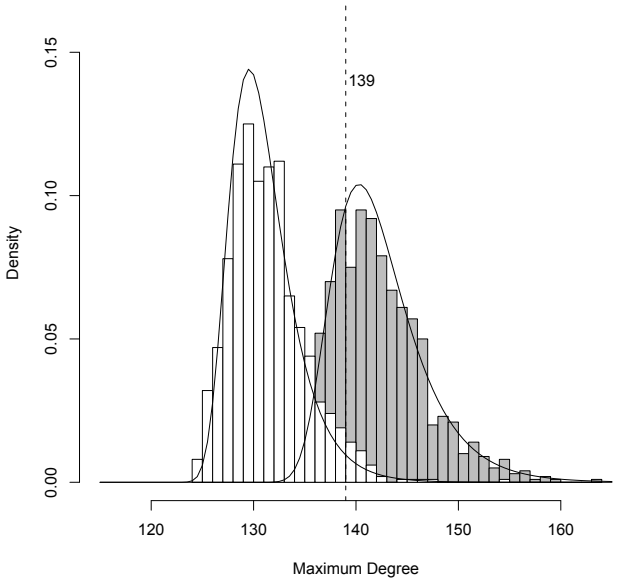

Figure 3.2: Hypothesis test for maximum degree when the null hypothesis is H_0 : $\kappa(1000, 0.1)$, and the alternative hypothesis is $H_A : \kappa(950, 50, 0.1, 0.5)$.

Maximum degree is a simple local graph invariant. The exact distribution is complex. However the limiting distribution of $\delta(\kappa(n, p))$ is a Gumbel(a, b) [9]

$$f_{\delta(\kappa(n,p))}(d) \to \frac{1}{b} \exp\left[-\frac{d-a}{b} - \exp\left(\frac{d-a}{b} \right) \right],$$

where

$$\begin{aligned} a &= pn + \sqrt{2p(1-p)n\log n}\left(1 - \frac{\log\log n}{4\log n} - \frac{\log(2\sqrt{\pi})}{2\log n}\right), \\ b &= \frac{\sqrt{2p(1-p)(n-1)\log n}}{2\log n}. \end{aligned}$$

The maximum degree of the model $\kappa(n-m, m, p, q)$ is also distributed as a Gumbel(a, b) when $m = \Omega(\sqrt{n})$,

$$\begin{aligned} a &= qm + p(n-m) + \sqrt{mq(1-q)2\log m}\left(1 - \frac{\log\log n}{4\log n} - \frac{\log(2\sqrt{\pi})}{2\log n}\right), \\ b &= \frac{\sigma_{E+F}}{\sqrt{2\log m}}, \end{aligned}$$

$E \sim$Binomial$(m-1, q)$, $F \sim$Binomial$(n-m, p)$, and σ_{E+F} is the standard deviation of the sum of $E + F$ [69].

3.2.2.3 Maximum Average Degree

Exact computation of MAD is possible in polynomial time, also known as finding the maximum density subgraph [70]. The algorithm solves min-cut max-flow a logarithmic number of times. This invariant is difficult to implement so instead we consider two approximations to maximum average degree; $MAD_g(G)$ and $MAD_e(G)$, which we define next.

We consider a simple greedy algorithm $MAD_g(G)$ to estimate the maximum average degree of a graph. At each iteration, this algorithm greedily chooses the vertex with the smallest degree to remove from the graph. At each iteration we compute the average degree of the induced subgraph. When there are no vertices left, the largest

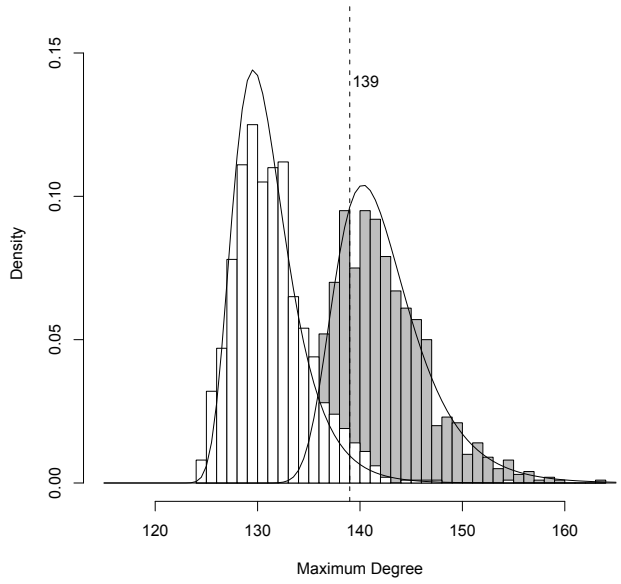

Figure 3.3: Hypothesis test for maximum average degree when the null hypothesis is $H_0 : \kappa(1000, 0.1)$, and the alternative hypothesis is $H_A : \kappa(950, 50, 0.1, 0.5)$.

average degree is returned as $MAD_g(G)$. This simple greedy procedure approximates maximum average degree (Scheinerman and Ullman) [71].

Another approximation to maximum average degree is the largest eigenvalue of the adjacency matrix, denoted as $MAD_e(G)$. Rayleigh-Ritzs Theorem [72] states that if A is Hermitian, the largest eigenvalue of A is $d_{\max}$, and the smallest eigenvalue of A is $d_{\min}$, then for all $x \in \mathbb{C}^n$

$$\lambda_{\max} = \max_{x \in \mathbb{R}^n : x \neq 0} \frac{x^T A x}{x^T x},$$
$$\lambda_{\min} = \min_{x \in \mathbb{R}^n : x \neq 0} \frac{x^T A x}{x^T x}.$$

Let $A = [a_{ij}]$ be the adjacency matrix for graph G. Consider restricting x to

be $x \in \{0,1\}^n$, then $\frac{x^T A x}{x^T x}$ is the average degree of the related induced subgraph wherein $x_i = 1$ if and only if v_i is present in the induced subgraph. This implies that $MAD_e(G) = d_{\max} \geq MAD(G)$.

In experiments comparing the power of MAD_e and MAD_g, MAD_e shows better performance. Figure 3.4 shows $\beta(MAD_e) - \beta(MAD_g)$ for $n = 1000, p = 0.1$ and various m, q, i.e. $\kappa(1000-m, m, 0.1, q)$. For each value of m and q, $R = 1000$ replicates where used. This demonstrates (in one example) how MAD_e compares with MAD_g.

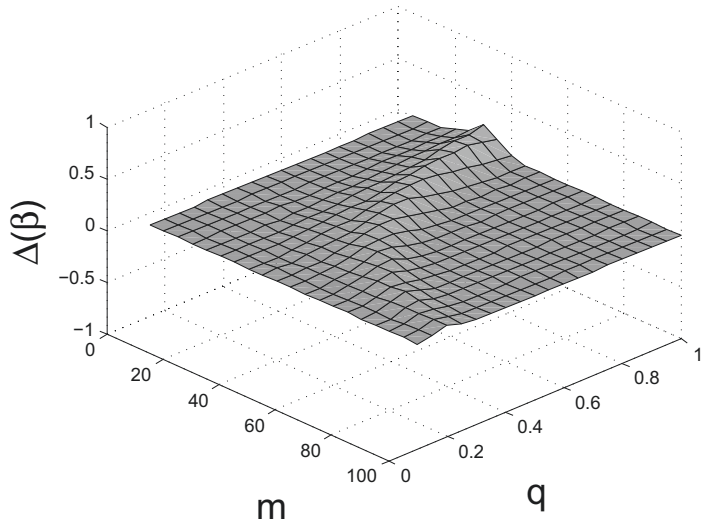

Figure 3.4: Statistical power difference surface for $\beta(MAD_e) - \beta(MAD_g)$ with $n = 1000$ and $p = 0.1$ over a range of $(m, q) \in \Theta_A$, via Monte Carlo. $\beta(MAD_e)$ dominates $\beta(MAD_g)$ in this space of parameters.

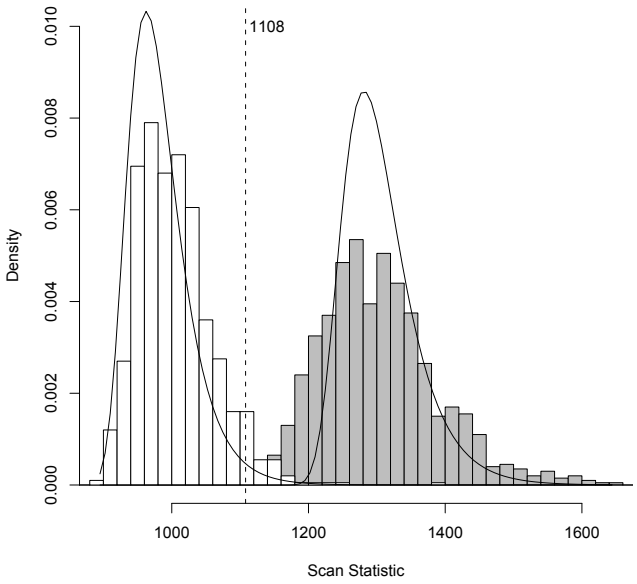

Figure 3.5: Hypothesis test for scan statistic.

3.2.2.4 Scan Statistic

Recall that we consider the k^{th} order neighborhood for a k^{th} order scan statistic. Limited by computation power, we will only consider the case $k = 1$. The first order scan statistic is very similar to maximum degree, except that the scan statistic includes neighbor-to-neighbor edges. As with maximum degree, the limiting distribution is shown to be a Gumbel(a, b) for an Erdős-Rényi $\kappa(n, p)$ graph by Rukhin [35]

$$a = \frac{1}{2}p^3n^3 + p^2\sqrt{p(1-p)n^3}\sqrt{2\log n}\left(1 - \frac{\log\log n - \log(4\pi^2)}{4\log n}\right),$$
$$b = \frac{p^2\sqrt{p(1-p)n^3}}{\sqrt{2\log(n)}}.$$

The limiting distribution for a $\kappa(n-m, m, p, q)$ random graph is also know to be

a Gumbel,

$$a = N_{n,p,m,q} + p\binom{N_{n,p,m,q}}{2} + (q-p)\binom{\mu_E}{2}$$
$$b = \left(1 - \frac{p}{2} + pN_{n,p,m,q}\right)\frac{\sigma_{E+F}}{\sqrt{2\log m}},$$

where $E \sim$Binomial$(m-1, q)$, $F \sim$Binomial$(n-m, p)$, μ_{E+F} and σ_{E+F} are the mean and standard deviation of the sum of $E+F$ respectively, and

$$\begin{aligned} N_{n,p,m,q} &= \mu_{E+F} + z_m\sigma_{E+F} \\ z_m &= \sqrt{2\log m}\left(1 - \frac{\log\log m}{4\log m} + \frac{\log(2\sqrt{\pi})}{2\log m}\right). \end{aligned}$$

This is also shown by Rukhin [73]

3.2.2.5 Number of Triangles

The number of triangles can be computed as

$$\tau(G) = \frac{\mathrm{tr}(A^3)}{6},$$

where A is the adjacency matrix of G.

Note that the i^{th}, j^{th} entry of A^3 is the number of length-three walks from i to j. Thus $\mathrm{tr}(A^3)$ is the number of all three closed walks, starting and ending at the same vertex. The unlabeled triangles are over counted exactly 6 times. Three times for starting at any of the three vertices times two times for each traversed orientation (clockwise or counter-clockwise).

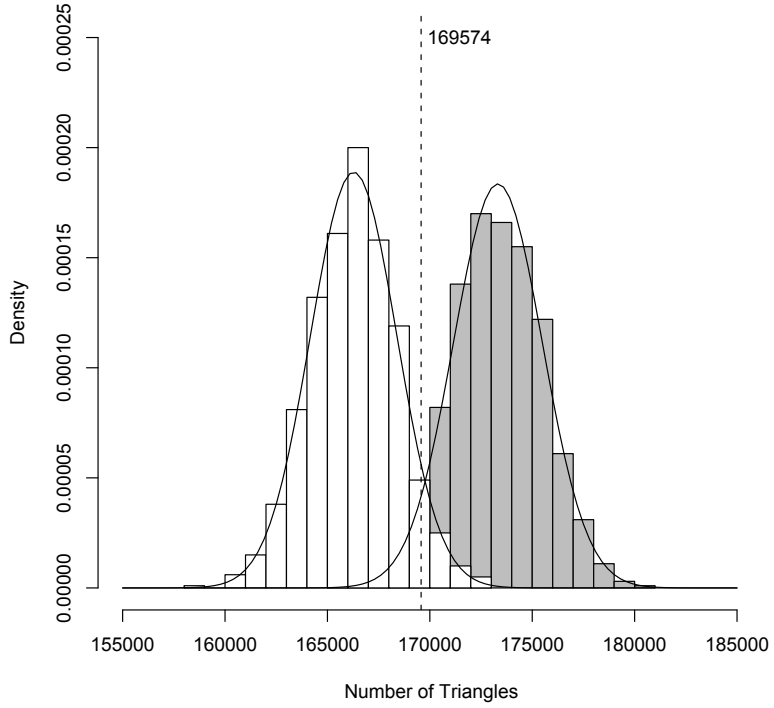

Figure 3.6: Hypothesis test for number of triangles when the null hypothesis is H_0 : $\kappa(1000, 0.1)$, and the alternative hypothesis is $H_A : \kappa(950, 50, 0.1, 0.5)$.

For random graph $G \sim \kappa(n,p)$ Nowicki and Wierman [74] show that when $\tau(G)$ is appropriately normalized, $\tau(G)$ is asymptotically normal. Specifically

$$\tau \sim \text{normal}\left(\binom{n}{3}p^3, \left[\binom{n-2}{1}p^2\right]^2 \binom{n}{2}p(1-p)\right)$$

Rukhin [35] shows $\tau(G)$ is also asymptotically normal when $G \sim \kappa(n-m,m,p,q)$.

$$\begin{aligned}\tau \sim \text{normal}\Bigg(&\binom{m}{3}q^3 + \binom{m}{2}\binom{n-m}{1} + \left[\binom{m}{1}\binom{n-m}{2} + \binom{n-m}{3}\right]p^3, \\ &\binom{m}{2}q(1-q)\left[\binom{m-2}{1}q^2 + \binom{n-m}{1}p^2\right]^2 \\ &+\binom{m}{1}\binom{n-m}{1}p(1-p)\left[\binom{m-1}{1}pq + \binom{n-m-1}{1}p^2\right]^2 \\ &+\binom{n-m}{2}p(1-p)\left[\binom{n-2}{1}p^2\right]^2\Bigg)\end{aligned}$$

3.2.2.6 Clustering Coefficient

Remember that the clustering coefficient is a measure of "closure" in a graph.

3.2.2.7 Average Path Length

Recall that average path length is a measure of the "small world" phenomenon.

3.2.3 Power Relationship with n_2, q

In this section, we present the power surface plots of 8 graph invariants described in the previous Section in figure 3.9. The experiments use the random graph model

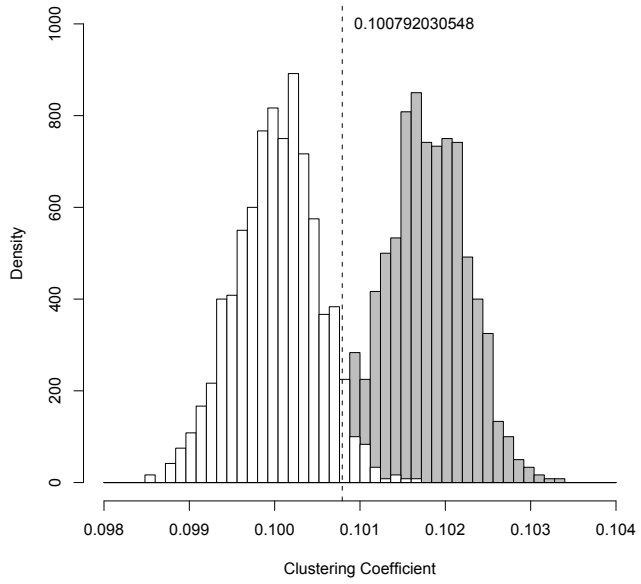

Figure 3.7: Hypothesis test for clustering coefficient when the null hypothesis is $H_0 : \kappa(1000, 0.1)$, and the alternative hypothesis is $H_A : \kappa(950, 50, 0.1, 0.5)$.

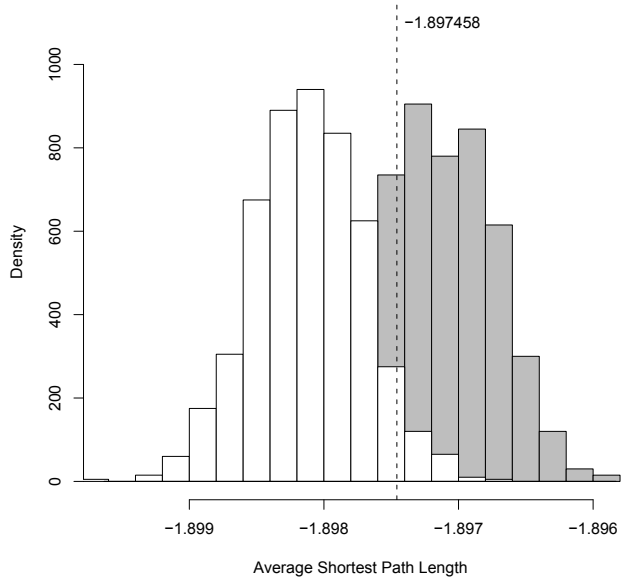

Figure 3.8: Hypothesis test for average path length when the null hypothesis is $H_0 : \kappa(1000, 0.1)$, and the alternative hypothesis is $H_A : \kappa(950, 50, 0.1, 0.5)$.

$\kappa(1000 - m, m, 0.1, q)$. We are testing the hypothesis $p = 0.1$ against the alternative hypothesis $q > 0.1$. The value of m ranged over $\{5, 10, 15, 20, \ldots, 100\}$. The value of q ranged over $\{0.10, 0.15, 0.20, \ldots, 0.90\}$. Each square in the plot is the power of the statistic generated from $R = 1000$ replicates.

We can see from the plots that with a large enough q and m, all of the statistics are able to identify the difference between the null and alternative hypotheses.

3.2.4 Power Difference plots

To analyze the power of our graph invariants, we now compute the difference of two graph invariants T_1, T_2 to get $\hat{\beta}(T_1) - \hat{\beta}(T_2)$. Most of the pairs of invariants have one that performs better in one set of parameters, but one invariant does not dominate the other in the entire parameter space. Figure 3.10 contains the power difference plots of five graph invariants. Out of the eight graph invariants, these five were not completely dominate in performance and have areas of interest the plots.

3.2.5 Most Powerful Statistic

For a fixed choices of n and p, we can compute the most powerful statistic in our hypothesis test for various q and m values. We have computed the most powerful statistic for $n = 100$ $p = 0.1$, $n = 100$ $p = 0.4$, and $n = 1000$ $p = 0.1$.

With $n = 100$ $p = 0.4$, scan statistic S_1 dominates as the most powerful statistic.

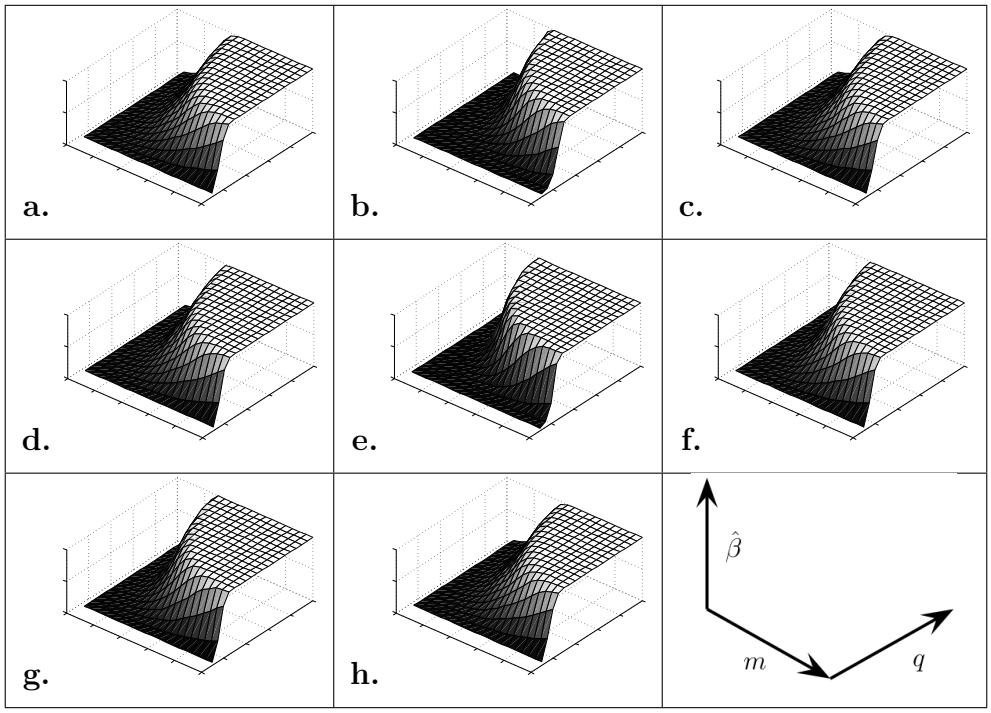

Figure 3.9: Power surface plots for the various graph invariants, as obtained from the Monte Carlo simulations for $n = 1000$, $p = 0.1$, $m \in \{5, 10, 15, \ldots, 100\}$, $q \in \{0.10, 0.15, 0.20, \ldots, 0.90\}$, $\alpha = 0.05$, and $R = 1000$. **a.** Number of edges, Size(G). **b.** Maximum degree, $\delta(G)$. **c.** Greedy maximum average degree approximation, $\text{MAD}_g(G)$. **d.** Eigenvalue maximum average degree approximation, $\text{MAD}_e(G)$. **e.** Scan statistic, $S_1(G)$. **f.** Number of Triangles, $\tau(G)$. **g.** Global clustering coefficient, CC(G). **h.** Average Path Length, APL(G). Powers range from approximately α for small m or q to approximately 1 for large m and q for all invariants. Substantial differences exist, but may not be apparent, between the various invariants for moderate m, q; these differences are readily apparent in the pairwise comparisons (Figure 3.10).

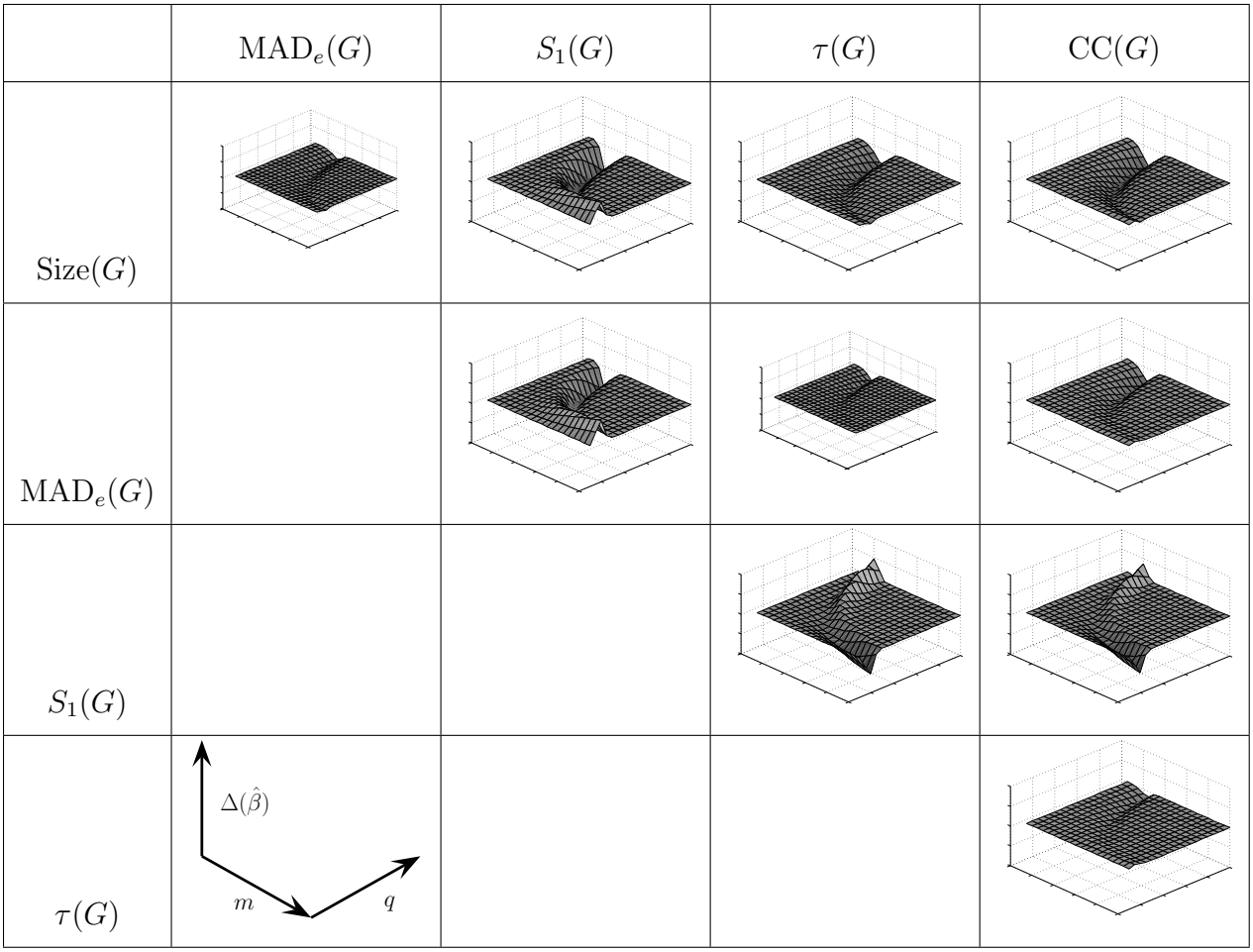

Figure 3.10: Comparative power surfaces for the various graph invariants, as obtained from Monte Carlo simulations for $n = 1000$, $p = 0.1$, $m \in \{5, 10, 15, \ldots, 100\}$, $q \in \{0.10, 0.15, 0.20, \ldots, 0.90\}$, $\alpha = 0.05$, and $R = 1000$. Each surface plot is representative of the power of the row invariant minus the power of the column invariant (e.g. the upper left corner depicts the power difference of size and exponential MAD_e $\hat{\beta}_{\text{Size}(G)} - \hat{\beta}_{\text{MAD}_e(G)}$) from Figure 3.9. Since powers are approximately α for small m or q and approximately 1 for large m and q for all invariants, power differences are approximately 0 in these regions. Substantial differences are readily apparent between the various invariants for moderate m, q in these comparative power surfaces.

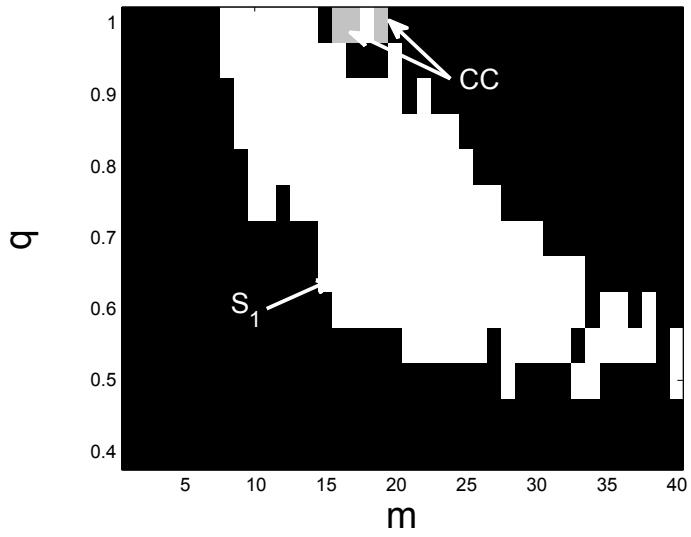

Figure 3.11: Most powerful statistic for $n = 100, p = 0.4$.

There are some values of large q and $m \approx 20$ where clustering coefficient has larger power. This required more simulates to be certain of the behavior of the clustering coefficient and scan statistic.

In the sparser graph $n = 100$ $p = 0.1$, there is no definitively most powerful graph invariant (see Figure 3.12). The scan statistic S_1 performs well for smaller m and $q > 0.5$. Meanwhile, the eigenvalue maximum average degree performs well for larger m and smaller q values. The number of triangles τ is most powerful sporadically in between the two.

Finally in $n = 1000$ $p = 0.1$ in Figure 3.13, scan statistic S_1 dominates most of the plot. Number of triangles τ and clustering coefficient CC outperform the scan statistic S_1 for smaller values of q and m.

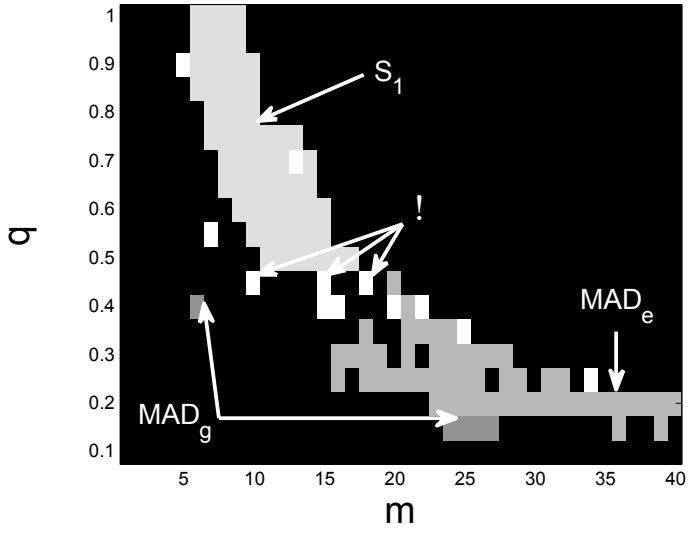

Figure 3.12: Most powerful statistic for $n = 100, p = 0.1$.

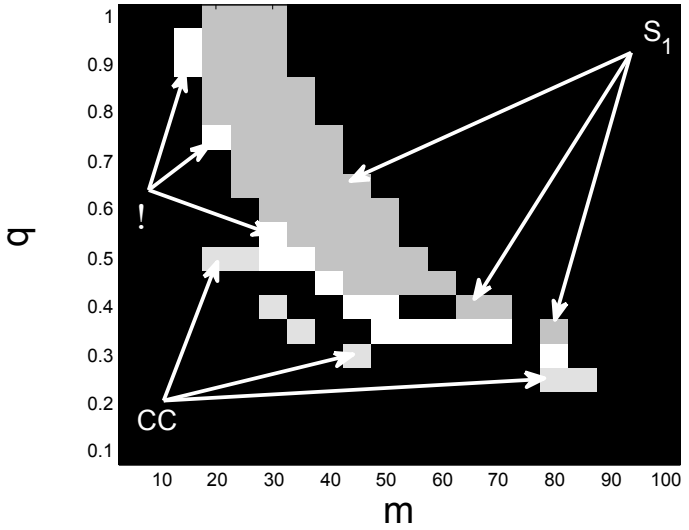

Figure 3.13: Most powerful statistic for $n = 1000, p = 0.1$.

3.3 Densest k-subgraph

A closely related problem is the densest k-subgraph problem, which is finding the subgraph of size k with the most edges

$$\arg\max_{V' \subset V, |V'|=k} \text{size}(G[V']).$$

If we find the densest "m-subgraph," we could study many interesting statistics. For example one could compute graph invariants of this subgraph. Unfortunately, the densest k-subgraph problem is known to be NP-hard by a reduction from the clique problem. Even in planer graphs, the densest k-subgraph problem has been shown by Keil and Brecht [75] to be NP-hard. It has been further shown by Khot [76] that there does not exist a polynomial-time approximation scheme (PTAS) for the densest k-subgraph problem by a reduction from the Minimum Distance of Code problem. However, there do exist approximation algorithms that can find a subgraph within a bounded ratio of the number of edges between the densest k-subgraph and the approximate solution, such as [77] for Feige, Peleg, Kortsarz.

Chapter 4

Vertex Nomination

Given a graph $G = (V, E)$ drawn from a stochastic block random graph $\kappa(b, \Lambda)$, where one block is of particular interest, vertex nomination is the task of creating a list of vertices such that vertices from the block of interest are in abundance at the top of the list. Vertex nomination is useful in situations where only a limited number of vertices can be examined, to discover block membership. We propose several vertex nomination schemes, include a canonical vertex nomination scheme, Metropolis-Hastings sampling vertex nomination scheme, spectral embedding vertex nomination scheme, and graph matching vertex nomination scheme. We derive theoretical results for performance, and compare the schemes on simulated and real data. Much of this section is submitted as [78].

Using the notation from Section 1.4 for $G = (V, E)$ distributed as a stochastic block random graph $\kappa(b, \Lambda)$, where $V = (v_1, v_2, \ldots, v_n)$ and there are K blocks

$V_1, V_2, \ldots, V_K$. For convenience and without the loss of generality, let the block of interest be the first block V_1. For simplicity, we assume knowledge of N and Λ, where $N = [n_k] = [|V_k|] \in \mathbb{N}^K$ is a vector denoting the cardinality of the blocks and Λ is the matrix of block connectivity probabilities.

Definition A **nomination list** $\Phi_G = (v_{\varphi(1)}, v_{\varphi(2)}, \ldots, v_{\varphi(m)})$ of vertices is a list of non-seed vertices. A **vertex nomination scheme** Φ is a procedure which produces a nomination list.

Vertex nomination is most applicable in situations where one has limited resources to process vertices. For example, if the graph is the communication network of a corporation with corrupt employees, one might have limited resources to investigate employees (see Section 4.8.2). Another example is a collection of political blog, where one has limited time to read blogs and one wants to read the most liberal blogs (see Section 4.8.2). For other recent work on vertex nomination, see [79, 80].

4.1 In Relation to Classification

The task of vertex nomination is very closely related to classification. In statistical classification, (X, Y) is an $\mathbb{R}^d \times \{1, 2, 3, \ldots K\}$-valued random pair, and a joint distribution of (X, Y), say $\mathcal{F}_{X,Y}$, determines the probability of observing the pairs, i.e. $(X_1, Y_1), (X_2, Y_2), \ldots (X_n, Y_n) \overset{iid}{\sim} \mathcal{F}_{X,Y}$ [81]. X is referred to as a feature vector and Y is referred to as a class label.

Definition The task of **classification** is to predict a class $Y \in \{1, 2, 3, \dots K\}$ for a new observation of features $X \in \mathbb{R}^d$, given a sequence of training data whose class membership are known, i.e. $(X_1, Y_1), (X_2, Y_2), \dots (X_n, Y_n) \overset{iid}{\sim} \mathcal{F}_{X,Y}$.

Definition The Bayes classifier predicts the class as the most likely class given the conditional distribution $\mathcal{F}_{Y|X}$. Let the density function of $\mathcal{F}_{Y|X}$ be $f_{Y|X}$, then the **Bayes classifier** for an observed data point x is $\arg\max_y f_{Y|X}(y|x)$.

For any distribution $\mathcal{F}_{X,Y}$, the Bayes classifier is the optimal classifier [81] and achieves the lowest possible misclassification error. Since the joint distribution is unknown in practice, this classifier is typically not available.

Related to classification, the task of ordering new data is referred to as information retrieval. Information retrieval is usually presented in the setting of retrieving resources from a library or database. Given a query, for example a keyword or keyphrase, the task of information retrieval returns a list of documents with a high concentration of relevant documents near the top of the list. The term "information retrieval" almost exclusively refers to the setting of natural language, where one wishes to retrieve documents, web pages, etc.

Let A be an adjacency matrix, let a_i be the i^{th} column in the adjacency matrices A, and let b_i be the block membership of vertex v_i. In our setting, the feature vector $X_i := a_i \in \{0, 1\}^n$, and the label $y_i := b_i \in \{1, 2, 3, \dots K\}$.

The tasks of classification and vertex nomination are closely related enough that given a nomination list Φ_G, one could classify the vertices. For example, one could

classify the first n_1 vertices in the nomination list Φ_G as being from V_1. Conversely, most classification methods have an innate ordering, which can be easily adapted to generate a nomination list.

4.2 Performance Metrics

In this section, we present and discuss the evaluation metric used to measure the performance of vertex nomination.

4.2.0.0.1 Mean Average Precision

Let Φ be a nomination scheme with $\Phi_G = \{v_{\varphi(1)}, v_{\varphi(2)}, \dots v_{\varphi(m)}\}$ as its nomination list for a graph G.

Definition The **precision at** j for a vertex nomination list Φ_G is defined as

$$P(\Phi_G, j) = \frac{\sum_{i=1}^{j} \mathbb{1}[v_{\varphi(i)} \in V_1]}{j}.$$

Precision can be thought of as the accuracy of nominating the first j vertices to be from block V_1.

Definition The **average precision** is defined as

$$\begin{aligned} AP'(\Phi_G) &= \frac{\sum_{j=1}^{n} \mathbb{1}[v_{\varphi(j)} \in V_1] P(j)}{|V_1|} = \frac{\sum_{j=1}^{n} \mathbb{1}[v_{\varphi(j)} \in V_1] \frac{\sum_{i=1}^{j} \mathbb{1}[v_{\varphi(i)} \in V_1]}{j}}{|V_1|} \\ &= \frac{\sum_{j=1}^{n} \sum_{i=1}^{j} \mathbb{1}[v_{\varphi(j)} \in V_1] \mathbb{1}[v_{\varphi(i)} \in V_1]}{j|V_1|} = \frac{\sum_{j=1}^{n} \sum_{i=1}^{j} \mathbb{1}[v_{\varphi(i)}, v_{\varphi(j)} \in V_1]}{j|V_1|}. \end{aligned}$$

Average precision is not the average of precision from 1 to n but, rather, it is the integral of precision over recall.

Another average precision we can consider is by literally taking the average of precision over the first n_1 positions in the nomination list which, in the best of worlds, would be the positions occupied by the vertices of V_1.

Definition The **literal average precision** is defined as

$$AP(\Phi_G) = \frac{1}{|V_1|}\sum_{j=1}^{n_1} P(j) = \frac{1}{|V_1|}\sum_{j=1}^{n_1}\frac{\sum_{i=1}^{j}\mathbb{1}[v_{\varphi(i)} \in V_1]}{j}.$$

Definition The **Mean Average Precision** of Φ_G as defined in the information retrieval community is

$$\begin{aligned} MAP'(\Phi) &= \mathbb{E}_\kappa[AP'(\Phi)] = \mathbb{E}_\kappa\left[\frac{1}{|V_1|}\sum_{j=1}^{n} P(j)\mathbb{1}[v_{\varphi(j)} \in V_1]]\right] \\ &= \mathbb{E}_\kappa\left[\sum_{j=1}^{n}\sum_{i=1}^{j}\frac{\mathbb{1}[v_{\varphi(i)}, v_{\varphi(j)} \in V_1]}{j|V_1|}\right], \end{aligned}$$

where κ is the distribution of the random graph model. This particular mean average precision is most commonly used.

Definition Literal Mean Average Precision is

$$\begin{aligned} MAP(\Phi) &= \mathbb{E}_\kappa\left[\frac{1}{|V_1|}\sum_{j=1}^{n_1} P(j)\right] = \mathbb{E}_\kappa\left[\frac{1}{n_1}\sum_{j=1}^{n_1}\frac{|\{i \in [j] : \Phi_G(i) \in V_1\}|}{j}\right] \\ &= \mathbb{E}_\kappa\left[\sum_{i=1}^{j}\frac{\mathbb{1}[v_{\varphi(i)} \in V_1]}{j}\right]. \end{aligned}$$

Notice that literal mean average precision is linear in the indicator function. Meanwhile, mean average precision has interaction terms, $\mathbb{1}[v_{\varphi(i)}, v_{\varphi(j)} \in V_1]$. This makes

the literal mean average precision more analytically approachable than the information retrieval MAP'. Henceforth, we shall use MAP and refer to it as mean average precision.

4.3 Canonical

4.3.1 Scheme

Given an observed graph G drawn from a stochastic block model $\kappa(N, \Lambda)$, the canonical vertex nomination scheme orders the vertices by conditional probability of their being in the block of interest V_1. Recall that $N = [n_k] \in \mathbb{N}^K$ are the sizes of the blocks and $\Lambda \in [0,1]^{K \times K}$ is the matrix of edge probabilities. In this model, the block membership function $b : V \mapsto \{1, 2, \ldots, K\}$ is uniformly distributed over $\mathcal{B}$, where $\mathcal{B}$ is the set of all block membership functions $b : V \mapsto \{1, 2, \ldots, K\}$ such that for each $k \in \{1, 2, \ldots, K\}$, $n_k = |b^{-1}(k)|$. For any $v_i \in V$, the conditional probability of a vertex being interesting, upon observing the graph, is

$$\begin{aligned} \mathbb{P}[v_i \in V_1 | G] &= \sum_{b \in \mathcal{B}} \mathbb{P}[v_i \in V_1 | G, b] \mathbb{P}[b|G] = \sum_{b \in \mathcal{B}} \mathbb{1}[v_i \in V_1 | b] \mathbb{P}[b|G] \\ &= \sum_{b \in \mathcal{B}} \frac{\mathbb{1}[v_i \in V_1 | b] \mathbb{P}[G|b] \mathbb{P}[b]}{\sum_{b \in \mathcal{B}} \mathbb{P}[G|b] \mathbb{P}[b]} = \sum_{b \in \mathcal{B}} \frac{\mathbb{1}[v_i \in V_1 | b] \mathbb{P}[G|b]}{\sum_{b \in \mathcal{B}} \mathbb{P}[G|b]}. \end{aligned}$$

The probability of realizing the graph G given the memberships $\mathbb{P}[G|b]$ is computed as

$$\mathbb{P}[G|b] = \prod_{k=1}^{K} \prod_{l=k}^{K} \lambda_{k,l}^{e(V_k, V_l)} (1 - \lambda_{k,l})^{c(V_k, V_l) - e(V_k, V_l)}, \tag{4.1}$$

where $e(V_k, V_l)$ is the number of edges in G with one endpoint in V_k and the other endpoint in V_l and

$$c(V_k, V_l) = \begin{cases} \binom{m_k+n_k}{2}, & \text{if } k = l \\ (m_k + n_k)(m_l + n_l), & \text{otherwise} \end{cases}$$

the number of possible edges in E between V_k and V_l.

Definition The **canonical vertex nomination scheme** for random graph distributed $\kappa(N, \Lambda)$ orders the vertices by the probability $\mathbb{P}[v_i \in V_1|G]$ and is denoted

$$\Phi^*_G = (v_{\varphi^*(1)}, v_{\varphi^*(2)}, \ldots, v_{\varphi^*(n)}).$$

When there are seed vertices, we further restrict the block membership functions in $\mathcal{B}$ so that for all $b \in \mathcal{B}$ and for all seed vertices v that $b(v)$ is the block associated with v.

Now, we present the canonical vertex nomination scheme for random graphs distributed as $\kappa(b, \Lambda)$. This method is an alternative model to $\kappa(N, \Lambda)$. Here, the block membership function b is not a random variable. Therefore, the block membership of vertex $v_i \in V$, $b(v_i)$ is not a random variable. In other words, either a vertex is a member of the block of interest, $b(v_i) = 1$, or the vertex is not, $b(v_i) \neq 1$. There is no distribution or "probability" for $b(v_i) = 1$. Let us carefully define a conditional probability to consider.

Consider the sample space of graphs in the stochastic block model $\kappa(b, \Lambda)$. Furthermore, we assume that S, M, N, and Λ are known, but b is not known. For simplicity of explanation, let us start with no seed vertices, i.e., $S = \varnothing, M = N$.

Consider the bijection $\rho : V \mapsto V$ of the vertices and consider the block membership function $b(\rho(\cdot))$. This produces an graph which has the same stochastic block random graph distribution but a different block membership function. Thus any relabeling of the graph produces an equivalent stochastic block model.

For a given graph G on the vertex set V, consider the set of all graphs isomorphic to G, denoted as $\langle G \rangle$. For any G with no symmetry, i.e., G only has a trivial automorphism group, for any $H \in \langle G \rangle$, there exists a unique permutation $\xi_{G,H} : \{1, 2, \ldots, n\} \mapsto \{1, 2, \ldots, n\}$, which maps the vertex indices of graph G to the vertex indices of graph H.

Recall, there are $2^{\binom{n}{2}}$ labeled graphs in the sample space. We condition on all sample points isomorphic to G, $\langle G \rangle$. When G only has the trivial automorphism group, there are $n!$ such graphs. For any $v_i \in V$, the event we consider in the conditional sample space $\langle G \rangle$ is all sample points such that $b(v_{\xi_{G,H}(i)}) = 1$. That is, all sample points $H \in \langle G \rangle$ such that vertex aligned to v_i in G, $v_{\xi_{G,H}(i)}$, is in block V_1. The conditional probability is written as

$$\mathbb{P}[\{H \in \langle G \rangle : b(v_{\xi_{G,H}(i)}) = 1\} | \langle G \rangle]. \tag{4.2}$$

Definition The **canonical vertex nomination scheme** orders the vertices by the conditional probability $\mathbb{P}[\{H \in \langle G \rangle : b(v_{\xi_{G,H}(i)}) = 1\} | \langle G \rangle]$ and is denoted

$$\Phi^*_G = (v_{\varphi^*(1)}, v_{\varphi^*(2)}, \ldots, v_{\varphi^*(n)}).$$

Thus by definition, for all $i \in \{1, 2, 3, \dots n-1\}$,

$$\mathbb{P}[\{H \in \langle G \rangle : b(v_{\xi_{G,H}(\varphi^*(i))}) = 1\} | \langle G \rangle] \geq \mathbb{P}[\{H \in \langle G \rangle : b(v_{\xi_{G,H}(\varphi^*(i+1))}) = 1\} | \langle G \rangle].$$

Later in Theorem 4.3.2, we show that ranking by these probabilities yields the highest, by MAP metric, among all vertex nomination schemes. Thus, the canonical vertex nomination scheme is the best possible scheme.

When the parameters N, Λ of the graph are known, and there is no symmetry in the observed graph G (G only has the trivial automorphism group), current methods to compute $\mathbb{P}[\{H \in \langle G \rangle : b(v_{\xi_{G,H}(i)}) = 1\} | \langle G \rangle]$ require exponential runtime.

Let Ψ_n be the set of all permutations on the set $\{1, 2, \dots n\}$. The probability space of $\mathbb{P}[\{H \in \langle G \rangle : b(v_{\xi_{G,H}(i)}) = 1\} | \langle G \rangle]$ can be represented as all possible permutations $\xi_{G,H} \in \Psi_n$,

$$\mathbb{P}[\{H \in \langle G \rangle : b(v_{\xi_{G,H}(i)}) = 1\} | \langle G \rangle] = \frac{\sum_{H \in \langle G \rangle} \mathbb{1}[b(v_{\xi_{G,H}(i)}) = 1] \mathbb{P}[H]}{\sum_{H \in \langle G \rangle} \mathbb{P}[H]}.$$

The probability of a graph G given parameters b and Λ is

$$\mathbb{P}[G] = \prod_{k=1}^{K} \prod_{l=k}^{K} \lambda_{k,l}^{e(V_k, V_l)} (1 - \lambda_{k,l})^{c(V_k, V_l) - e(V_k, V_l)}, \tag{4.3}$$

where $e(V_k, V_l)$ is the number of edges in G with one endpoint in V_k and the other endpoint in V_l and

$$c(V_k, V_l) = \begin{cases} \binom{m_k + n_k}{2}, & \text{if } k = l \\ (m_k + n_k)(m_l + n_l), & \text{otherwise} \end{cases}$$

the number of possible edges not in E between V_k and V_l.

Notice that even though we do not know b, we can still compute this probability. We can simply consider a surrogate b' such that for all $k \in \{1, 2, \dots K\}$ the number of vertices which map to k in b, $|b^{-1}(k)|$ is the same as the number of vertices which map to k in b', $|b'^{-1}(k)|$. This is because any relabeling of the vertices does not affect the probability of the graph.

In the case where there are seed vertices S, part of the block membership function is observed. Let the seed vertices be $v_1, v_2, \dots, v_s$. Then, instead of all possible partitions $\xi_{G,H} \in \Psi_n$, consider permutations $\xi_{G,H} \in \Psi_n$ such that for all $i \in \{1, 2, \dots, s\}$, $\xi_{G,H}(i) = i$. This reduces the set of permutations from $n!$ to $m!$.

In the case where the automorphism group is not trivial, the conditional probability is no longer well defined. For graphs in the equivalence class $\langle G \rangle$, there exist valid multiple permutations $\xi_{G,H}$. If we happen to observe a graph with symmetry, then we treat it as if it has no symmetry. That is, we compute the probability as Equation **??**. Although this quantity is incorrect, the number of graphs which have no symmetry quickly drops to zero as $m + n$ goes to infinity [82, 83].

Note that

$$\mathbb{P}[v_i \in V_1 | G]$$

in the model $\kappa(N, \Lambda)$ is computationally equal to

$$\mathbb{P}[\{H \in \langle G \rangle : b(v_{\xi_{G,H}(i)}) = 1\} | \langle G \rangle]$$

in the model $\kappa(b, \Lambda)$. These models are very closely related. Recall that the difference between the $\kappa(N, \Lambda)$ and $\kappa(b, \Lambda)$ is that in the second model, we have conditioned an

a specific b while in the first model b is uniformly distributed over $\mathcal{B}$.

4.3.2 Theoretical Results

In this section, we prove that the canonical nomination scheme has a MAP greater than or equal to the MAP of any other nomination scheme. Thus, the canonical nomination scheme is the optimal vertex nomination scheme.

Lemma 4.3.1

Suppose that $\alpha_1, \alpha_2, \ldots \alpha_n$ *and* $\beta_1, \beta_2, \ldots, \beta_n$ *are non-increasing, nonnegative sequences of real numbers. Let* $\varphi \in \Psi_n$ *be a permutation, then* $\sum_{i=1}^{n} \alpha_i \beta_i \geq \sum_{i=1}^{n} \alpha_i \beta_{\varphi(i)}$.

Proof First consider the case where $\beta_1, \beta_2, \ldots, \beta_n = 1, 1, \ldots, 1, 0, \ldots 0$, where there are m ones. In this case, it is clear that the lemma is true. Consider any permutation $\varphi \in \Psi_n$ and φ^{-1} the inverse of the permutation $\varphi(\varphi^{-1}(i)) = \varphi^{-1}(\varphi(i)) = i$, without loss of generality, let $\alpha_{\varphi_1^{-1}} \geq \alpha_{\varphi_2^{-1}} \geq \ldots \geq \alpha_{\varphi_m^{-1}}$. Then $\varphi_i^{-1} \leq i$ and

$$\sum_{i=1}^{n} \alpha_i \beta_{\varphi_i} = \sum_{i=1}^{n} \alpha_{\varphi_i^{-1}} \beta_i \leq \sum_{i=1}^{n} \alpha_i \beta_i.$$

Now consider the general case. Let $\delta_i = \beta_i - \beta_{i-1}$ for $i = 2, 3, \ldots, n$, and $\delta_1 = \beta_1$. Notice if we use the sequence $\delta_i, \delta_i, \ldots, \delta_i, 0, \ldots, 0$ where there are i δ_i followed by $n-i$ zeros as the sequence of β's, then we are in the first case (with a multiplicative factor) and the inequality holds. Denote this sequence $\delta_i, \delta_i, \ldots, \delta_i, 0, \ldots, 0$ as $\gamma_{i1}, \gamma_{i2}, \ldots, \gamma_{in}$. Then, we have $\sum_{j=1}^{n} \gamma_{ij} = \sum_{j=1}^{i} \delta_i = \beta_i$. Repeating this n times, once for each δ_i,

yields

$$\begin{aligned}\sum_{i=1}^{n}\alpha_i\beta_{\varphi_i} &= \sum_{i=1}^{n}\alpha_{\varphi_i^{-1}}\beta_i = \sum_{i=1}^{n}\alpha_{\varphi_i^{-1}}\sum_{j=1}^{i}\delta_j = \sum_{i=1}^{n}\alpha_{\varphi_i^{-1}}\sum_{j=1}^{n}\gamma_{ij} = \sum_{j=1}^{n}\sum_{i=1}^{n}\alpha_{\varphi_i^{-1}}\gamma_{ij} \\ &\leq \sum_{j=1}^{n}\sum_{i=1}^{n}\alpha_i\gamma_{ij} = \sum_{i=1}^{n}\alpha_i\sum_{j=1}^{n}\gamma_{ij} = \sum_{i=1}^{n}\alpha_i\beta_i.\end{aligned}$$

∎

Now, we are ready to prove that vertex nomination scheme has an optimal MAP.

Theorem 4.3.2

For every vertex nomination scheme Φ_G, the mean average precision of Φ_G^* is greater than or equal to the mean average precision of Φ_G.

Proof Let $G \sim \kappa(M, N, \Lambda)$ be a random graph. Let $\alpha_i = \frac{1}{n_1}\sum_{j=i}^{n_1}\frac{1}{j}$ for $i \leq n_1$, and $\alpha_i = 0$ otherwise. This α_i sequence is nonnegative and non-increasing, thus we can apply Lemma 4.3.1. Recall that Φ_G^* orders vertices by non-increasing conditional

probability given the observed graph, so

$$\begin{aligned} E\left[\sum_{i=1}^{n}\alpha_i \mathbb{1}[\Phi_G(i)\in V_1]\right] &= \sum_{i=1}^{n}\alpha_i \mathbb{P}[\Phi_G(i)\in V_1] \\ &= \sum_{i=1}^{n}\alpha_i\left(\sum_{G}\mathbb{P}[G]\ P[\Phi_G(i)\in V_1|G]\right) \\ &= \sum_{G}\mathbb{P}[G]\left(\sum_{i=1}^{n}\alpha_i\ \mathbb{P}[\Phi_G(i)\in V_1|G]\right) \\ &\leq \sum_{G}\mathbb{P}[G]\left(\sum_{i=1}^{n}\alpha_i\ \mathbb{P}[\Phi^*_G(i)\in V_1|G]\right) \\ &= \sum_{i=1}^{n}\alpha_i\ \mathbb{P}[\Phi^*_G(i)\in V_1] \\ &= E\left[\sum_{i=1}^{n}\alpha_i \mathbb{1}[\Phi^*_G(i)\in V_1]\right]. \end{aligned}$$

The literal mean average precision defined in Section 4.2 is this expression. ∎

Theorem 4.3.2 shows that the canonical vertex nomination scheme is optimal for any set of nonnegative, and non-increasing sequence of weights.

4.4 Metropolis-Hastings Sampling

4.4.1 Scheme

The canonical nomination scheme is not feasible for graphs larger than about twenty vertices. Therefore, it is not practical for most graphs. This section describes methods to approximate the conditional probability, allowing us to estimate the canonical vertex nomination scheme for much larger graphs. For simplicity we de-

scribe sampling from $\mathcal{B}$ in the model $\kappa(N, \Lambda)$. This is similar $b(v_{\xi_{G,H}(\cdot)})$ when sampling H from the sample space of $\langle G \rangle$ in the model $\kappa(b, \Lambda)$.

An intelligent way to sample is to use Metropolis-Hastings algorithm. First used by Nicholas Metropolis et al. [84] to estimate states in plasma physics simulation, the Metropolis-Hastings algorithm is a method for generating random samples from a probability distribution that is difficult to directly sample from. The samples are generated from a Markov chain Monte Carlo (MCMC), and only the ratio probabilities of samples needs to be known. The Metropolis-Hastings algorithm is often used for approximating a distribution or an expectation.

Algorithm 4.6 Metropolis-Hastings Sampling

Draw a x_{-B} uniformly at random from $\mathcal{X}$ ▷ Initialization

for $t = -B + 1$ to T **do**

Draw a x' from the distribution $Q(\cdot|x_t)$ ▷ Generate Candidate

$\alpha \leftarrow \frac{P(x')Q(x_t|x')}{P(x_t)Q(x'|x_t)}$ ▷ Compute acceptance ratio

if $\alpha \geq 1$ or with probability α **then**

$x_{t+1} \leftarrow x'$ ▷ Accept new candidate

else

$x_{t+1} \leftarrow x_t$ ▷ Reject new candidate

end if

end for

The Metropolis-Hastings Algorithm gives a practical way to estimate

$$\mathbb{E}\left[\mathbb{1}[v_i \in V|G]\right],$$

by generating an MCMC of samples from $\mathcal{B}$ distributed as $\mathbb{P}[b|G]$.

The first step of the Metropolis-Hastings algorithm is to generate a initial starting point x_0. This can be a uniformly random point from the sample space $\mathcal{X}$. At the t^{th} step, we randomly pick a new candidate x' from a distribution $Q(x'|x_t)$. If this distribution $Q(x|y)$ is symmetric, e.g., $Q(x|y) = Q(y|x)$, we accept the new state x' if $\alpha \geq 1$ or with probability α, where

$$\alpha = \frac{\mathbb{P}[x']}{\mathbb{P}[x_t]}.$$

Then, we repeat sampling until there are $B+T$ samples, $x_{-B+1}, x_{-B+2}, x_{-B+3}, \ldots, x_T$. The first B samples are burn-in, and are discarded, to ensure the underlying Markov Chain are sufficiently close to stationarity.

4.4.1.0.2 Neighboring Block Membership Function

We propose a transition distribution $Q(b'|b^t)$, such that it depends on the previous block membership function and is symmetric. We consider a subset of block membership functions to transition to and call these neighboring block membership functions.

Definition Let $b \in \mathcal{B}$ be a block membership function and V_k^b be the k^{th} block determined by b. For any $i, j \in \{1, 2, \ldots, n\}$, $i \neq j$, consider the following function

b' such that $v_j \in V_i^{b'}$ and $v_i \in V_j^{b'}$, otherwise for all $l \neq i, j$, $b(v_l) = b'(v_l)$. Thus, the block membership function b' differs from b for exactly two vertices. We define a b' as **neighboring partition** of b, and the act of exchange two vertices in a partition is called a **swap**. We denote the swap of two vertices of partition b as $b(v_i, v_j)$.

Let b^t be the block membership function of the t^{th} iteration in this Metropolis-Hastings algorithm. We select the next candidate b' by first selecting two vertices and swapping them. The first vertex v_i is selected uniformly at random from V_1 and the second vertex v_j is selected uniformly at random from $V \backslash V_1$, i.e., $b' := b^{t-1}(v_i, v_j)$. Then we compute the acceptance rate

$$\alpha := \frac{\mathbb{P}[b'|G]}{\mathbb{P}[b^{t-1}|G]} = \frac{\frac{\mathbb{P}[G|b']\mathbb{P}[b']}{\mathbb{P}[G]}}{\frac{\mathbb{P}[G|b^{t-1}]\mathbb{P}[b^{t-1}]}{\mathbb{P}[G]}} = \frac{\mathbb{P}[G|b']}{\mathbb{P}[G|b^{t-1}]},$$

to decide whether or not to accept b' or keep b^t as prescribed by the algorithm. Recall $\mathbb{P}[G|b]$ is computed via Equation 4.1. Then we repeat this procedure until some convergence criterion is met or a predetermined number of iterations is reached.

Calculating the acceptance rate α using Equation 4.3 is not particularly efficient. A more clever way to compute $\mathbb{P}[G|b']$ is to use the previously computed $\mathbb{P}[G|b^{t-1}]$. The values of $\mathbb{P}[G|b']$ and $\mathbb{P}[G|b^{t-1}]$ only differ by two columns of the expanded Λ matrix, $\overline{A}$. Using the log likelihood ratio, we significantly increase the efficiency.

Consider the log likelihood ratio of G for two block membership functions b and for $b' = b(v_i, v_j)$

$$\log\left(\frac{\mathbb{P}[G|b']}{\mathbb{P}[G|b]}\right),$$

Algorithm 4.7 Canonical Metropolis-Hastings Sampling

Uniformly at random select partition $b^{-B} \in \mathcal{B}$ ▷ Initialization

for $t = -B + 1$ to T **do**

Select a vertex $v_i \in V_1$ and $v_j \in V \backslash V_1$ ▷ Generate Candidate

$b' \leftarrow b^{t-1}(v_i, v_j)$

Compute $\alpha = \frac{\mathbb{P}[G|b']}{\mathbb{P}[G|b^{t-1}]}$

if $\alpha \geq 1$ or with probability α **then**

$b^t \leftarrow b'$ ▷ Accept new candidate

else

$b^t \leftarrow b^{t-1}$ ▷ Reject new candidate

end if

end for

where $v_i \in V_1^b$ and $v_j \notin V_1^b$. Let the adjacency matrix A' be the adjacency matrix after swapping vertices v_i and v_j. This results in an adjacency matrix A with rows i and j swapped, and columns i and j swapped. Then we can rewrite the log likelihood ratio as

$$\begin{aligned}\log\left(\frac{\mathbb{P}[G|b']}{\mathbb{P}[G|b]}\right) &= \sum_{\{l,m\}\in\binom{[n]}{2}} a'_{lm}\log\left(\frac{\lambda_{b'_l,b'_m}}{1-\lambda_{b'_l,b'_m}}\right) - \sum_{\{l,m\}\in\binom{[n]}{2}} a_{lm}\log\left(\frac{\lambda_{b_l,b_m}}{1-\lambda_{b_l,b_m}}\right) \\ &= \sum_{\{l,m\}\in\binom{[n]}{2}} \left(a'_{lm}\log\left(\frac{\lambda_{b'_l,b'_m}}{1-\lambda_{b'_l,b'_m}}\right) - a_{lm}\log\left(\frac{\lambda_{b_l,b_m}}{1-\lambda_{b_l,b_m}}\right)\right).\end{aligned}$$

Notice that for most indices $l, m \in \binom{\{1,2,\ldots,n\}}{2}$, the terms are identical. The indices are not identical for terms when $l \neq i, j$ or $m \neq i, j$, i.e., row and columns i and j. Since the graph is undirected, we only need to consider columns i and j. This simplifies to

$$\begin{aligned}\log\left(\frac{\mathbb{P}[G|b']}{\mathbb{P}[G|b]}\right) = \sum_{l\neq i,j}\Bigg[&a'_{li}\log\left(\frac{\lambda_{b'_l,b'_i}}{1-\lambda_{b'_l,b'_i}}\right) + a'_{lj}\log\left(\frac{\lambda_{b'_l,b'_j}}{1-\lambda_{b'_l,b'_j}}\right) \\ &- a_{li}\log\left(\frac{\lambda_{b_l,b_i}}{1-\lambda_{b_l,b_i}}\right) - a_{lj}\log\left(\frac{\lambda_{b_l,b_j}}{1-\lambda_{b_l,b_j}}\right)\Bigg].\end{aligned}$$

Recall that the partition of vertices v_i and v_j are swapped in A'. Thus $b_i = b'_j$, $b_j = b'_i$, and for all $l \neq i, j$, $b_l = b'_l$, Allowing further simplification, we have

$$\begin{aligned}
\log\left(\frac{\mathbb{P}[G|b']}{\mathbb{P}[G|b]}\right) = & \sum_{l\neq i,j}\left[a_{li}\log\left(\frac{\lambda_{b_l,b_j}}{1-\lambda_{b_l,b_j}}\right) + a_{lj}\log\left(\frac{\lambda_{b_l,b_i}}{1-\lambda_{b_l,b_i}}\right)\right. \\
& \left. - a_{li}\log\left(\frac{\lambda_{b_l,b_i}}{1-\lambda_{b_l,b_i}}\right) - a_{lj}\log\left(\frac{\lambda_{b_l,b_j}}{1-\lambda_{b_l,b_j}}\right)\right] \\
= & \sum_{l\neq i,j}\left[a_{li}\left(\log\left(\frac{\lambda_{b_l,b_j}}{1-\lambda_{b_l,b_j}}\right) - \log\left(\frac{\lambda_{b_l,b_i}}{1-\lambda_{b_l,b_i}}\right)\right)\right. \\
& \left. + a_{lj}\left(\log\left(\frac{\lambda_{b_l,b_i}}{1-\lambda_{b_l,b_i}}\right) - \log\left(\frac{\lambda_{b_l,b_j}}{1-\lambda_{b_l,b_j}}\right)\right)\right] \\
= & \sum_{l\neq i,j}(a_{li}-a_{lj})\left(\log\left(\frac{\lambda_{b_l,b_j}}{1-\lambda_{b_l,b_j}}\right) - \log\left(\frac{\lambda_{b_l,b_i}}{1-\lambda_{b_l,b_i}}\right)\right) \\
= & \sum_{l\neq i,j}(a_{li}-a_{lj})\log\left(\frac{\lambda_{b_l,b_j}(1-\lambda_{b_l,b_i})}{\lambda_{b_l,b_i}(1-\lambda_{b_l,b_j})}\right). \qquad (4.4)
\end{aligned}$$

Computing this factor only requires examining two columns, which reduces the computation time from $O(n^2)$ to $O(n)$. Thus canonical Metropolis-Hastings sampling is much more practical than directly computing $\mathbb{P}[G|b]$. Since the algorithm has no fixed number of samples, we are able to scale and apply the algorithm to larger graphs.

Definition Let the partitions obtained from canonical Metropolis-Hastings sampling be $(b^{-B+1}, b^{-B+2}, \ldots, b^1, b^2, \ldots, b^T)$. The **Metropolis-Hastings vertex nomination scheme**, denoted by Φ_G^{MH}, orders the vertices by estimating the conditional probability $\mathbb{P}[G|b]$ as

$$\widehat{\mathbb{P}}[G|b] := \frac{\sum_{t=1}^{T}\mathbb{1}[v_i \in V_1^{b^t}]}{T}.$$

4.4.2 Number of Samples

In this section, we explore the number of samples needed in the Metropolis-Hastings vertex nomination scheme Φ^{MH}.

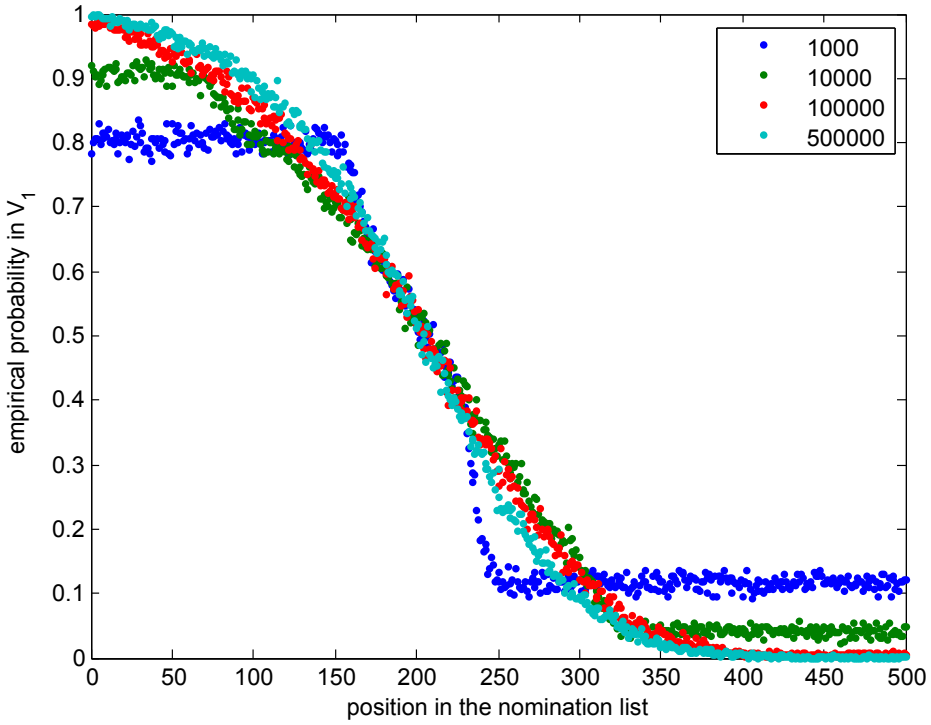

Figure 4.1: Φ^{MH} using 1000, 10000, 100000, 50000 samples. The simulated graphs have 20 seeds and 500 non-seed vertices. This plot demonstrates the increase in performance with more samples.

Figure 4.1 demonstrates the increase in performance with respect to the number of samples used to estimate $\mathbb{P}[G|b]$. This is done on the mid-sized simulation data (see Section 4.8.1 for simulation parameters). The plot shows the vertex nomination accuracy of Metropolis-Hastings vertex nomination using 1000, 10000, 100000, 500000 samples (all with 25000 burn-in). In Section 4.8, we use 500000 samples and 25000

Samples	1000	10000	100000	500000
MAP	0.80	0.875	0.923	0.945
Runtime	9.5	12.8	45.6	191

Table 4.1: MAP and runtime of Φ^{MH} for different numbers of samples. The runtime in this table is in seconds.

burn-in for Φ^{MH}.

4.4.3 Theoretical Results

Theorem 4.4.1 *The Metropolis-Hastings vertex nomination scheme with sufficient number of samples converges to the true distribution,*

$$\widehat{\mathbb{P}}[v_i \in V_1|G] \to \mathbb{P}[v_i \in V_1|G]$$

Thus Metropolis-Hastings vertex nomination scheme converges to the canonical vertex nomination scheme, and the $MAP(\Phi^{MH})$ converges to $MAP(\Phi^*)$.

The Markov process $\{b_t\}_{t=1,2,\ldots}$ is ergodic with stationary distribution $\mathbb{P}[v_i \in V_1|G]$. With enough samples the canonical Metropolis-Hastings sampling algorithm converges to the target distribution [85–89].

Proof

4.5 Spectral Partitioning

4.5.1 Scheme

Another vertex nomination scheme we develop is based on the spectral partitioning algorithm from Section 2.3. Applying spectral partitioning on all $s+m$ vertices yields the embedding $\widehat{X}$, and K clusters from minimizing $\|\widehat{X} - C\|_F$, where C has k distinct rows. After performing the spectral partitioning, we determine which cluster is most likely to be in block V_1. We determine which block is most likely V_1 by using the seed vertices from block V_1, denoted by U_1. Let C_k be the vertices in cluster k. We call the cluster with the most U_1 vertices $\widehat{V}_1$, let the centroid of this cluster be $\widehat{C}_1$. We rank vertices in V by their Euclidean distance from $\widehat{C}_1$. Let Φ^S denote the spectral partitioning vertex nomination scheme, and the order of the vertices as $\Phi^S_G = (v_{\varphi^S(1)}, v_{\varphi^S(2)}, \dots v_{\varphi^S(n)})$.

Algorithm 4.8 Spectral Partitioning Nomination

$ODO^T \leftarrow A$ ▷ Compute spectral decomposition

$\widehat{U} \leftarrow [U_1, U_2, \dots U_d], \widehat{D} \leftarrow diag(D_1, D_2, \dots D_d)$ ▷ Low rank approximation

$\widehat{X} \leftarrow \widehat{O}\widehat{D}^{\frac{1}{2}}$ ▷ Embed Vertices

$C \leftarrow \min_C \|\widehat{X} - C\|_F$ ▷ Cluster Embeddings

$c'_1, c'_2, \dots, c'_K \leftarrow$ distinct rows of C ▷ Estimate Block Membership Function

$\widehat{V}_1 \leftarrow \arg\max_{C_k : k \in \{1,2,\dots K\}} |\{u \in U_1 : u \in C_k\}|$ ▷ Estimate the V_1 from seeds.

To approximately solve the clustering $\min \|X - C\|_F$, we use the `Mclust` algo-

rithm. This is not equivalent to solving $\min \|X - C\|_F$. The approximate **spectral partitioning (SP) vertex nomination scheme** using `Mclust` is denoted $\Phi_G^{S^\dagger}$

4.5.2 Theoretical Results

In this section, we show that the mean average precision MAP of the spectral partitioning nomination scheme Φ_G^S converges to 1 as $n \to \infty$.

Theorem 4.5.1

Under the same conditions as Theorem 2.3.3, the average precision of the spectral partitioning nomination scheme Φ_G^S converges to 1 as $n \to \infty$. Thus the mean average precision of Φ_G^S also converges to 1 as $n \to \infty$.

Proof Lyzinski et al. [47] showed that with high probability there are no incorrectly clustered vertices as $n \to \infty$ under mild assumptions (Theorem 2.3.3). This implies that the mean average precision of Φ^S converges to 1 as $n \to \infty$. ∎

4.6 Seeded Graph Matching

Recall, the seeded graph matching algorithm from Section 2.4.2 takes two graphs as input and returns a permutation P as output. In the vertex nomination framework, we only have one graph, but we are also given N and Λ. Since the true expanded Λ matrix $\overline{A}$ is unknown, we construct a permutation of the true $\overline{A}$ say $\widetilde{A}$. For convenience, in $\widetilde{A}$ we group all vertices of the same block together. Let $\widetilde{b}$ be the associated

block membership function for $\widetilde{A}$. Thus $\widetilde{b} = [1, 1, \ldots, 1, 2, 2, \ldots, 2, 3 \ldots, K]$, where there are n_1 ones, n_2 twos, etc., so $\widetilde{A} := [\lambda_{\widetilde{b}_i, \widetilde{b}_j}] \in \mathbb{R}^{n \times n}$.

For the graph-matching vertex nomination scheme, we first perform seeded graph matching between A and $\widetilde{A}$. The graph matching objective function is $\arg\min_P \|A - P\widetilde{A}P^T\|_F$.

4.6.1 Residual

To evaluate the effectiveness of matching vertices, we compute a residual quantity and use it to rank the vertices. Explicitly, the residual quantity for vertex v_i is

$$r_i = \|[A - P\widetilde{A}P^T]_i\|_2^2 = \|[r_{i,1}, r_{i,2}, \ldots, r_{i,n+m}]\|_2^2 = \sum_{v_j \in V} (\mathbb{1}[v_i \sim v_j] - \Lambda_{\widetilde{b}(\psi(v_i)), \widetilde{b}(\psi(v_j))})^2.$$

Define $R = A - P\widetilde{A}P^T = [r_{ij}]$, then the residual for vertex v_i is $\|[r_{i,1}, r_{i,2}, \ldots, r_{i,n+m}]\|_2$, i.e. the ℓ_2-norm of the i^{th} row of the objective. Let $\psi : V^A \mapsto V^B$ denote the solution to the seeded graph matching problem. Notice that r_i is simply the decomposition of the graph matching objective,

$$\sum_{i=1}^{n} r_i = \|A - P\widetilde{A}P^T\|_F^2.$$

The **graph matching (GM) nomination scheme**, is defined as

$$\Phi_G^M := (v_{\varphi^M(1)}, v_{\varphi^M(2)}, \ldots, v_{\varphi^M(n)}).$$

such that first n_1 vertices in V_1 are ranked by the increasing order of the residual, and the remaining vertices not in V_1 are ranked by the decreasing residual.Notice that

the nomination depends on the predicted block membership of the vertex. Thus the algorithm treats vertices matched to different blocks differently.

Below we present a limiting result for graph matching vertex nomination scheme. This result requires no seeds, i.e. $s_1 = s_2 = \ldots = s_K = 0$. We consider with K and Λ fixed with graphs drawn from $\kappa(N, M, \Lambda)$. The values $n_1, n_2, \ldots, n_K$ are functions of n. We assume that there exists a constant $\gamma > 0$ such that for all $k = 1, 2, \ldots, K$, it holds that $n_k > \gamma n$ for all but a finite number of values of n.

Theorem 4.6.1

For the graph matching nomination scheme Φ^M, with the assumptions above

a. *If $\lambda_{1,1} \neq \lambda_{i,j}$ for all $(i,j) \neq (1,1)$ then there exists a real number $c > 0$ such that almost surely $\epsilon \leq c \log n$ for all but a finite number of values of n.*

b. *Almost surely the average precision of Φ^M converges to 1 as $n \to \infty$.*

c. *The mean average precision of Φ^M converges to 1 as $n \to \infty$.*

d. *If $\lambda_{i,j} \neq \lambda_{i',j'}$, for all $(i,j) \neq (i',j')$ then almost surely $\epsilon = 0$ for all but a finite number of values of n.*

Proof We begin with the proof of part (a) then part (d), and end with parts (b) and (c).

Proof of Theorem 4.6.1: part (a)

Let $\beta = \min |\lambda_{1,1} - \lambda_{i,j}|$ over all $(i,j) \neq (1,1)$. For any permutation $\psi \in \Psi_n$, the

graph matching objective function can be rewritten as

$$\begin{aligned} f(\psi) &= \sum_{\{i,j\}\in\binom{n}{2}} (\mathbb{1}[v_i \sim v_j] - \Lambda_{\widetilde{b}(\psi(v_i),\widetilde{b}(\psi(v_j))})^2 \\ &= \sum_{\{i,j\}\in\binom{n}{2}} \mathbb{1}[v_i \sim v_j]^2 + \Lambda^2_{\widetilde{b}(\psi(v_i),\widetilde{b}(\psi(v_j))} - 2\Lambda_{\widetilde{b}(\psi(v_i),\widetilde{b}(\psi(v_j))}\mathbb{1}[v_i \sim v_j] \end{aligned}$$

Minimizing $f(\psi)$ is equivalent to maximizing

$$g(\psi) = \sum_{\{i,j\}\in\binom{n}{2}} \Lambda_{\widetilde{b}(\psi(v_i),\widetilde{b}(\psi(v_j))}\mathbb{1}[v_i \sim v_j].$$

Let I denote the identity function, and without loss of generality let I be the true underlying permutation. Then

$$\begin{aligned} \mathbb{E}\left[g(I) - g(\psi)\right] &= \mathbb{E}\left[\sum_{\{i,j\}\in\binom{n}{2}} \left(\Lambda_{\widetilde{b}(v_i),\widetilde{b}(v_j)} - \Lambda_{\widetilde{b}(\psi(v_i),\widetilde{b}(\psi(v_j))}\right)\mathbb{1}[v_i \sim v_j]\right] \\ &= \sum_{\{i,j\}\in\binom{n}{2}} \left(\Lambda_{\widetilde{b}(v_i),\widetilde{b}(v_j)} - \Lambda_{\widetilde{b}(\psi(v_i),\widetilde{b}(\psi(v_j))}\right)\mathbb{P}[v_i \sim v_j] \\ &= \sum_{\{i,j\}\in\binom{n}{2}} \left(\Lambda_{\widetilde{b}(v_i),\widetilde{b}(v_j)} - \Lambda_{\widetilde{b}(\psi(v_i),\widetilde{b}(\psi(v_j))}\right)\Lambda_{\widetilde{b}(v_i),\widetilde{b}(v_j)} \\ &= \sum_{\{i,j\}\in\binom{n}{2}} \Lambda^2_{\widetilde{b}(v_i),\widetilde{b}(v_j)} - \Lambda_{\widetilde{b}(v_i),\widetilde{b}(v_j)}\Lambda_{\widetilde{b}(\psi(v_i),\widetilde{b}(\psi(v_j))} \\ &= \sum_{\{i,j\}\in\binom{n}{2}} \frac{1}{2}\left(\Lambda^2_{\widetilde{b}(v_i),\widetilde{b}(v_j)} - 2\Lambda_{\widetilde{b}(v_i),\widetilde{b}(v_j)}\Lambda_{\widetilde{b}(\psi(v_i),\widetilde{b}(\psi(v_j))} + \Lambda^2_{\widetilde{b}(v_i),\widetilde{b}(v_j)}\right) \end{aligned}$$

For any permutation ψ, we have

$$\sum_{\{i,j\}\in\binom{n}{2}} \Lambda^2_{\widetilde{b}(v_i),\widetilde{b}(v_j)} = \sum_{\{i,j\}\in\binom{n}{2}} \Lambda^2_{\widetilde{b}(\psi(v_i)),\widetilde{b}(\psi(v_j))}.$$

Thus

$$\begin{aligned}\mathbb{E}\left[g(I)-g(\psi)\right] &= \sum_{\{i,j\}\in\binom{n}{2}} \frac{1}{2}\left(\Lambda^2_{\widetilde{b}(v_i),\widetilde{b}(v_j)} - 2\Lambda_{\widetilde{b}(v_i),\widetilde{b}(v_j)}\Lambda_{\widetilde{b}(\psi(v_i),\widetilde{b}(\psi(v_j))} + \Lambda^2_{\widetilde{b}(\psi(v_i)),\widetilde{b}(\psi(v_j)))}\right) \\ &= \frac{1}{2}\sum_{\{i,j\}\in\binom{n}{2}} \left(\Lambda_{\widetilde{b}(v_i),\widetilde{b}(v_j)} - \Lambda_{\widetilde{b}(\psi(v_i),\widetilde{b}(\psi(v_j))}\right)^2 \qquad (4.5)\end{aligned}$$

Recall that Hoeffding's Inequality states that for any independent random variables $Z_1, Z_2, \ldots, Z_s$, such that for $i = 1, 2, \ldots, s$ the support of Z_i is $[a_i, b_i]$ and a_i, b_i are real numbers, then for any positive $t \in \mathbb{R}$,

$$\mathbb{P}\left[\left|\sum_{i=1}^{s} Z_i - \mathbb{E}\sum_{i=1}^{s}\right| \geq t\right] \leq 2\ e^{\left(\frac{-2t^2}{\sum_{i=1}^{s}(b_i-a_i)^2}\right)}$$

Consider when $s = \binom{n}{2}$ and $Z_i = \Lambda_{b(v_i),b(v_j)} - \Lambda_{b(\psi(v_i),b(\psi(v_j))}$, where i indexes the pair (u, v). Furthermore, take $t = \mathbb{E}[g(I) - g(\psi)]$,

$$\begin{aligned}&\mathbb{P}\left[\left|\sum_{i=1}^{s} Z_i - \mathbb{E}\sum_{i=1}^{s}\right| \geq t\right] \\ =\ &\mathbb{P}\Big[\ |g(I)-g(\psi)-\mathbb{E}\left[g(I)-g(\psi)\right]| \geq \mathbb{E}[g(I)-g(\psi)]\Big] \\ =\ &\mathbb{P}\Big[-(g(I)-g(\psi)-\mathbb{E}\left[g(I)-g(\psi)\right]) \geq \mathbb{E}[g(I)-g(\psi)]\Big|g(I)-g(\psi)-\mathbb{E}\left[g(I)-g(\psi)\right] < 0\Big] \\ &+\mathbb{P}\Big[g(I)-g(\psi)-\mathbb{E}\left[g(I)-g(\psi)\right] \geq \mathbb{E}[g(I)-g(\psi)]\Big|g(I)-g(\psi)-\mathbb{E}\left[g(I)-g(\psi)\right] \geq 0\Big] \\ =\ &\mathbb{P}\Big[0 \geq g(I)-g(\psi)\Big|g(I)-g(\psi) \leq \mathbb{E}\left[g(I)-g(\psi)\right]\Big] \\ &+\mathbb{P}\Big[g(I)-g(\psi) \geq 2\mathbb{E}[g(I)-g(\psi)]\Big|g(I)-g(\psi) \geq \mathbb{E}\left[g(I)-g(\psi)\right]\Big]\end{aligned}$$

Notice that $\mathbb{E}\left[g(I)-g(\psi)\right] \geq 0$ (Equation 4.5), and applying Hoeffding's Inequality

$$\begin{aligned}\mathbb{P}\Big[g(I)-g(\psi) \leq 0\Big] &\leq \mathbb{P}\left[\left|\sum_{i=1}^{s} Z_i - \mathbb{E}\sum_{i=1}^{s}\right| \geq t\right] \qquad (4.6) \\ &\leq 2\ e^{\left(\frac{-2\mathbb{E}^2[g(I)-g(\psi)]}{2\mathbb{E}^2[g(I)-g(\psi)]}\right)} = 2\ e^{-\mathbb{E}[g(I)-g(\psi)]}. \qquad (4.7)\end{aligned}$$

Now we bound Equation 4.7. Let the pair of vertices $\Omega_0 \subseteq \binom{n}{2}$ such that $v_i, v_j \in V_1$, yet $\psi(v_i), \psi(v_j) \notin V_1$, and the pair $\Omega_1 \subseteq \binom{n}{2}$ such that $v_i, v_j \in V_1$, and exactly one of $\psi(v_i)$ and $\psi(v_j)$ are in V_1.

Let $\epsilon_{\psi'} = |\{v_i \in V_1 : \psi(v_i) \notin V_1\}|$ for any permutations ψ', i.e. $\epsilon_{\psi'}$ is the number of vertices ψ' wrongly predicted to be in V_1. Recall that $\beta = \min |\lambda_{1,1} - \lambda_{i,j}|$, now by Equation 4.5 we have

$$\begin{aligned}
\mathbb{E}\left[g(I) - g(\psi)\right] &\geq \frac{1}{2} \sum_{\{v_i,v_j\} \in \Omega_0} \left(\Lambda_{\widetilde{b}(v_i),\widetilde{b}(v_j)} - \Lambda_{\widetilde{b}(\psi(v_i)),\widetilde{b}(\psi(v_j))}\right)^2 \\
&\quad + \frac{1}{2} \sum_{\{i,j\} \in \Omega_1} \left(\Lambda_{\widetilde{b}(v_i),\widetilde{b}(v_j)} - \Lambda_{\widetilde{b}(\psi(v_i)),\widetilde{b}(\psi(v_j))}\right)^2 \\
&\geq \frac{1}{2}\beta^2 \frac{\epsilon_\psi(\epsilon_\psi - 1)}{2} + \frac{1}{2}\beta^2(n_1 - \epsilon_\psi)\epsilon_\psi = \frac{1}{2}\beta^2\epsilon_\psi \left(\frac{2n_1 - \epsilon_\psi - 1}{2}\right) \\
&\geq \frac{1}{2}\beta^2\epsilon_\psi \left(\frac{2n_1 - n_1 - 1}{2}\right) = \frac{1}{2}\beta^2\epsilon_\psi \left(\frac{n_1 - 1}{2}\right) = \frac{1}{4}\beta^2\epsilon_\psi \left(n_1 - 1\right).
\end{aligned}$$

It holds that

$$\frac{n-1}{k} \geq \frac{n}{k+1} \tag{4.8}$$

for fixed positive integer k and large $n >> k$. In our case, $n = n_1$ and $k = \epsilon_\psi$,

$$\begin{aligned}
\mathbb{E}\left[g(I) - g(\psi)\right] &\geq \frac{1}{4}\beta^2\epsilon_\psi \left(n_1 - 1\right) \geq \frac{1}{5}\beta^2 n_1 \epsilon_\psi &(4.9)\\
&\geq \frac{\beta^2\gamma}{5} n\epsilon_\psi, &(4.10)
\end{aligned}$$

when n is large enough.

Let Ψ_n^ϵ be the set of permutations such that $\epsilon_\psi \geq \frac{10}{\beta^2\gamma}\log n$, and note that $|\Psi_n^\epsilon| \leq |\Psi_n| \leq n^n$. Then Equations 4.7, 4.10 and the sub-additivity of probability measure

give us

$$\mathbb{P}\Big[\exists\ \psi \in \Psi_n^\epsilon : g(I) \leq g(\psi)\Big] \leq \sum_{\psi\in\Psi_n^\epsilon} 2\ e^{-\frac{\beta^2\gamma}{5} n\epsilon_\psi \frac{10}{\beta^2\gamma}\log n} = \sum_{\psi\in\Psi_n^\epsilon} 2\ e^{-2n\epsilon_\psi \log n} \quad (4.11)$$

$$\leq 2n^n\ e^{-2n\epsilon_\psi \log n} = 2\ e^{n\log n - 2n\epsilon_\psi \log n} = \frac{2}{n^n}. \quad (4.12)$$

This is finitely summable over all n. Thus by the Borel-Cantelli Lemma, almost surely there are at most a finite number of values of n for which any optimal graph matching ψ has at least $\frac{10}{\beta^2\gamma}\log n$ number of mismatched vertices. ∎

Proof of Theorem 4.6.1: part d

For part d, let $\widetilde{\beta} = \min|\lambda_{i,j} - \lambda_{i',j'}|$ for $(i,j) \neq (i',j')$, and let $\epsilon_{\psi,k} = |v_i \in V_k : \psi(v_i) \notin V_k|$ for $k \in \{1,2,3,\dots K\}$, in particular $\epsilon_{\psi,1} = \epsilon_\psi$. For $k = 1,2,\dots K$, let $\Omega_{0,k} \subseteq \binom{V}{2}$ be the set of $\{i,j\} \in \binom{V}{2}$ such that both $v_i, v_j \in V_k$, but $v_{\psi(v_i)}, v_{\psi(v_j)} \notin V_i$. Furthermore, let $\Omega_{1,k} \subseteq \binom{V}{2}$ be the set of $\{i,j\} \in \binom{V}{2}$ such that $v_i, v_j \in V_k$, and exactly one of $v_{\psi(v_i)}$ or $v_{\psi(v_j)}$ are in V_k.

Let $g(\psi)$ be defined as the same in Equation 4.5. We use Equation 4.10 and follow

the same steps as equation 4.12 and 4.10,

$$\begin{aligned}
& \mathbb{E}\left[g(I) - g(\psi)\right] \\
= & \sum_{k=1}^{K}\left[\frac{1}{2}\sum_{\{i,j\}\in\binom{n}{2}}\left(\Lambda_{\widetilde{b}(v_i),\widetilde{b}(v_j)} - \Lambda_{\widetilde{b}(\psi(v_i),\widetilde{b}(\psi(v_j))}\right)^2\right] \\
\geq & \sum_{k=1}^{K}\left[\frac{1}{2}\sum_{\{i,j\}\in\Omega_{0,k}}\left(\Lambda_{\widetilde{b}(v_i),\widetilde{b}(v_j)} - \Lambda_{\widetilde{b}(\psi(v_i),\widetilde{b}(\psi(v_j))}\right)^2\right. \\
& \left. +\frac{1}{2}\sum_{\{i,j\}\in\Omega_{1,k}}\left(\Lambda_{\widetilde{b}(v_i),\widetilde{b}(v_j)} - \Lambda_{\widetilde{b}(\psi(v_i),\widetilde{b}(\psi(v_j))}\right)^2\right] \\
\geq & \sum_{k=1}^{K}\left[\frac{1}{2}\widehat{\beta}^2\frac{\epsilon_{\psi,k}(\epsilon_{\psi,k}-1)}{2} + \frac{1}{2}\widehat{\beta}^2(n_1-\epsilon_{\psi,k})\epsilon_{\psi,k}\right] \\
= & \sum_{k=1}^{K}\left[\frac{1}{2}\widehat{\beta}^2\epsilon_{\psi,k}\left(\frac{2n_1-\epsilon_{\psi,k}-1}{2}\right)\right] \\
\geq & \sum_{k=1}^{K}\left[\frac{1}{2}\widehat{\beta}^2\epsilon_{\psi,k}\left(\frac{2n_1-n_1-1}{2}\right)\right] \\
= & \sum_{k=1}^{K}\left[\frac{1}{2}\widehat{\beta}^2\epsilon_{\psi,k}\left(\frac{n_1-1}{2}\right)\right] = \sum_{k=1}^{K}\left[\frac{1}{4}\widehat{\beta}^2\epsilon_{\psi,k}\left(n_1-1\right)\right] \\
\geq & \sum_{k=1}^{K}\left[\frac{\widehat{\beta}^2\gamma}{5}n\epsilon_{\psi,k}\right] = \frac{\widehat{\beta}^2\gamma}{5}n\sum_{k=1}^{K}\epsilon_{\psi,k},
\end{aligned}$$

when n is large enough.

Let $\Psi_n^{\epsilon,k}$ to be the set of $\psi \in \Psi_n$ such that $\sum_{k=1}^{K}\epsilon_{\psi,k} \geq \frac{10}{\widehat{\beta}^2\gamma}\log n$. Also note the fact that $|\Psi_n^{\epsilon,k}| \leq |\Psi_n| \leq n^n$, and following the same steps of Equation 4.11,

$$\mathbb{P}\Big[\exists\psi \in \Psi_n^{\epsilon,k} : g(I) \leq g(\psi)\Big] \leq \sum_{\psi\in\Psi_n^{\epsilon,k}} 2\ e^{-2n\ \log n} \leq 2\ e^{n\ \log n - 2n\ \log n} = \frac{2}{n^n}.$$

∎

Proof of Theorem 4.6.1: parts b and c

From the above proofs, we have shown that almost surely for large enough n, it holds that $\epsilon \leq \frac{10}{\beta^2\gamma}\log n$. Thus almost surely

$$\begin{aligned}
1 - AP(\Phi^M) &= 1 - \frac{1}{n_1}\sum_{j=1}^{n_1}\frac{|\{i \in \{1,2,3,\dots j\} : v_{\varphi^M(i)} \in V_1\}|}{j} \\
&= \frac{1}{n_1}\sum_{j=1}^{n_1}\frac{n_1 - |\{i \in \{1,2,3,\dots j\} : v_{\varphi^M(i)} \in V_1\}|}{j} \\
&= \frac{1}{n_1}\sum_{j=1}^{n_1}\frac{|\{i \in \{1,2,3,\dots j\} : v_{\varphi^M(i)} \notin V_1\}|}{j} \\
&\leq \frac{1}{n_1}\sum_{j=1}^{n_1}\frac{\epsilon}{j} = \epsilon\frac{1}{n_1}\sum_{j=1}^{n_1}\frac{1}{j} \leq \frac{10}{\beta^2\gamma}\log n\left(\frac{1}{n_1}\sum_{j=1}^{n_1}\frac{1}{j}\right) \\
&\leq \frac{10}{\beta^2\gamma}\frac{\log n\ (1+\log n_1)}{n_1},
\end{aligned}$$

which converges to 0 as $n \to \infty$. The mean average precision of Ψ^M is bounded by 1 and the average precision of Ψ^M. Thus the mean average precision $MAP(\Psi^M)$ also converges to 1 as $n \to \infty$. ∎

4.6.1.1 Phenomenon of Inversion by SGM Nomination

For certain cases, graph matching vertex nomination nominates vertices in reverse order. This means that the vertices nominated to be in block V_1 are in order of increasing empirical probabilities to be in V_1 rather than decreasing probabilities. For example, consider vertex nomination for a graph G distributed as $\kappa(S, M, \Lambda)$, where

$$\Lambda = \begin{bmatrix} 0.2 & 0.15 \\ 0.15 & 0.1 \end{bmatrix}, S = [15, 0], M = [50, 50].$$

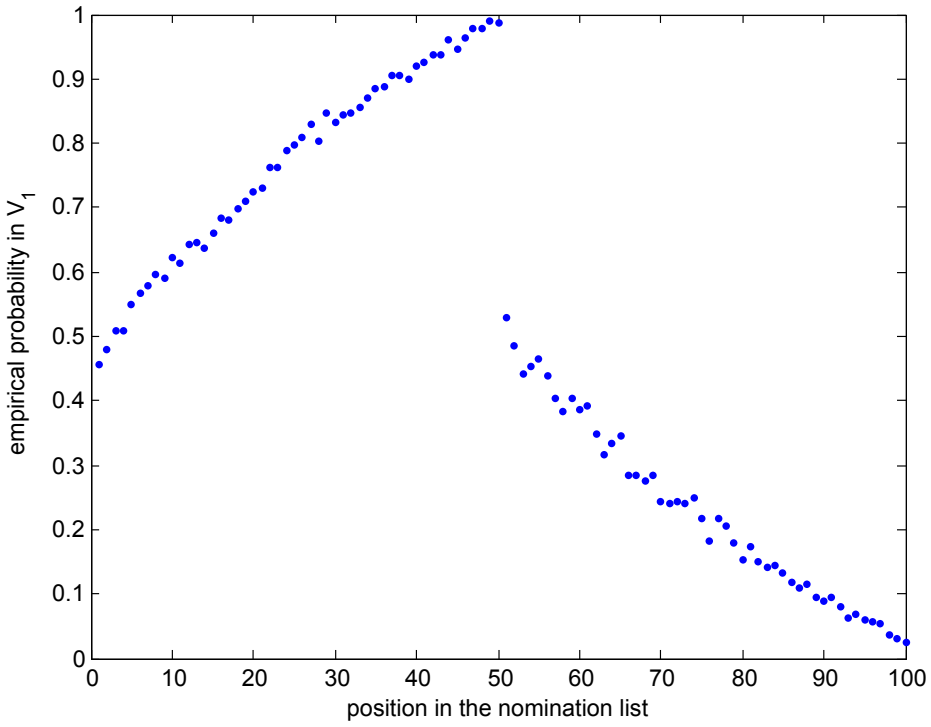

Figure 4.2: Example of accuracy inversion due to the graph matching vertex nomination.

Figure 4.2 shows the plot of nomination accuracy for 1000 Monte Carlo simulations.

In order to understand this inversion phenomenon, we consider the simplest case of an undirected two block model $\kappa(N, \Lambda)$,

$$\Lambda = \begin{bmatrix} p & r \\ r & q \end{bmatrix}.$$

This phenomenon of inversion is empirically explored in Figure 4.3. Our simulation parameters are $S = [15, 0]$, $M = [50, 50]$, $p \in \{0, 0.1, 0.2, \ldots, 1\}$, $q \in \{0, 0.1, 0.2, \ldots, 1\}$,$r \in \{0.1, 0.15, 0.2, \ldots, 1\}$, over 100 simulations for each pixel. The slope of the empirical probability is computed for the vertices in $\widehat{V_1}$. The red areas of the plot indicate the occurrence of inversion due to the graph matching vertex nomination.

We attempt a heuristic approach to explain this phenomenon. For simplicity, suppose both blocks have the same size n. The reversal only occurs in areas when the algorithm has incorrectly labeled some vertices as from V_1. Let L be the error of the graph matching vertex nomination scheme, then Ln vertices are matched to the wrong block.

Recall the graph matching nomination scheme is matching A to $\widetilde{A}$. We can heuristically estimate the regions of slope reversal by computing the expected residual r_i for i of vertices matched to V_1. Recall that

$$r_i = \sum_{v_j \in V} (\mathbb{1}[v_i \sim v_j] - \Lambda_{\widetilde{b}(\psi(v_i)),\widetilde{b}(\psi(v_j))})^2,$$

thus r_i is a sum of n terms, with each term as $\lambda_{i,\cdot}$ or $1-\lambda_{i,\cdot}$. Let us group the residuals

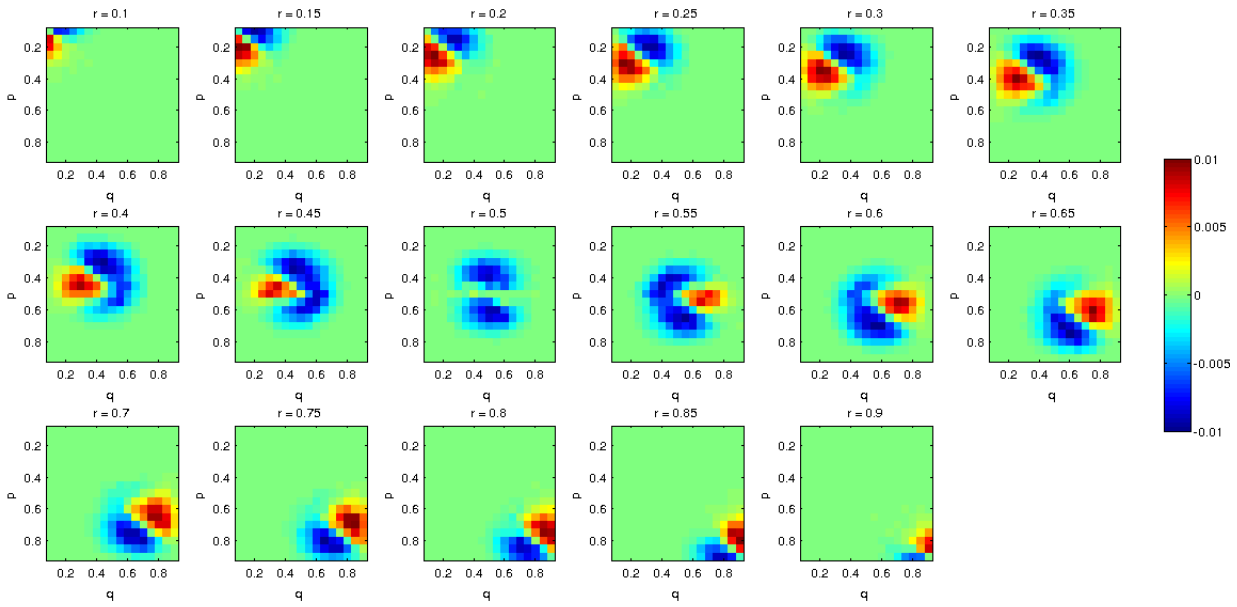

Figure 4.3: Plot of slope of the GM vertex nomination scheme accuracy with parameters $S = [15, 0], M = [50, 50]$, varying $p \in \{0, 0.1, 0.2, \ldots, 1\}$, $q \in \{0, 0.1, 0.2, \ldots, 1\}$,$r \in \{0.1, 0.15, 0.2, \ldots, 1\}$, and 100 simulations for each pixel.

of the same block together as $r_{i,b=k}$. Thus

$$r_{i,b=k} = \sum_{v_j \in V : \tilde{b}_j = k} (\mathbb{1}[v_i \sim v_j] - \Lambda_{\tilde{b}(\psi(v_i)),\tilde{b}(\psi(v_j))})^2.$$

The number of terms which are $(1-\lambda_{i,1})^2$ can be approximated as $X_{i,1} \sim$Binomial$(N(1), \lambda i, 1)$, and the number of terms which are $\lambda_{i,1}^2$ is $N(1) - X_{i,1}$. Then the expected value of $r_{i,b=k}$ is

$$\mathbb{E}[r_{i,b=k}] = (1 - \lambda_{i,1})^2 N(1) * \lambda_{i,1} + \lambda_{i,1}^2 N(1) * (1 - \lambda i, 1)$$

For convenience, define the function

$$f(a, b, L) = (1 - a)^2 Lb + a^2 L(1 - b).$$

Now we can estimate the expected residual r_i where $v_i \in V_1$,

$$\mathbb{E}[r_{V_1}] = f(p, p, 1 - L) + f(p, r, L) + f(r, r, 1 - L) + f(r, p, L).$$

Similarly the expected residual r_i for $v_i \in V_2$,

$$\mathbb{E}[r_{V_2}] = f(p, r, 1 - L) + f(p, q, L) + f(r, q, 1 - L) + f(r, r, L).$$

Notice that this relies on knowing the error. The actual distribution of the number of edges is not binomial, because the nominated vertices to be in V_1 are not distributed as Bernoulli(p). The nominated vertices are chosen via graph matching, this creating a biased sampling of the vertices in V_1.

Closer examination of the function $f(a, b, L)$ yields

$$\begin{aligned} f(a,b,L) - f(a,c,L) &= (1-a)^2Lb + a^2L(1-b) - [(1-a)^2Lc + a^2L(1-c)] \\ &= (1-a)^2L(b-c) + a^2L[1-b-(1-c)] \\ &= (1-a)^2L(b-c) + a^2L(c-b) \\ &= L(b-c)[(1-a)^2 - a^2]. \end{aligned}$$

If $a < 0.5$ and $b > c$, then $f(a, b, L) > f(a, c, L)$. This means, if $p < 0.5$, $p > q$, $q = r$, then

$$\begin{aligned} \mathbb{E}[r_{V_1}] - \mathbb{E}[r_{V_2}] &= f(p,p,1-L) + f(p,r,L) + f(r,r,1-L) + f(r,p,L) \\ &\quad -[f(p,r,1-L) + f(p,q,L) + f(r,q,1-L) + f(r,r,L)] \\ &= f(p,p,1-L) - f(p,r,1-L) + f(r,p,L) - f(r,r,L) > 0 \\ &\Rightarrow \mathbb{E}[r_{V_1}] > \mathbb{E}[r_{V_2}]. \end{aligned}$$

Thus in this case residuals of vertices in V_2 have lower expected values than V_1 vertices. Notice this occurs regardless of the value of the matching error L. This means that vertices with larger residuals are more likely to be from V_1. This behavior is verified in the simulation data as seen in Figure 4.4. The slope of when $q = r$ is plotted.

In the case when $q \neq r$, the boundary between the positive and negative slope is not as clear. Further complicating matters, the boundary depends on the value of L. Using the estimated accuracy from the simulation results, we can compute when $\mathbb{E}[r_{V_2}] > \mathbb{E}[r_{V_1}]$ and compare with the case when the slope is positive (Figure 4.5). Our heuristic derivation predicts the inversion behavior.

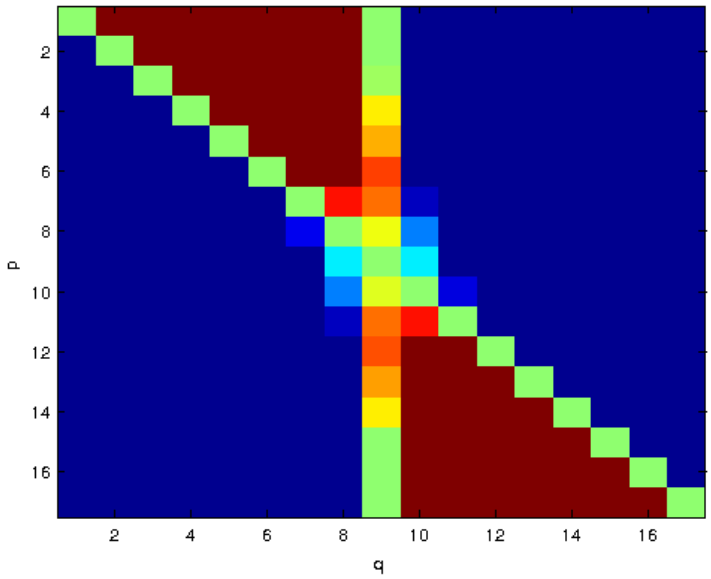

Figure 4.4: Plot of slope of the nomination accuracy when $r = q$.

For cases with more than 2 blocks the inversions are not significantly better behaved. If the parameters in the Λ matrix $\lambda_{ij} < 0.5$ or $\lambda_{ij} > 0.5$ for all $i, j \in 1, 2, \ldots, n$, then the percentage of inversions slightly decreases with respect to the number of blocks. When there are no restrictions on λ_{ij} then the percentage of inversions drops dramatically with respect to the number of blocks.

Thus we develop an alternative method, which still utilizes graph matching, but does not suffer from inversions.

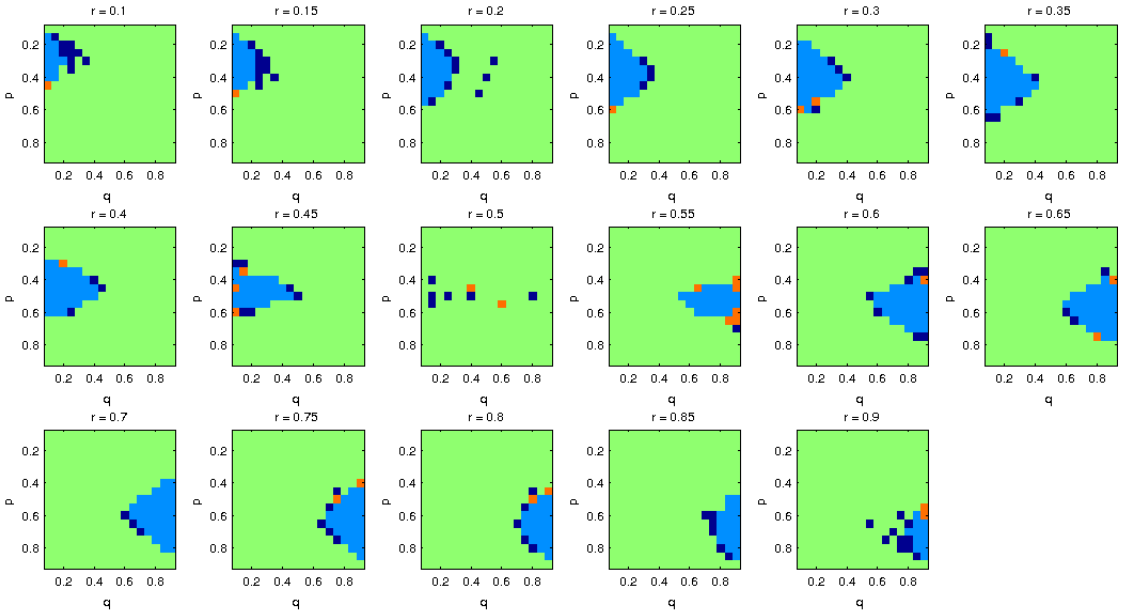

Figure 4.5: Plot of GM vertex nomination scheme inversion compared with predicted inversion. The plot is green when there is neither inversion nor predicted inversion. The light blue denotes inversion and it is correctly predicted. The dark blue area denotes inversion, but not predicted. The orange area denotes predicted inversion, but not occurred.

4.6.2 Most Likely Partition

As an alternative to residuals, we can use probability $\mathbb{P}[v \in V_1|G]$ rank the vertices. We can compute the probability of observing the graph G given the block membership function b as in Equation 4.1,

$$\mathbb{P}[G|b] = \prod_{i=1}^{n} \prod_{j=i+1}^{n} \lambda_{b_i,b_j}^{a_{ij}} (1 - \lambda_{b_i,b_j})^{1-a_{ij}}$$

Taking a log of both sides this becomes,

$$\begin{aligned}
\log(\mathbb{P}[G|b]) &= \log\left(\prod_{i<j} \lambda_{b_i,b_j}^{a_{ij}} (1 - \lambda_{b_i,b_j})^{1-a_{ij}}\right) \\
&= \sum_{i<j} \log(\lambda_{b_i,b_j}^{a_{ij}} (1 - \lambda_{b_i,b_j})^{1-a_{ij}}) \\
&= \sum_{i<j} a_{ij} \log(\lambda_{b_i,b_j}) + (1 - a_{ij}) \log(1 - \lambda_{b_i,b_j}) \\
&= \sum_{i<j} \log(1 - \lambda_{b_i,b_j}) + a_{ij}(\log(\lambda_{b_i,b_j}) - \log(1 - \lambda_{b_i,b_j})) \\
&= \sum_{i<j} \log(1 - \lambda_{b_i,b_j}) + a_{ij} \log\left(\frac{\lambda_{b_i,b_j}}{1 - \lambda_{b_i,b_j}}\right) \\
&= C + \sum_{i<j} a_{ij} \log\left(\frac{\lambda_{b_i,b_j}}{1 - \lambda_{b_i,b_j}}\right).
\end{aligned}$$

Notice that maximizing the probability is equivalent to the graph matching objective of matching the adjacency matrix A with the matrix

$$\widetilde{A} = \left[\log\left(\frac{\lambda_{b_i,b_j}}{1 - \lambda_{b_i,b_j}}\right)\right].$$

Thus we can use the graph matching algorithm to estimate the most likely partition of the graph,

$$P^* = \underset{P \in \mathcal{P}_n}{\arg\max}\, \mathbb{P}[v_i \in V|G].$$

After finding the most likely partition, we still need a method to generate the nomination list. Recall the canonical vertex nomination scheme nominates vertices by

$$\mathbb{P}[v_i \in V_1 | G] = \frac{\sum_{b'' \in \mathcal{B}: b''_i = 1} \mathbb{P}[G, b'']}{\sum_{b' \in \mathcal{B}} \mathbb{P}[G, b']}. \tag{4.13}$$

To approximate this, we sum over the set of all neighboring block membership functions, a much smaller set of functions. Let the set of all neighboring block membership functions of $\widehat{b}^*$ be denoted by $\mathcal{B}_{\widehat{b}^*}$, where $\widehat{b}^*$ is the estimated most likely block membership function from graph matching. Then we order the vertices by

$$\prod_{v_i \in V_1^{\widehat{b}^*}, v_j \in V \setminus V_1^{\widehat{b}^*}} \frac{\mathbb{P}[G, \widehat{b}^*(v_i, v_j)]}{\mathbb{P}[G, \widehat{b}^*]}. \tag{4.14}$$

for **most likely partition (ML) nomination scheme**. Notice that Equation 4.14 is similar to the numerator of Equation 4.13 except that Equation 4.14 uses a product instead of a sum. Using a product instead of a sum prevents probabilities of large values to dominate. Also empirical results show that ranking using Equation 4.13 yields better performance than using Equation 4.13.

4.7 Mixed Membership Stochastic Block-model

For comparison with the other methods, we have also included a mixed membership stochastic block model (see Section 2.5). Recall in the model that for all $i \in [n]$,

a mixed block membership distribution is drawn as $\pi_i \sim$Dirichlet(α) for the vertex v_i. The probability distribution $\pi_i = [\pi_{ij}]$ is the distribution of blocks for vertex v_i. Thus π_{i1} is the probability that vertex v_i acts as a vertex from block V_1. If $\pi_{i1} = 1$, then with probability 1 vertex v_i acts as a vertex from block V_1.

Just like in the spectral partitioning nomination scheme, we can create a vertex nomination scheme by ordering the vertices by π_{i1}. Let us define the mixture membership stochastic block-model vertex nomination scheme as

$$\Phi_G^{MMB} = (v_{\varphi_1^{MMB}}, v_{\varphi_2^{MMB}}, v_{\varphi_3^{MMB}}, \dots v_{\varphi_n^{MMB}}),$$

such that for all $i, j \in \{1, 2, 3, \dots n\}$, φ_i, φ_j satisfy the property that

$$i < j \text{ if and only if } \pi_{i1} \geq \pi_{j1}$$

Mixed membership stochastic block model is a different model from the stochastic block model $\kappa(k, \Lambda)$. Thus, under the idealized setting of the stochastic block model, Φ_G^{MMB} is not designed to perform well compared to our vertex nomination schemes, which are asymptotically optimal under the stochastic block model. Real data does not necessarily fit the stochastic block model. Thus, in our real data experiment, we compare the performance of Φ_G^{MMB} with the other four schemes.

4.8 Performance

In this section, we compare the performance of the nomination schemes on simulation and real data.

4.8.1 Simulated Data

All four vertex nomination schemes: canonical (CAN), Metropolis-Hastings (MH), most likely partition (ML), and spectral partitioning (SP) have computational limitations. The canonical scheme is only feasible in very small graphs, on the order of 10 vertices. Most likely partition nomination scheme is feasible up to order of 1000 vertices, and spectral partitioning nomination scheme is feasible on even bigger graphs, however its performance on small graphs (order 10) is no better than chance. The Metropolis-Hastings vertex nomination scheme is feasible on many scales. We fixed burn-in at 25000 samples, and 500000 samples after burn-in. In practice, the number of samples should be adjusted for every problem. We fixed the number of samples for simplicity.

These simulation experiments are designed to compare the schemes at various scales. In the experiments we have $K = 3$ blocks in a stochastic block model. In order to maintain an adequate level of difficulty at different scales, the parameters M, N, and Λ are also scaled. Define

$$\Lambda(\alpha) = \alpha \begin{bmatrix} .5 & .3 & .4 \\ .3 & .8 & .6 \\ .4 & .6 & .3 \end{bmatrix} + (1-\alpha) \begin{bmatrix} .5 & .5 & .5 \\ .5 & .5 & .5 \\ .5 & .5 & .5 \end{bmatrix},$$

which can be thought of as a mixture model between a 3-block stochastic block model and an Erdős-Rényi model with $p = 0.5$. Larger α contains more signal, because with a small α the model is closer to an Erdős-Rényi model creating more indistinguishable

blocks. Also define $\vec{n}(\beta) = \beta[4; 3; 3]$. For the three experiments we have the following parameters:

$$N = \vec{n}(1) = \begin{bmatrix} 4 \\ 3 \\ 3 \end{bmatrix}, \quad M = \begin{bmatrix} 4 \\ 0 \\ 0 \end{bmatrix}, \quad \Lambda = \Lambda(1) = \begin{bmatrix} .5 & .3 & .4 \\ .3 & .8 & .6 \\ .4 & .6 & .3 \end{bmatrix};$$

$$N = \vec{n}(50) = \begin{bmatrix} 200 \\ 150 \\ 150 \end{bmatrix}, \quad M = \begin{bmatrix} 20 \\ 0 \\ 0 \end{bmatrix}, \quad \Lambda = \Lambda(.3) = \begin{bmatrix} .50 & .44 & .47 \\ .44 & .59 & .53 \\ .47 & .53 & .44 \end{bmatrix};$$

$$N = \vec{n}(1000) = \begin{bmatrix} 4000 \\ 3000 \\ 3000 \end{bmatrix}, \quad M = \begin{bmatrix} 40 \\ 0 \\ 0 \end{bmatrix}, \quad \Lambda = \Lambda(.13) = \begin{bmatrix} .500 & .474 & .487 \\ .474 & .539 & .513 \\ .487 & .513 & .474 \end{bmatrix}.$$

Figure 4.6 compares vertex nomination scheme via CAN, MH, ML, and SP. From the figure, notice that ML vertex nomination scheme is very close to CAN, demonstrating that ML could be used as an approximation to CAN vertex nomination scheme. Surprisingly, MH vertex nomination scheme performs better than the CAN vertex nomination scheme. This may be caused by CAN vertex nomination scheme's improper handling of symmetric graphs. Also 500000 samples is significant oversampling, since the $\binom{10}{4,3,3} = 4200$.

The small experiment is repeated 50000 items, the medium experiment is repeated 200 times and the large experiment is repeated 100 times.

Next in figure 4.7, ML vertex nomination scheme is plotted with SP vertex nom-

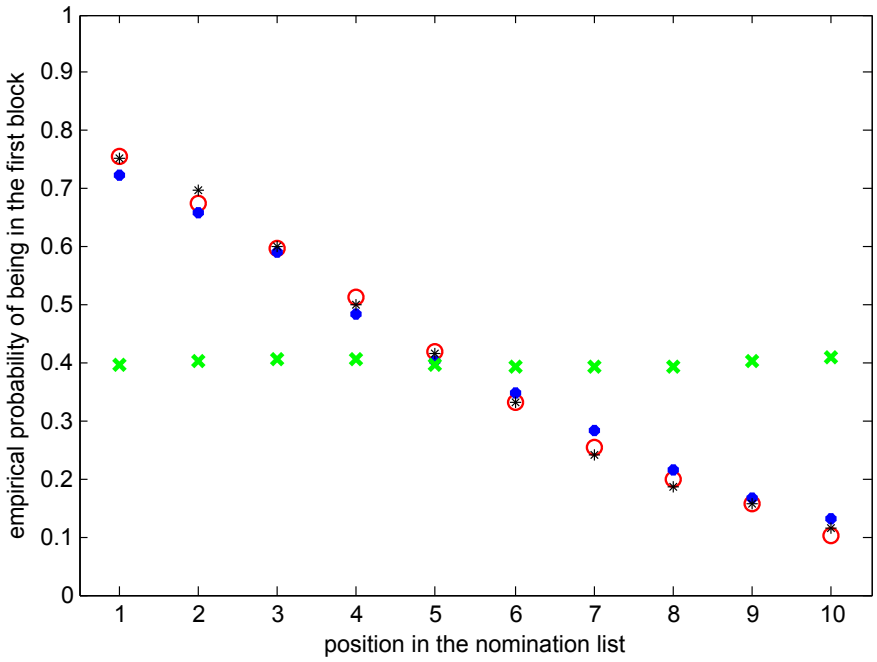

Figure 4.6: We compare four nomination schemes: CAN (red), MH (black), ML (blue), and SP (green). The simulated graphs have 4 seed and 10 non-seed vertices. CAN, MH, and ML all perform similarly, but SP performs about chance.

ination scheme. Canonical vertex nomination scheme is currently computationally infeasible at 500 vertices, thus it is not plotted. SP (blue) is able to perform admirably, but MH (black) and ML (red) are significantly better. This demonstrates that in the situation that is possible to use graph matching, it should be favored over spectral-partitioning.

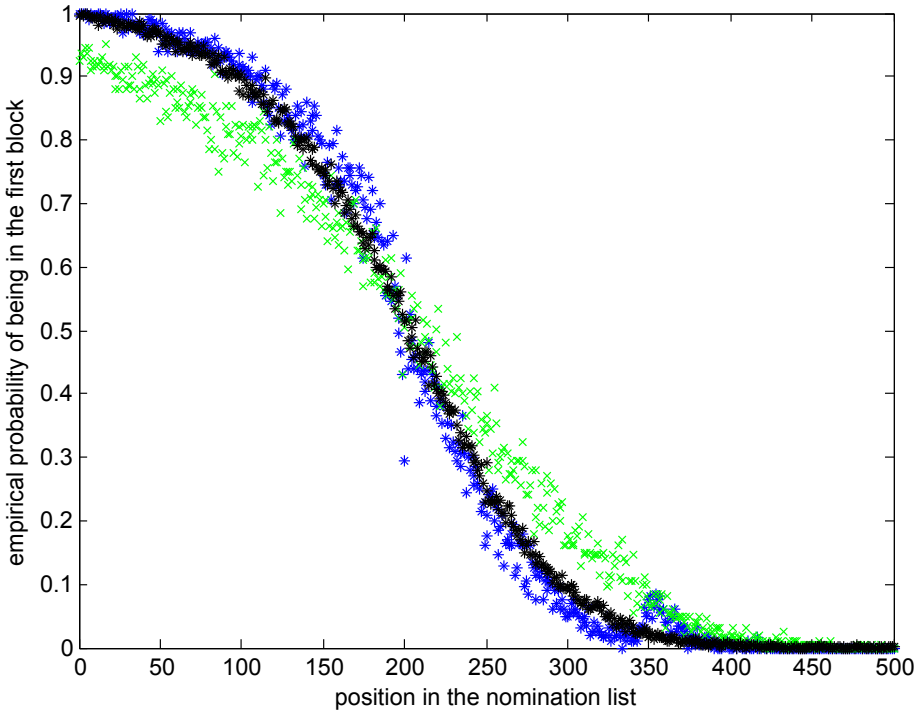

Figure 4.7: We compare three nomination schemes: MH, ML, and SP. SP (blue) is able to perform admirably, but MH (black) and ML (red) are significantly better. This simulation has 20 seed and 500 non-seed vertices.

In the Figure 4.8, we have the third experiment where only MH and SP are computationally practical. At this scale, seeded graph matching's complexity of $O(n^3)$ is very noticeable. Later in Section 5, we present a variation of the graph matching algorithm with lower complexity.

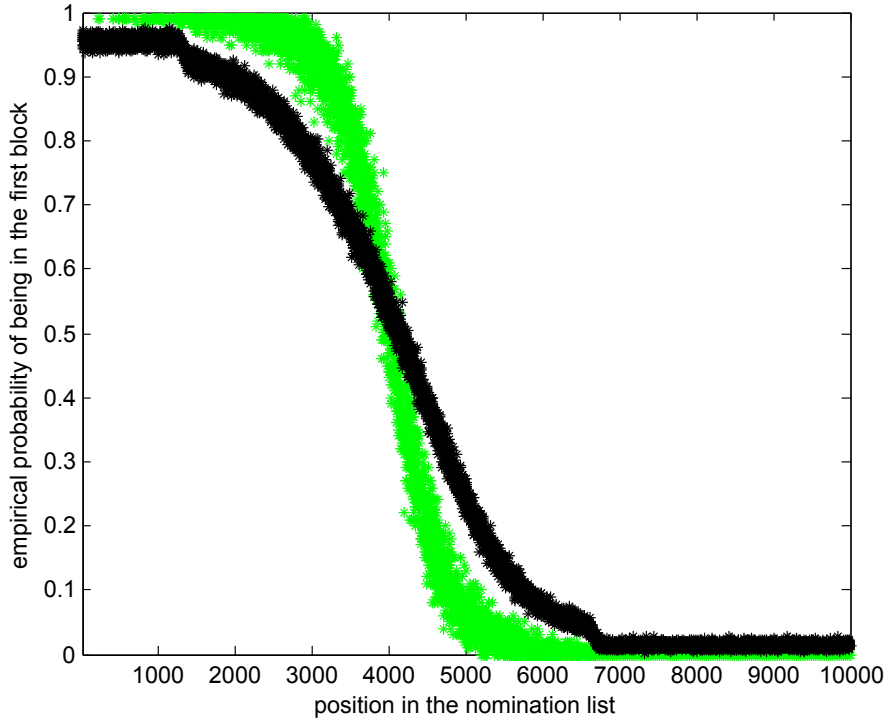

Figure 4.8: We compare two nomination schemes: ML and SP. SP (blue) performs better than MH. This simulation has 40 seed and 10000 non-seed vertices.

MAP	CAN	ML	MH	SP
$n = 10$	0.6948	0.6515	0.6993	0.3991
n=500	N\A	0.9303	0.9452	0.7330
n=10000	N\A	N\A	0.9281	0.9859

Table 4.2: MAP values for simulated data experiments.

Runtime	CAN	ML	MH	SP
$n = 10$	1.4	.04	138	.02
n=500	N\A	286	191	.8
n=10000	N\A	N\A	546	534

Table 4.3: Runtime in seconds for simulated data experiments.

Note that since we are using $\vec{n}(\beta)$, 0.4 is chance accuracy for all three experiments, and 0.4 is also chance MAP.

These experiments run in Matlab on a standard modern desktop. The following table presents the average runtime of vertex nomination on one graph.

4.8.2 Real Data

In this section, we explore the effectiveness of our vertex nomination scheme on real data. Four datasets are presented and performance of four vertex nomination schemes are compared. The four scheme we compare are MH, ML, spectral, and MMB

vertex nomination schemes. The first dataset is a email communication network from the Enron Corporation. The next dataset is the neuron Network of a worm. The third dataset is a weblog reference network. Finally we present a movie graph, created from info-boxes on Wikipedia.

The MH and ML vertex nomination scheme need knowledge of the parameters. We do not assume Λ is provided as it was in the synthetic examples. Instead, we estimate the Λ via the empirical estimate using the seed vertices as a sample. For example, to estimate λ_{12} we sum up the number of edges between vertices in V_1 and V_2 and divide by $n_1 n_2$.

4.8.2.1 Enron Email Data

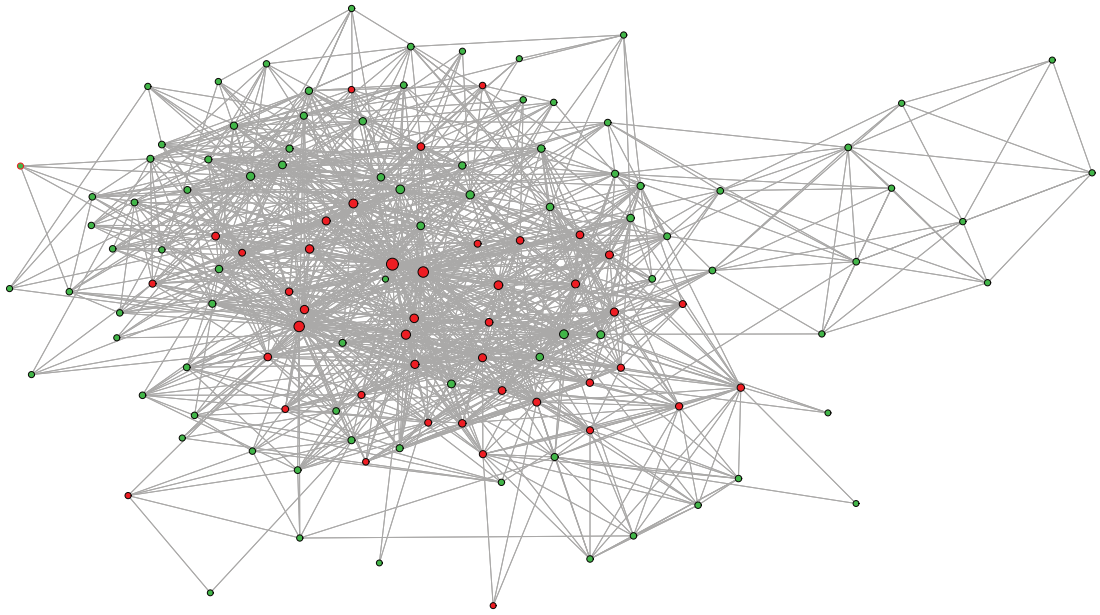

Figure 4.9: Visualization of the Enron graph data.

The data to be used is the email graph from the Enron Corporation. There are more than 600,000 emails sent between Enron employees, with 184 distinct email addresses over 198 weeks from 1998 to 2002. The graph we use has 130 vertices, where each vertex is an employee's email address in Enron with known job title. Two vertices are adjacent if an email is sent from either email address to the other in either direction. The 130 vertices are then divided into 2 blocks. One block contained the "upper-echelon" or higher management (43 vertices). This block contained job titles of CEO, president, vice president, chief manager, company attorney, and chief employee. The other block contained the "lower-echelon," containing the job titles employee, employee administrative, specialist, analyst, trader, director, and manager (87 vertices). Of the 130 vertices 30 are chosen as seeds, with 10 from the upper-echelon and 20 from the lower-echelon. This experiment is repeated 30000 times.

Notice that before in the model we assumed Λ to be a known parameter. This is not possible in this case, since the underlying model is not a stochastic block model. Furthermore even if the underlying model is a stochastic block model, we would not know the value of Λ. We estimate Λ using the seed vertices, for example with the 10 upper-echelon seeds we have the graph $G[U_1]$ (100 Bernoulli trials) to estimate λ_{11}.

4.8.2.2 *Caenorhabditis elegans* Neuron Network

One well studied neural network is of *Caenorhabditis elegans* (*C.elegans*) neural network. *C.elegans* are a nematode (roundworm), about 1 mm in length that live

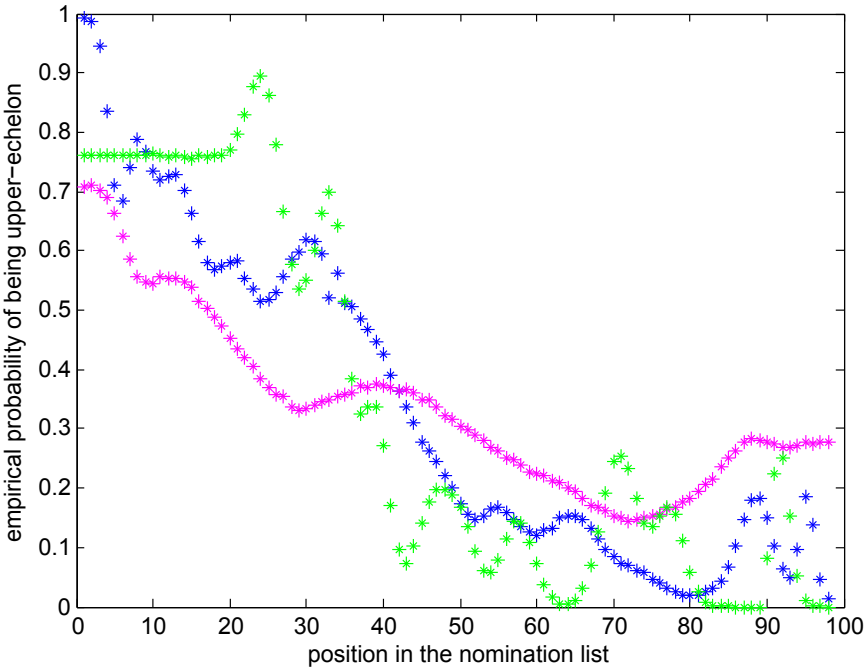

Figure 4.10: VN via SGM and VN via SP for Enron data.

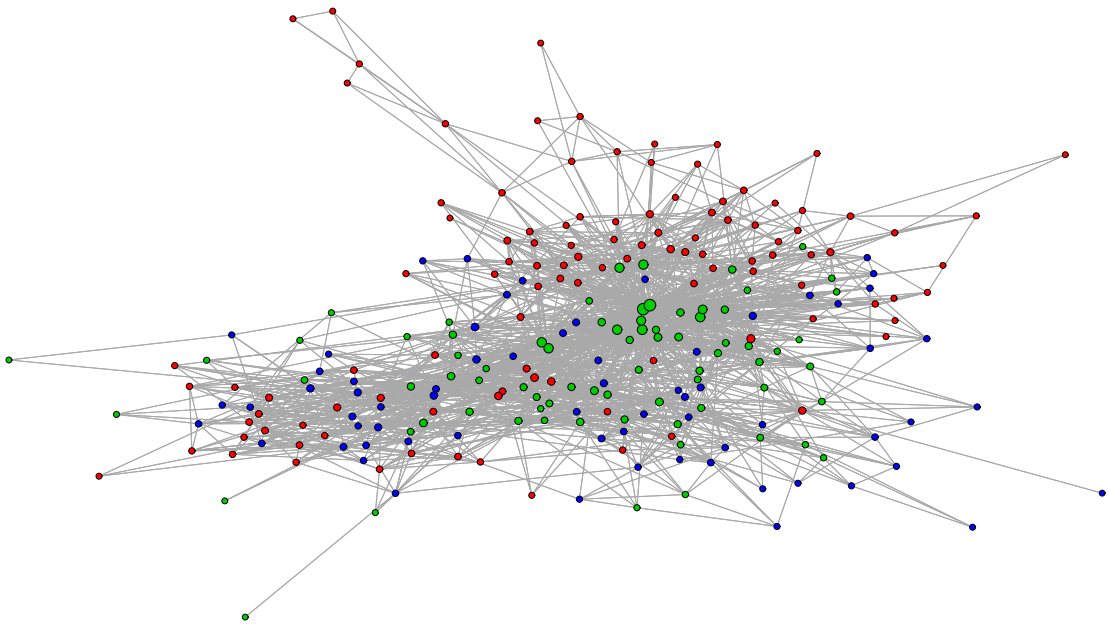

Figure 4.11: Visualization of the *C.elegans* graph data.

in soil. There are two networks, one is the chemical synapses and the other is the electrical synapses with 279 vertices. In each network the neurons are divided in to 67 sensory, 76 inter-neurons, and 110 motor neurons. We focus on the chemical network and nominate the motor neurons. We use 60 vertices as seeds, 30 motor neurons, 20 inter-neurons, and 10 sensory neurons, chosen uniformly at random for each of the 1000 simulates.

In Figure 4.12 MH and ML nomination schemes have almost identical performance, with MH slightly outperforming ML. The MMB nomination scheme performs similarly to spectral, but is much more stable.

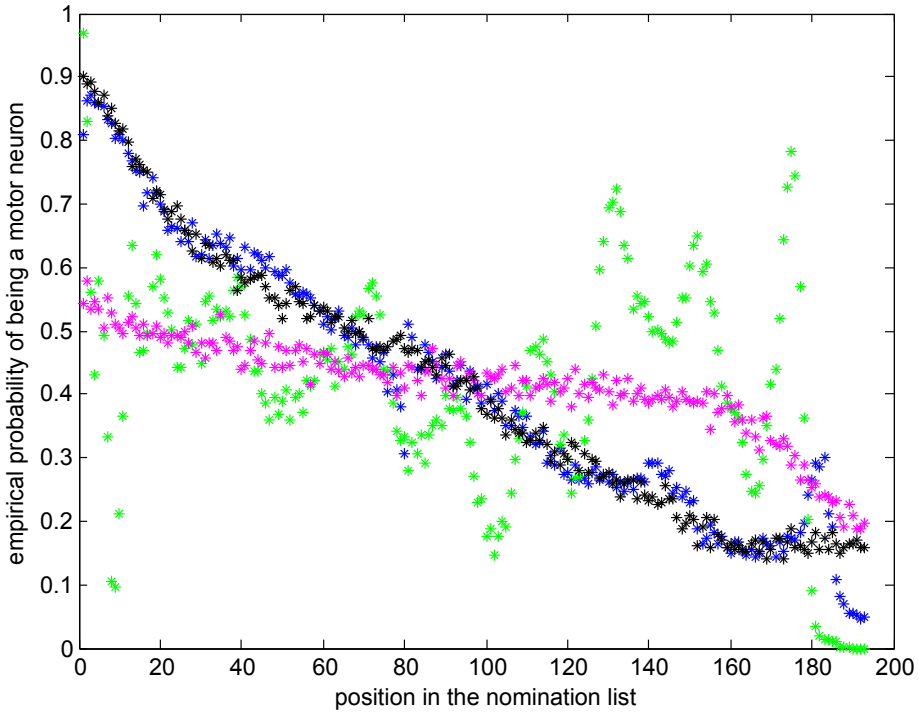

Figure 4.12: We compare three methods of vertex nomination: SP (green), ML (blue), MH (black), and MMB (purple). For this dataset, ML performs the best, MMB performs the second best, and SP performs like chance.

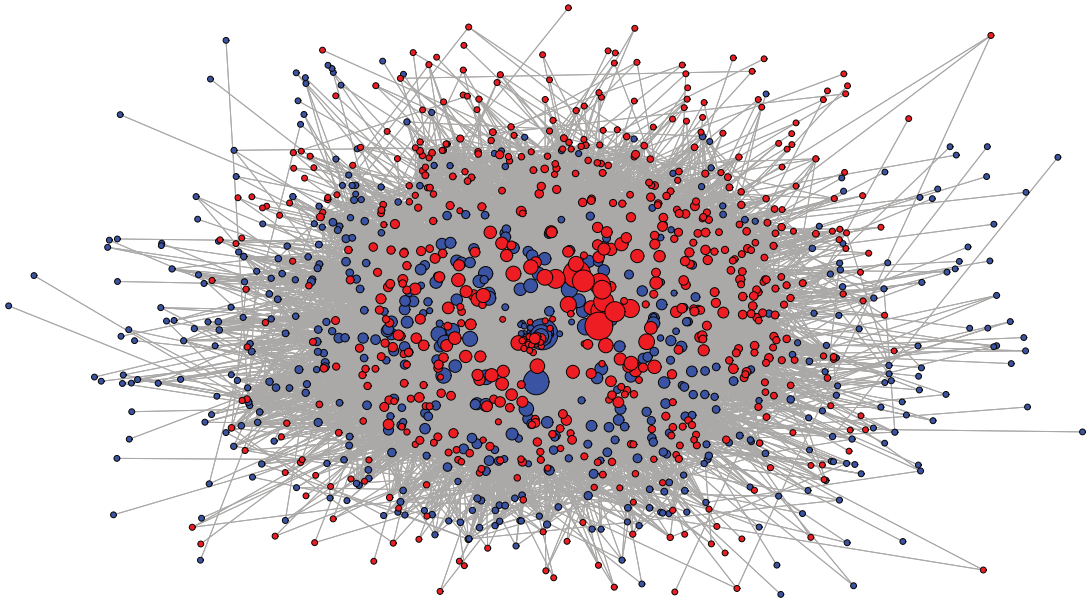

Figure 4.13: Visualization of the political blog graph data.

4.8.2.3 Political Blog Data

This dataset is created from web-logs during the 2004 US presidential election [90]. The blogs are the vertices of the graphs. There is an edge between two blogs if one blog links to the other blog. Each of the 1490 blogs are considered to be either conservative or liberal. There are 636 conservative blogs and 588 liberal blogs. After removing isolated vertices there are 1224 blogs remaining. We binarized and symmetrized the data before applying our algorithms. In this experiment we chose 80 seeds from each political orientation. The experiment is repeated 1000 times.

The blog dataset is well separated, and all except MMB are able to perform well. MH performs the best, followed by ML, then spectral and finally MMB.

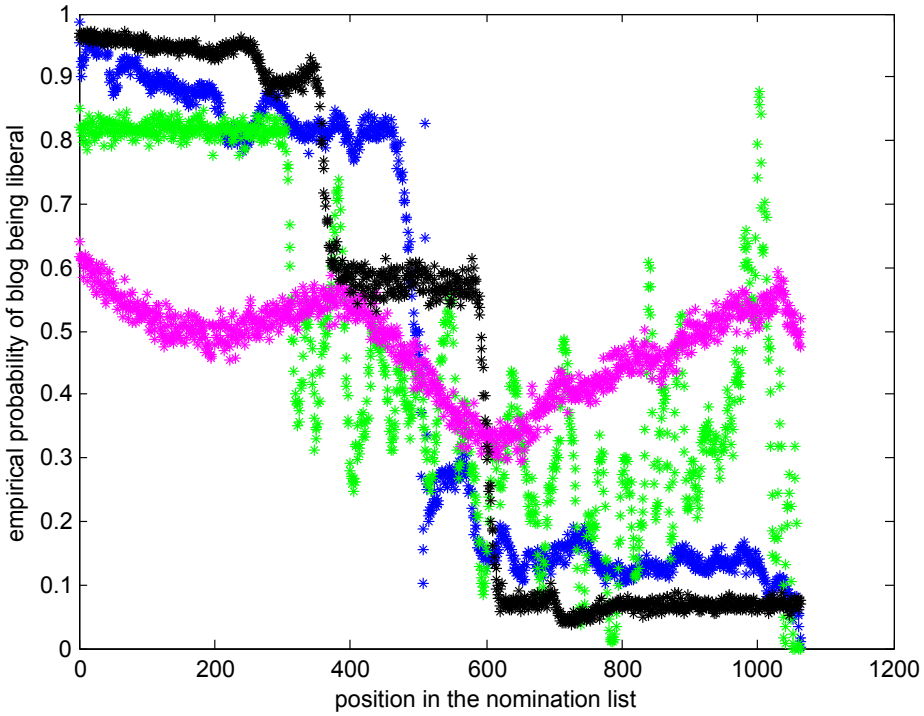

Figure 4.14: We compare four methods of vertex nomination: SP (green), ML (blue), MH (black), and MMB (purple). For this dataset, MH performs the best, ML performs second best, spectral performs third best, and MMB performs like chance.

4.8.2.4 Movie Dataset

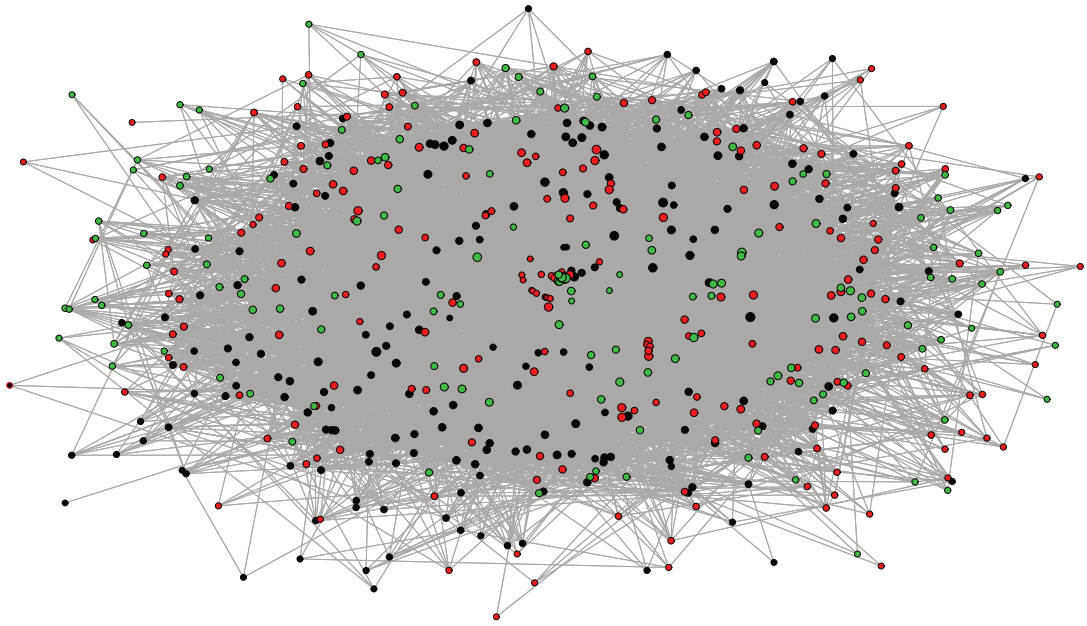

Figure 4.15: Visualization of the movie graph data.

The Movie data set was created by scrapping movie info-boxes from Wikipedia. All movies from 5 movie studios (20^{th} Century Fox, Columbia Pictures, Paramount, Universal, and Warner Bros.) from 2000 to 2010 were collected. The directors, producers, and actors (from the info-box) were saved along with categories (such as movie genre). Each vertex in the movie graph is a movie from one of the 5 studios between 2000 to 2010 and belongs to exactly one category out of comedy, action thriller, and drama. Two movies have an edge between them if they have at least a director, producer, or actor in common. There are 227 comedies, 157 action thriller, and 235 dramas movies. This makes the total number of vertices 619. This experiment

is also repeated 1000 times.

We nominate the movies for being a comedy. A total of 90 vertices, 30 from each category, were chosen as seeds. The MAP of the ML nomination scheme is 0.5814, of the spectral nomination scheme is 0.3500, and of the MMB nomination scheme 0.3766. Spectral and MMB nomination schemes performed approximately the same as chance.

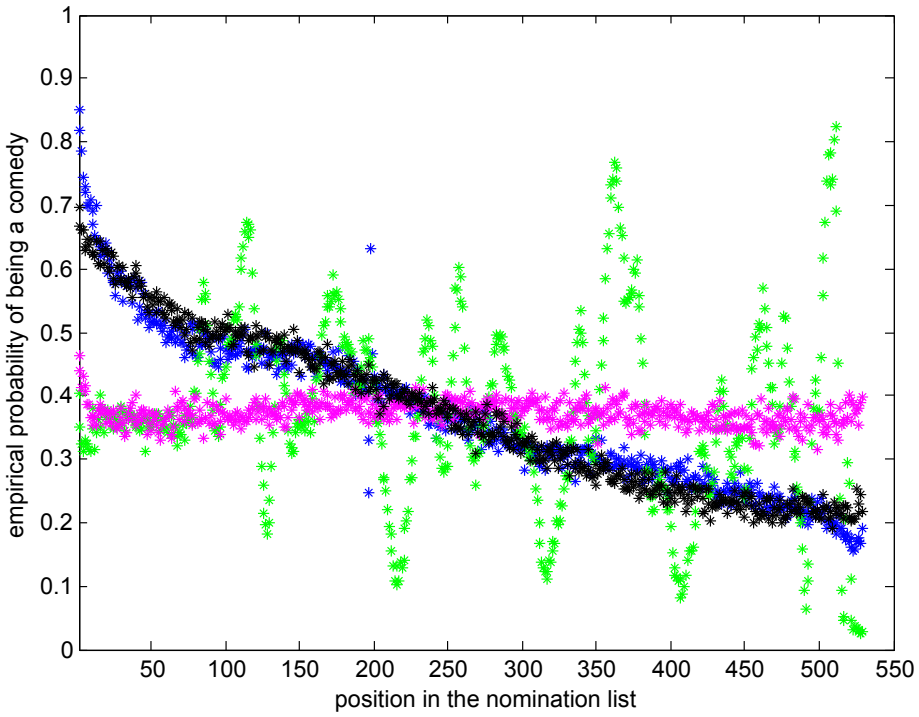

Figure 4.16: We compare four methods of vertex nomination: ML (blue), MH (black), SP (green), and MMB (purple). For this dataset, ML performs the best.

Method	ML	MH	SP	MMB	Chance
Enron	0.7779		0.7619	0.5970	0.3367
C.elegans	0.7272	0.7379	0.5096	0.5041	0.4145
Blog	0.8922	0.9317	0.7856	0.5429	0.4774
Movie	0.5814	0.5707	0.3764	0.3766	0.3724

Table 4.4: MAP values for ML, MH, Spectral, MMB, and chance vertex nomination schemes.

4.8.2.5 Discussion

Areas of the graphs where the accuracy appear to be flat are an artifact of ties in the MMB nomination scheme. The vertices are ranked by their posterior probabilities in the MMB scheme. When the vertices all have the same posterior probabilities, all of the vertices appear in the nearly the same positions. We randomize the ordering in each simulation, which smooths out the ties into a plateau in the plot.

Note that the spectral and mixed membership stochastic block model do not fully utilize the seed vertices. For example, consider the spectral nomination scheme. The seed vertices are only utilized after the spectral partitioning. Spectral partitioning is computed on all vertices with no special care given to seed vertices. Regardless of how many times we re-sample the seed vertices, the embedding and clustering do not change. The seed vertices are only used to determine $\widehat{V}_1$.

Therefore in the real data experiments, there is only embedding and clustering of seed and non-seed vertices for all experiments. If we pick the same cluster as

$\widehat{V}_1$ for all experiments, then there is only one nomination ordering, and we are only remove different subsets of seed vertices for each experiment. This causes of the sinusoidal artifact in the plots. Another cause of the sinusoidal artifact is that spectral partitioning may be finding sub-communities in the data. The sub communities may may be more or less likely to be from one block or another.

Chapter 5

Large Seeded Graph Matching

There exist many graph matching algorithms (see Section 2.4), each having advantages and disadvantages; some are fast but have low accuracy e.g. QCV and others are slower but have higher accuracy, e.g. GLAG. With current computational resources, the state-of-the-art approximate seeded graph matching algorithms are practically limited to matching graphs on the order of 1000 vertices. To bijectively match graphs of larger orders of magnitude, a new algorithm must be developed. Our large seeded graph matching algorithm first divides the graph into small subgraphs, which subsequent graph matching algorithms can more easily process. The work of this section is submitted as [91].

There have been others who have employed divide-and-conquer techniques to graph matching [4, 92, 93]. Our algorithm uses spectral partitioning to divide the graph into subgraphs. Our algorithm's theoretical basis and a priori use of seed

vertices distinguishes it from the current literature.

5.1 Algorithm

The Large Seeded Graph Matching (LSGM) algorithm is designed to be a scalable graph matching algorithm. In order to make the algorithm scalable, we use a divide-and-conquer approach, decomposing the large graph into smaller paired subgraphs such that each pair is then independently matchable. This method is flexible and does not require that we use any specific graph matching algorithm to match the subgraphs.

The key to the mechanics of this algorithm is Lemma 2.2.1, which bounds the error in the estimated latent position (suitably rotated) for random graphs distributed as $\kappa(b, \Lambda)$. From this, we show Lemma 5.2.1, which states that any two ρ-correlated $\kappa(b, \Lambda)$ graphs (defined in Section 1.4.7) have adjacency spectral embeddings (ASE) that approximately differ by an orthogonal transformation. However, finding this orthogonal transformation via Procrustes, requires us to know the complete alignment of the two graphs. We know the correspondence of the seed vertices. We find the Procrustes transformation on the seed vertices and hope to align the full graphs as using the resulting transformation. Seed vertices are optional in graph matching, but are necessary for LSGM. Having aligned the two graphs, we next divide the vertices into smaller subgraphs, which state-of-the-art graph matching algorithms are able to

handle. As each of these matchings is treated as a separate graph matching problem, this makes the algorithm fully parallelizable after dividing the vertices.

Algorithm 5.9 Large Seeded Graph Matching

1: $\mathcal{V}_A\Sigma_A\mathcal{V}_A^T \leftarrow A$, $\mathcal{V}_B\Sigma_B\mathcal{V}_B \leftarrow B$ ▷ Compute ASE

$\widehat{X}' \leftarrow \widehat{\mathcal{V}}_A\widehat{\Sigma}_A^{\frac{1}{2}}$, $\widehat{Y} \leftarrow \widehat{\mathcal{V}}_B\widehat{\Sigma}_B^{\frac{1}{2}}$

2: $Q \leftarrow \arg\min_{R'} \|\widehat{X}'_s R' - \widehat{Y}_s\|_F$ ▷ Solve Procrustes for Seed Vertices

$\widehat{X} \leftarrow \widehat{X}'Q$ ▷ Align ASE of the Full Graphs

3: $C \leftarrow \arg\min_C \|C - [\widehat{X}, \widehat{Y}]\|_F$, where C has K distinct rows ▷ Cluster $[\widehat{X}, \widehat{Y}]$

4: Partition $\widehat{X}$ and $\widehat{Y}$ into evenly sized clusters ▷ Divide

5: Run GM in parallel on graphs $G^A[U^A \cup C_k^A]$ and $G^B[U^B \cup C_k^B]$ ▷ Conquer

6: Return combined SGM results

Let $G^A = (V^A, E^A)$ and $G^B = (V^B, E^B)$ be two graphs. We use the notation from Section 1.4.7 and 2.4. Let the associated adjacency matrices of G^A and G^B be A and B respectively. Partition the respective vertex sets V^A and V^B into seed U^A and U^B and non-seed vertex sets W^A and W^B.

Step 1: The initial step of the algorithm is to compute the adjacency spectral embedding (ASE) for both adjacency matrices A and B. Recall, to compute the ASE, we need to compute the eigenvalue decomposition $A = \mathcal{U}_A\Sigma_A\mathcal{V}_A^T$ and $B = \mathcal{U}_B\Sigma_B\mathcal{V}_B^T$. Then the ASE of A and B are $\widehat{X}^A = \widehat{\mathcal{U}}_A\widehat{\Sigma}_A^{\frac{1}{2}}$ and $\widehat{X}^B = \widehat{\mathcal{U}}_B\widehat{\Sigma}_B^{\frac{1}{2}}$ respectively, where $\widehat{\mathcal{U}}_A$ and $\widehat{\mathcal{U}}_B$ are the first $d = \text{rank}(\Lambda)$ columns of $\mathcal{U}_A$ and $\mathcal{U}_B$ respectively and $\widehat{\Sigma}_A^{\frac{1}{2}}$ and $\widehat{\Sigma}_B^{\frac{1}{2}}$ are the (element-wise) square root of the leading $d \times d$ principle sub-matrices of

Σ_A and Σ_B respectively. Without loss of generality, let $\widehat{X}_s^A$ and $\widehat{X}_s^B$ be the first s rows of $\widehat{X}^A$, $\widehat{X}^B$, and $\widehat{X}_{\bar{s}}^A, \widehat{X}_{\bar{s}}^B$ be the last m rows of $\widehat{X}^A$ and $\widehat{X}^B$ respectively, i.e. the spectral embedding of the seed and non-seed vertices. Thus, $\widehat{X}^A = [\widehat{X}_s^A; \widehat{X}_{\bar{s}}^A]$ and $\widehat{X}^B = [\widehat{X}_s^B; \widehat{X}_{\bar{s}}^B]$.

Step 2: The next step is to use only the embedded seed vertices $\widehat{X}_s^A$ and $\widehat{X}_s^B$ to align the two graphs via a Procrustes transformation [94]. The orthogonal Procrustes problem with scaling is to find

$$\widehat{Q} = \underset{R \in R(d)}{\arg\min} \|\widehat{X}_s^A R - \widehat{X}_s^B\|_F,$$

where

$$R(d) = \left\{ \begin{array}{rcl} R \in \mathbb{R}^{d \times d} & : & R = S_1 \Omega S_2 \text{ for diagonal } S_1, S_2 \in \mathbb{R}^{d \times d} \\ & & \text{and } \Omega \in \mathbb{R}^{d \times d} \text{ s.t. } R^T R = I \end{array} \right\}.$$

The role of $\widehat{Q}$ is to be a surrogate for the transformation to align the ASE of the full graphs.

Step 3: With the latent positions aligned, we can cluster all $2n$ vertices $\widehat{X}^{A,B} = [\widehat{X}^A \widehat{Q}; \widehat{X}^B] \in \mathbb{R}^{2n \times d}$ simultaneously. Ideally, we want to compute

$$\widehat{C} := \underset{C \in \mathbb{R}^{2n \times d}}{\arg\min} \sum_{i=1}^{2n} \left\| \begin{bmatrix} \widehat{X}^A Q \\ \widehat{X}^B \end{bmatrix}_i - C \right\|, \tag{5.1}$$

where C is a matrix with K distinct rows. We use k-means to approximately cluster, because of its wide availability and theoretical tractability. In other words, C is a matrix containing the centroids of each vertex. We choose K clusters, because we

have K blocks.

Step 4: Ideally, each cluster would contain the same number of vertices from each graph [60]. However, this usually does not happen. Let c_k^A, c_k^B be the number of vertices from cluster k from graph A and B respectively, with $c_1^A + c_1^B \geq c_2^A + c_2^B \geq \ldots \geq c_K^A + c_K^B$. For all $k \in \{1, 2, 3, \ldots K\}$ clusters, we create two corresponding subgraphs $G_k^A \subset G^A$ and $G_k^B \subset G^B$. We set the order of subgraphs G_k^A and G_k^B to be

$$\widehat{m}_k := \left\lceil \frac{c_k^A + c_k^B}{2} \right\rceil - \mathbb{1}\left[\sum_{i=1}^{K} \left\lceil \frac{c_i^A + c_i^B}{2} \right\rceil \geq k + n \right].$$

The first term averages the sizes of cluster k from both graphs and rounds up (ceiling function). The ceiling function is defined on real numbers $x \in \mathbb{R}$ as

$$\lceil x \rceil = \min\{n \in \mathbb{Z} : n \geq x\}.$$

Note that the sum of the first terms might sum to a value greater than n. To correct this, the second term reduces the largest clusters by one, until sum is exactly $\sum_k \widehat{m}_k = n$. Having determined the order of each subgraph, the next step is to assign vertices to subgraphs. Initially, all vertices start unassigned. For $k = 1, 2, \ldots, K$, the subgraphs are formed by assigning unassigned vertices sequentially. To create the k^{th}-subgraph vertex sets, the $\widehat{m}_k$ closest unassigned vertices of G^A and G^B are assigned to G_k^A and G_k^B respectively. Let $\widehat{W}_k^A$ and $\widehat{W}_K^B$ denote the non-seed vertices in subgraphs G_k^A and G_k^B respectively, with sizes $\widehat{m}_k = |\widehat{W}_k^A| = |\widehat{W}_k^B|$.

Step 5: Recall that the s seeded vertices are $U = (v_1, v_2, \ldots v_s)$. The next step of the algorithm is to graph match the K clusters in parallel. For clusters which are

small enough to match efficiently, we directly apply the graph matching algorithm. For all clusters which are too large to be matched, LSGM is applied (recursively) to further divide these clusters.

In the matching subroutines, we could also use seeded graph matching instead of graph matching. If including all of the seed vertices is not too large for the graph matching algorithm, we would use all of the seed vertices, i.e. we match the subgraphs $G^A[U^A \cup \widehat{W}_k^A]$ and $G^B[U^B \cup \widehat{W}_k^B]$. Let the output permutation matrix excluding seed vertices for matching $G^A[U^A \cup \widehat{W}_k^A]$ and $G^B[U^B \cup \widehat{W}_k^B]$ be the $\widehat{P}_k \in \mathbb{R}^{\widehat{m}_k \times \widehat{m}_k}$. Using more seed vertices improves performance of our graph matching algorithm, SGM. Again this step can be done in parallel, as we can match these graphs separately. If $|U^A \cup \widehat{W}_k^A|$ has too many vertices for the seeded graph matching algorithm, we select a subset of vertices as seeds for the graph matching. We attempt to select the seeds which provide the most information, see Section 5.1.1 for details.

Step 6: After matching all of the subgraphs separately, the matchings for each subgraph are combined together into one bijection. Call $\widehat{P}_k$ the permutation from the graph matching subgraph k. Once we have the permutations $\widehat{P}_k$, we combine the permutations into

$$\widehat{P} = \oplus_{k \in \{1,2,3,\ldots K\}} \widehat{P}^{(k)},$$

where $\oplus$ is the matrix direct sum. This provides the matching for all the vertices across the entire graph.

5.1.1 Selective Seeding

For extremely large graphs, it may not be computationally possible to include all seeds for each SGM computation. Thus, seeds must be selectively chosen. We expect optimal performance in SGM is achieved by selecting seed vertices with the maximum mutual information between the seeds and non-seeds in the subgraph [61]. We use a greedy selection process to approximate the optimal subset of seeds. We iteratively select seeds with maximum added entropy. Entropy is a measure of uncertainty of a random variable in bits and is defined as [95]

$$H(X) = E[-\log_2(P[X])].$$

Suppose we want to select a total of $\widehat{s}_k$ seeds for the k^{th} subgraph, and the non-seed vertices of the k^{th} subgraph G_k^A and G_k^B are $\widehat{W}_k^A$ and $\widehat{W}_k^B$ respectively. First we initialize the seed vertices for graphs G_k^A and G_k^B to the empty set, $\widehat{U}_k^A = \varnothing$ and $\widehat{U}_k^B = \varnothing$ respectively, then we iteratively select seed vertices $v_{i_1}^A, v_{i_2}^A, \ldots, v_{i_{\widehat{s}_k}}^A$ and corresponding $v_{i_1}^B, v_{i_2}^B, \ldots, v_{i_{\widehat{s}_k}}^B$ via

$$i_t \in \underset{j \neq \{i_1, i_2, \ldots, i_{t-1}\}}{\arg\max} H(\widehat{\mathcal{F}}(G^A[v_j^A \cup \widehat{U}_k^A \cup \widehat{W}_k^A])) + H(\widehat{\mathcal{F}}(G^B[v_j^B \cup \widehat{U}_k^B \cup \widehat{W}_k^B])),$$

where $\widehat{\mathcal{F}}(\cdot)$ is the empirical cumulative distribution function of the columns of the seed-to-non-seed adjacency matrix of the subgraph. At each iteration, this procedure selects the seed vertex that increases the empirical entropy the most.

5.1.2 Computational Efficiency

In this algorithm, there are a few computationally intensive steps: the spectral embedding, Procrustes rotation, and graph matching. For spectral embedding we only use the first d eigenvectors and eigenvalues. For large graphs ideally $d << n$, when $d \leq \sqrt{n}$, then this step has complexity $O(n^2 d)$ [96]. Solving the orthogonal Procrustes problem has complexity at most $O(nd^2)$. Finally SGM has a runtime of $O(n^3)$ for a graph of size n, see Section 2.4.2. In LSGM, SGM is not computed on the whole graph, as SGM is applied to the K clusters directly. Let $c_{\max}$ be the maximum size of all the clusters, $c_{\max} = \max_k c_k$, then SGM's runtime is $O((c_{\max} + s)^3)$.

Suppose there exists an $\alpha > 0$, such that $s = o(n^{1-\alpha})$, $K = \Omega(n^\alpha)$, and $c_{\max} = O(n^{1-\alpha})$. When the independent SGM routines for each cluster are not parallelized, the graph matchings would have complexity

$$O(K(c_{\max} + s)^3) = O(n^\alpha (n^{1-\alpha} + n^{1-\alpha})^3) = O(n^\alpha n^{3(1-\alpha)}) = O(n^{3-2\alpha}).$$

If $\alpha \geq 1/2$ then the complexity becomes $O(n^2)$. With $\alpha \approx 1/2$, the serial LSGM algorithm has complexity $O(n^2 d + nd^2 + n^2) = O(n^2 d)$.

If we assume that the computer resource has $O(K)$ cores such that the SGM subroutines are fully parallelized, then the complexity is $O((c_{\max} + s)^3)$, where s is the number of seeds. Note that if $c_{\max} = \Theta(n)$, then parallelized SGM would have complexity $O(n^3)$ and LSGM is as computationally intensive as SGM. However, if c_k is too large, LSGM is again applied on cluster k, recursively partitioning the cluster.

If $c_{\max} = O(n^{1-\alpha})$, the complexity of the SGM steps is

$$O((c_{\max} + s)^3) = O((n^{1-\alpha} + n^{1-\alpha})^3) = O(n^{3(1-\alpha)}).$$

For $\alpha \geq 1/3$ the complexity of the SGM step is $O(n^2)$. If $\alpha \approx 1/3$ then the parallelized LSGM algorithm has complexity $O(n^2 d + nd^2 + n^2) = O(n^2 d)$, which is a large improvement over $O(n^3)$. It is interesting and important to note that parallelizing the algorithm is not necessary to have the improved runtime.

5.2 Theoretical Results

In this section, we will prove consistency results for the LSGM algorithm. First, we restate some convergence results. Next, we show that with some natural conditions, the ASE embeddings of A and B almost always approximately differ by an orthogonal rotation. Finally, we show that we can approximate this rotation by using the seed vertices and hence LSGM almost always returns the true alignment function.

Let G^A and G^B be ρ-correlated $\kappa(S, M, \Lambda)$ graphs. Let A and B be their respective adjacency matrices. Without loss of generality, let the true alignment function be the identity mapping, thus the block membership function is $b = b^A = b^B$.

Theorem 5.2.1

If the following hold:

1. *There exist constants* $\epsilon_1, \epsilon_2 > 0$ *such that* $K = O(n^{1/3-\epsilon_1})$ *and* $\min_i n_i = \Omega(n^{2/3+\epsilon_2})$

2. *Define*

$$\beta = \beta(n, d, \delta_d) = \frac{260d \log n}{\delta_d n^{1/2}},$$

if $i, j \in \{1, 2, 3, \dots n\}$ *are such that* $x_i \neq x_j$ *then* $\|x_i - x_j\|_2 > 6n^{1/6}\beta$.

3. *Without loss of generality, let* $\{x_i\}_{i=1}^s$ *be the latent positions corresponding to the seed vertices. Assume there exists an* α *satisfying* $\alpha > 4\beta$ *and* $\sqrt{n}\beta/\alpha = o(n^{\epsilon/2})$ *such that*

$$\min_{v:\|V\|_2=1} \|X_s v^T\| \geq \alpha\sqrt{s}.$$

Then for all but finitely many n, *the* $\widehat{b}$ *of satisfies* $\widehat{b} = b$.

Given the above assumptions, for all but finitely many n, the $\widehat{b}$ of equation 5.1 satisfies $\widehat{b} = b$ and for all $k \in \{1, 2, 3, \dots K\}$, $\widehat{P}^{(k)} = \{I_{\widehat{m}_k}\}$. Therefore $\widehat{P} = \oplus_{k=1}^K \widehat{P}^{(k)} = I_n$ is equal to the identity matrix and $\widehat{P}$ is the true alignment function. This theorem states that given some mild assumptions for the sequence of graphs $G_n \sim \kappa(n, \pi, \Lambda)$ the LSGM algorithm is able to find the true permutation almost always.

The following lemma is a version of Lemma 2.2.1, for two graphs with the same latent positions, i.e. Λ.

Lemma 5.2.2 *Sussman et al. [19] and Lyzinski et al. [47]*

With previous notation, let

$$\delta_d := \min_{i,j \leq d+1, i \neq j} \frac{|\sigma_i(XX^T) - \sigma_j(XX^T)|}{n},$$

where $\sigma_i(\cdot)$ means the i^{th} largest eigenvalue in magnitude. Let

$$R_A = \underset{R \in R(d)}{\arg\min} \|\widehat{X}^A - XR\|_F, \quad R_B = \underset{R \in R_(d)}{\arg\min} \|\widehat{X}^B - XQ\|_F.$$

If $d = o(\sqrt{n})$ then it holds that for all but finitely many n,

$$\|\widehat{X}^A - XR_A\|_{2,\infty} \leq \beta, \quad \|\widehat{X}^B - XR_B\|_{2,\infty} \leq \beta. \tag{5.2}$$

Now, we want to show that instead of two rotations R_A and R_B relating $\widehat{X}^A$ and $\widehat{X}^B$ to X, there exists one rotation between $\widehat{X}^A$ and $\widehat{X}^B$.

Lemma 5.2.3

With assumptions as in Theorem 5.2.1. For all but finitely many n it holds that

$$\|\widehat{X}^A Q - \widehat{X}^B\|_{2,\infty} \leq 8\beta/\alpha + 2\beta,$$

where

$$Q = \underset{R \in R(d)}{\arg\min} \|\widehat{X}_s^A R - \widehat{X}_s^B\|_F$$

Proof Let

$$\widetilde{Q} \;=\; R_A^T R_B.$$

It follows from lemma 5.2.2 that $\|\widehat{X}^A\widetilde{Q} - \widehat{X}^B\|_{2,\infty} \leq 2\beta$. Thus

$$\begin{aligned}
\|\widehat{X}^A_s Q - \widehat{X}^B_s\|_F &\leq \|\widehat{X}^A_s\widetilde{Q} - \widehat{X}^B_s\|_F = \sqrt{\sum_s \|\widehat{X}^A_s\widetilde{Q} - \widehat{X}^B_s\|_2} \\
&\leq \sqrt{\sum_s \|\widehat{X}^A_s\widetilde{Q} - \widehat{X}^B_s\|_\infty} = 2\beta\sqrt{s} \\
\Rightarrow 2\beta\sqrt{s} &\geq \|\widehat{X}^A_s Q - \widehat{X}^B_s\|_F \\
&= \|\widehat{X}^A_s Q - \widehat{X}^A_s\widetilde{Q} + \widehat{X}^A_s\widetilde{Q} - \widehat{X}^B_s\|_F \\
&\geq \|\widehat{X}^A_s(Q - \widetilde{Q})\|_F - \|\widehat{X}^A_s\widetilde{Q} - \widehat{X}^B_s\|_F \\
&\geq \|\widehat{X}^A_s(Q - \widetilde{Q})\|_F - 2\beta\sqrt{s}, \\
\Rightarrow 4\beta\sqrt{s} &\geq \|\widehat{X}^A_s(Q - \widetilde{Q})\|_F \\
\Rightarrow \|\widehat{X}^A_s(Q - \widetilde{Q})\|_F &\leq 4\beta\sqrt{s}.
\end{aligned}$$

Using the assumption

$$\min_{v:\|v\|_2=1} \|X_s v^T\|_2^2 \geq \alpha^2 s,$$

and the SVD decomposition of $Q - \widetilde{Q} = \mathcal{V}_Q \Sigma_Q \mathcal{U}_Q^T$, we have

$$\begin{aligned}
\|\widehat{X}^A_s(Q - \widetilde{Q})\|_F &= \|X^A_s R_A(Q - \widetilde{Q}) - X^A_s R_A(Q - \widetilde{Q}) + \widehat{X}^A_s(Q - \widetilde{Q})\|_F \\
&\geq \|X_s R_A(Q - \widetilde{Q})\|_F - \|(\widehat{X}^A_s - X_s R_A)(Q - \widetilde{Q})\|_F \\
&\geq \sqrt{\sum_{i=1}^{s}\sum_{j=1}^{d} \langle X_i, R_A\mathcal{V}_j\rangle \Sigma^2_{j,j}} - 2\beta\sqrt{s}\|Q - \widetilde{Q}\|_F \\
&\geq (\alpha - 2\beta)\sqrt{s}\|Q - \widetilde{Q}\|_F,
\end{aligned}$$

Hence

$$\begin{aligned}(\alpha - 2\beta)\sqrt{s}\|Q - \widetilde{Q}\|_F &\leq \|\widehat{X}_s^A(Q - \widetilde{Q})\|_F \leq 4\beta\sqrt{s} \\ \|Q - \widetilde{Q}\|_{2,\infty} &\leq \|Q - \widetilde{Q}\|_F \leq \frac{4\beta}{\alpha - 2\beta}.\end{aligned}$$

Since $\|\widehat{X}^A\|_{2,\infty} \leq 1$ and $\alpha > 4\beta$, we have

$$\begin{aligned}\|\widehat{X}^A Q - \widehat{X}^B\|_{2,\infty} &= \|\widehat{X}^A(Q - \widetilde{Q}) + \widehat{X}^A\widetilde{Q} - \widehat{X}^B\|_{2,\infty} \\ &\leq \|\widehat{X}^A(Q - \widetilde{Q})\|_{2,\infty} + \|\widehat{X}^A\widetilde{Q} - \widehat{X}^B\|_{2,\infty} \\ &\leq \|\widehat{X}^A\|_{2,\infty}\frac{4\beta}{\alpha - 2\beta} + 2\beta \\ &\leq \frac{8\beta}{\alpha} + 2\beta.\end{aligned}$$

∎

Lemma 5.2.4

For all but finitely many n, it holds that

$$\left\| \begin{bmatrix} \widehat{X}^B \\ \widehat{X}^A Q \end{bmatrix} - \begin{bmatrix} XR_B \\ XR_B \end{bmatrix} \right\|_{2,\infty} \leq \frac{8\beta}{\alpha} + 3\beta$$

Proof

$$\left\| \begin{bmatrix} \widehat{X}^B \\ \widehat{X}^A Q \end{bmatrix} - \begin{bmatrix} XR_B \\ XR_B \end{bmatrix} \right\|_{2,\infty} = \max\{\|\widehat{X}^B - XR_B\|_{2,\infty}, \|\widehat{X}^A Q - XR_B\|_{2,\infty}\}.$$

Recall from equation 5.2 that the first term $\|\widehat{X}^B - XR_B\|_{2,\infty}$ is bounded by β. The second term

$$\|\widehat{X}^A Q - XR_B\|_{2,\infty} \leq \|\widehat{X}^A Q - \widehat{X}^B\|_{2,\infty} + \|\widehat{X}^B - XR_B\|_{2,\infty} \leq \frac{8\beta}{\alpha} + 3\beta.$$

With these Lemmas we are ready to prove Theorem 5.2.1.

Proof Let $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_K$ be the ℓ_2-balls of radius $n^{1/6}\beta$ around the K rows of XR_B. If $x_i \neq x_j$, then by assumption in part 2,

$$6r = 6n^{1/6}\beta < \|x_i - x_j\|_2 = \|(x_i - x_j)R_B\|_2,$$

and $\mathcal{N}_i$'s are all disjoint.

Let $\widehat{X}^{A,B} = [\widehat{X}^A Q; \widehat{X}^B] \in \mathbb{R}^{2n\times d}$ and $X^{A,B} = [X^A R_B; X^A R_B] \in \mathbb{R}^{2n\times d}$. Let $(\widehat{C}, \widehat{b})$ be the optimal clustering solution of clustering $\widehat{X}^{A,B}$. Suppose there exists an index $i \in \{1, 2, 3, \ldots 2n\}$ such that $\|X_i^{A,B} - \widehat{C}_i\| > 2r$. Lemma 5.2.2 states that for a sequence of graphs G_n almost always $\|\widehat{X}^A - XR_A\|_{2,\infty} \leq \beta$ and $\|\widehat{X}^B - XR_B\|_{2,\infty} \leq \beta$. Thus, by the triangle inequality

$$\|\widehat{X}_i^{A,B} - \widehat{C}_i\| \geq \|X_i^{A,B} - \widehat{C}_i\| - \|\widehat{X}_i^{A,B} - X_i^{A,B}\| \geq 2r - \beta.$$

Then

$$\|\widehat{X}^{A,B} - \widehat{C}\|_F > \sqrt{\min_j n_j}(2r - \beta).$$

Recall assumption 1 is $\min_j n_j = \Omega(n^{2/3+\epsilon_2})$ for a constant $\epsilon_2 > 0$. This means that

$$\begin{aligned}\|\widehat{X}^{A,B} - \widehat{C}\|_F &> \sqrt{\min_j n_j}(2r - \beta) = \Omega(n^{2/6+\epsilon_2/2})(2n^{1/6}\beta - \beta) = \Omega(n^{2/6+\epsilon_2/2})(2n^{1/6} - 1)\beta \\ &= \Omega\left(n^{2/6+\epsilon_2/2} n^{1/6} \frac{\log n}{n^{1/2}}\right) = \Omega(n^{\epsilon_2/2}).\end{aligned}$$

This contradicts Lemma 5.2.4. Therefore $\|X^{A,B} - \widehat{C}\|_{2,\infty} \leq 2r$. This with Equation (5.2), yields

$$\begin{aligned} \|X^{A,B} - \widehat{C}\|_{2,\infty} &= \|X^{A,B} - \widehat{X}^{A,B} + \widehat{X}^{A,B} - \widehat{C}\|_{2,\infty} \\ &\leq \|X^{A,B} - \widehat{X}^{A,B}\|_{2,\infty} + \|\widehat{X}^{A,B} - \widehat{C}\|_{2,\infty} \\ &\leq \beta + 2r = (2 + o(1))r. \end{aligned}$$

For $i, j \in \{1, 2, 3, \ldots 2n\}$ such that $\widehat{C}_i \neq \widehat{C}_j$, $\|X_i^{A,B} - X_j^{A,B}\| > 6r$ (by assumption 2). This implies

$$\begin{aligned} \|\widehat{X}_i^{A,B} - \widehat{C}_j\|_2 &\geq \|\widehat{X}_i^{A,B} - X_j^{A,B}\| - \|X_j^{A,B} - \widehat{C}_j\| \\ &\geq \|X_i^{A,B} - X_j^{A,B}\| - \|\widehat{X}_i^{A,B} - X_i^{A,B}\| - \|X_j^{A,B} - \widehat{C}_j\| \\ &\geq 6r - \beta - 2r = 4r - \beta = (4 + o(1))r. \end{aligned}$$

Therefore, for the sequence of graphs $G_n \sim \kappa(n, \pi, \Lambda)$ it almost always holds that $\widehat{b} = [b; b]$, i.e.

$$\min_{\varphi \in \Psi_K} |\{i \in \{1, 2, 3, \ldots 2n\} : b_i^{A,B} \neq \varphi(\widehat{b}_i)\}| = 0.$$

∎

5.3 Performance

To evaluate the performance of the algorithm, we apply LSGM on simulated data and real data. Since there are many parameters to LSGM, we examine some of the parameters with different experiments. The first section compares SGM to LSGM,

the next section examines different max cluster sizes, and the last section shows performance of LSGM over various values of M and ρ.

5.3.1 Simulated Data

To evaluate our algorithm, we designed several experiments to test and study LSGM. The simulations are ρ-correlated stochastic block models $\kappa(b, \Lambda)$. All of the experiments are paired, meaning that all graph matching methods use the same graphs and seed vertices.

5.3.1.1 Comparison with SGM

It is important to compare LSGM with SGM, to understand how much performance is lost. We are certainly sacrificing performance when using the LSGM algorithm for the following two main reasons. First when the clusters are created, if we incorrectly assign vertices to a cluster, then the incorrectly assigned vertices are guaranteed to be incorrect. Second, in the small matchings, vast amount of the adjacency matrix is discarded. All the off-diagonal sub-matrices are discarded, but they would have been used in SGM.

In order to run both LSGM and SGM, the size of the graph must be small enough for SGM to be feasible, and large enough such that LSGM is also feasible. The

parameters of the comparison are

$$\rho = 0.7, N + M = \begin{bmatrix} 200 \\ 200 \\ 200 \end{bmatrix}, \Lambda = \begin{bmatrix} 0.6 & 0.3 & 0.2 \\ 0.3 & 0.7 & 0.3 \\ 0.2 & 0.3 & 0.7 \end{bmatrix},$$

and the number of seed vertices tested are $m = 3, 4, 5, 6, 7$. The seeds are drawn uniformly from the 600 vertices.

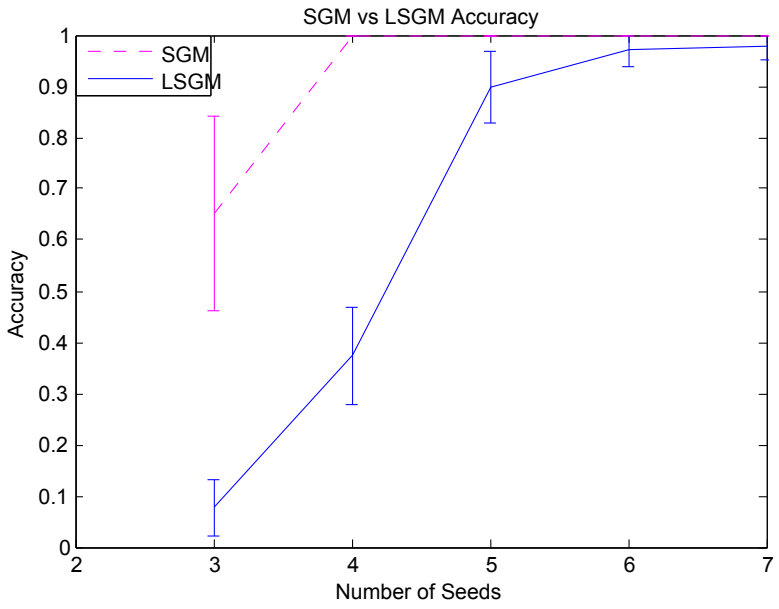

Figure 5.1: This plot compares the accuracy of SGM and LSGM, with 2 standard error bars.

From the figure we can see that SGM is able to perfectly recover the graph with 4 seeds, but LSGM needs 6 seeds to have comparable performance.

5.3.1.1.1 Comparison of Graph Matching Algorithms

In this section, we compare the graph matching algorithms in the standard graph matching setting, and we compare them in the LSGM setting. The algorithms we compare are described in Section 2.4.3. We use the abbreviations from that section as well. Let FAQ be the abbreviation for fast approximate quadratic assignment problem covered in Section 2.4.2, and let RAND be the method of uniformly at random selecting a permutation from Ψ_n.

In this first table presents the accuracy of graph matching algorithms for different sized graphs. The graphs are ρ-correlated, with $\rho = 0.9$. They are two block model with

$$\Lambda = \begin{bmatrix} .6 & .3 \\ .3 & .6 \end{bmatrix}.$$

Each block has an equal number of vertices. Each experiment is simulated 100 times.

From Table 5.1 GLAG performs the best in most cases. PATH also performs relatively well. QCV only does well for $n = 100$, and performs about the same as FAQ for the rest. RANK performs better than U, but both have very low accuracy.

Now we repeat the experiment: two block model, correlation $\rho = 0.9$, and 100 simulates. Except, we perform LSGM and use 3 randomly selected seed vertices. Since there are seeds, we can now also use SGM as a graph matching algorithm.

Notice in the LSGM accuracy Table 5.2 that even the random algorithm has increased in accuracy. Thus LSGM is finding relevant subgraphs. In fact the accuracy of

n	100	200	300	400
RAND	0.0075	0.0048	0.0020	0.0033
U	0.0395	0.0160	0.0083	0.0074
RANK	0.0515	0.0290	0.0167	0.0135
QCV	0.9665	0.1760	0.0178	0.0138
FAQ	0.1855	0.1180	0.0583	0.0143
PATH	1.0000	0.3152	0.3583	0.0632
GLAG	0.8250	0.5500	0.3672	0.2719

Table 5.1: GM accuracy for different sized graphs.

n	100	200	300	400
RAND	0.0252	0.0128	0.0086	0.0058
U	0.0983	0.0366	0.0219	0.0148
RANK	0.0888	0.0477	0.0322	0.0258
QCV	0.9743	0.9313	0.8414	0.3441
FAQ	0.2094	0.1145	0.0690	0.0268
PATH	0.9840	0.9451	0.9557	0.6067
GLAG	0.7440	0.5548	0.4006	0.3030
SGM	0.9740	0.9550	0.9587	0.9558

Table 5.2: LSGM accuracy for different sized graphs.

$n =$	100	200	300	400
RAND	0.15	0.27	0.49	0.80
U	0.17	0.36	0.77	1.99
Rank	0.17	0.31	0.59	1.08
QCV	0.96	1.03	0.80	1.59
FAQ	1.84	8.42	21.1	38.0
PATH	5.85	50.0	101	682
GLAG	13.4	125	375	824

Table 5.3: GM runtime for one graph at different sized graphs in seconds.

SGM is almost 1, thus LSGM is dividing the graph into paired subgraphs well. PATH and QCV greatly benefit from use of LSGM. The other graph matching algorithms do not perform significantly better or worse than on the complete graph.

5.3.1.1.2 Runtime

One large factor in considering, which algorithm to use for a specific application is runtime. Table 5.3 has the average runtimes for performing one graph matching in seconds.

This table shows that PATH and GLAG have significantly longer runtimes than all other algorithms. FAQ's runtime is in between in the middle.

Table 5.4 has the runtime for LSGM with the various graph matching algorithms.

$n =$	100	200	300	400
RAND	0.0981	0.1308	0.1794	0.2552
U	0.0993	0.1419	0.2131	0.3470
RANK	0.0974	0.1377	0.1953	0.3002
QCV	0.3071	1.2485	2.9638	3.4646
FAQ	0.5194	3.1227	9.1300	16.6761
PATH	2.2159	9.9000	15.8238	69.3173
GLAG	8.5387	33.8370	109.4802	261.7222
SGM	0.1460	0.7852	2.1338	4.4985

Table 5.4: LSGM runtime for one graph at different sized graphs in seconds.

The runtimes for LSGM are mostly smaller than in graph matching, except for QCV. Note that SGM is much lower runtime than FAQ, and higher accuracy. Using seed vertices in the other graph matching algorithms should be studied. However, this requires more intimate knowledge of all algorithms and is beyond the scope of our study.

5.3.1.2 Different cluster sizes

Now that we have compared graph matching with large seeded graph matching algorithm, we examine the effect that the maximum cluster size has on accuracy. In these simulations, we use ρ-correlated stochastic block models with 10 blocks with

max cluster size	100	200	300	400	500
RAND	0.0090	0.0041	0.0034	0.0029	0.0013
U	0.0099	0.0043	0.0039	0.0028	0.0021
Rank	0.0135	0.0086	0.0050	0.0043	0.0036
QCV	0.0163	0.0127	0.0081	0.0063	0.0038
FAQ	0.0097	0.0057	0.0043	0.0042	0.0026
PATH	0.0138	0.0101	0.0122	0.0089	0.0029
GLAG	0.0147	0.0103	0.0088	0.0065	0.0067
SGM	0.4382	0.4187	0.5057	0.4951	0.5761
Oracle	0.5287	0.4801	0.5356	0.5266	0.5874

Table 5.5: LSGM accuracy for different graph matching algorithms at different max cluster sizes with $\rho = 0.6$.

100 vertices each, 20 randomly chosen seed vertices, and $\rho = 0.6$. The Λ matrix is $\lambda_{ii} = 0.3$ on the diagonals and $\lambda_{ij} = 0.6$ for $i \neq j$ on off-diagonal entries. In these experiments, the oracle accuracy is also computed. The oracle accuracy is the maximum possible accuracy given the clustering in LSGM.

The results presented in Table 5.5 shows that the optimal max cluster size is about 300 vertices. The only exception is SGM, which closely matches the oracle accuracy.

In the next example, the parameters are all the same except $\rho = 0.9$, results are shown in Table5.6. With increased correlation, the accuracy also increased. Again SGM closely matches the oracle accuracy. Some of the graph matching algorithms

max cluster size	100	200	300	400	500
RAND	0.0126	0.0046	0.0035	0.0025	0.0023
U	0.0170	0.0092	0.0052	0.0048	0.0037
RANK	0.0285	0.0153	0.0119	0.0091	0.0059
QCV	0.1575	0.1141	0.0780	0.0592	0.0161
FAQ	0.0478	0.0173	0.0092	0.0075	0.0047
PATH	0.2556	0.2863	0.3447	0.2167	0.0917
GLAG	0.0993	0.0775	0.0719	0.0554	0.0411
SGM	0.6823	0.6483	0.6761	0.6971	0.7029
Oracle	0.6831	0.6483	0.6762	0.6972	0.7029

Table 5.6: LSGM accuracy for different graph matching algorithms at different max cluster sizes with $\rho = 0.9$.

max cluster size	100	200	300	400	500
RAND	3.2596	2.6439	2.4327	2.4810	2.9755
U	3.4398	2.8298	2.9099	3.2896	5.1298
RANK	3.3780	2.7582	2.6970	2.7694	3.6935
QCV	8.9928	15.3820	20.2138	20.4410	12.4421
FAQ	11.9592	27.1262	44.1289	60.6957	88.6032
SGM	7.7951	18.6063	31.8337	42.6478	56.5288

Table 5.7: LSGM runtime for different graph matching algorithms at different max cluster sizes with $\rho = 0.6$.

perform about the same as the lower correlation, U, RANK, FAQ, GLAG, while QCV, PATH, and SGM perform moderately better with higher correlation.

5.3.1.2.1 Runtime

Here are the runtimes for the 10-block stochastic block model simulation experiments. First in Table 5.7 is the runtime for when $\rho = 0.6$.

Next in Table 5.8 is the runtime for the $\rho = 0.9$ LSGM experiments. The runtimes shows that when $\rho = 0.9$, the runtime is generally the same. Only SGM has noticeably quicker runtimes.

max cluster size	100	200	300	400	500
RAND	3.4980	2.7161	2.6049	3.0694	3.3027
U	3.6668	2.9568	3.1161	3.8657	5.3905
RANK	3.5652	2.9044	2.8677	3.3720	3.9648
QCV	8.9891	14.5505	23.4681	18.5633	13.0882
FAQ	11.6440	26.6528	45.0535	58.4471	83.2101
SGM	5.1524	12.4141	20.9582	31.7462	45.4332

Table 5.8: LSGM runtime for different graph matching algorithms at different max cluster sizes with $\rho = 0.9$.

5.3.1.3 Range of Effectiveness

To demonstrate the regions where LSGM is effective we generated a surface plot in figure 5.2. For the ρ-correlated stochastic block models, $m \in \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$ and $\rho \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ are varied while $d = 10$,$K = 900$, and $M + N = 30 \cdot \vec{1}$ are fixed. This makes the graph size 27000 vertices, well beyond the size for which SGM is feasible.

Notice that LSGM is able to perform well with only 15 seed vertices and correlation $\rho = 0.7$, or 25 seed vertices and correlation $\rho = 0.5$.

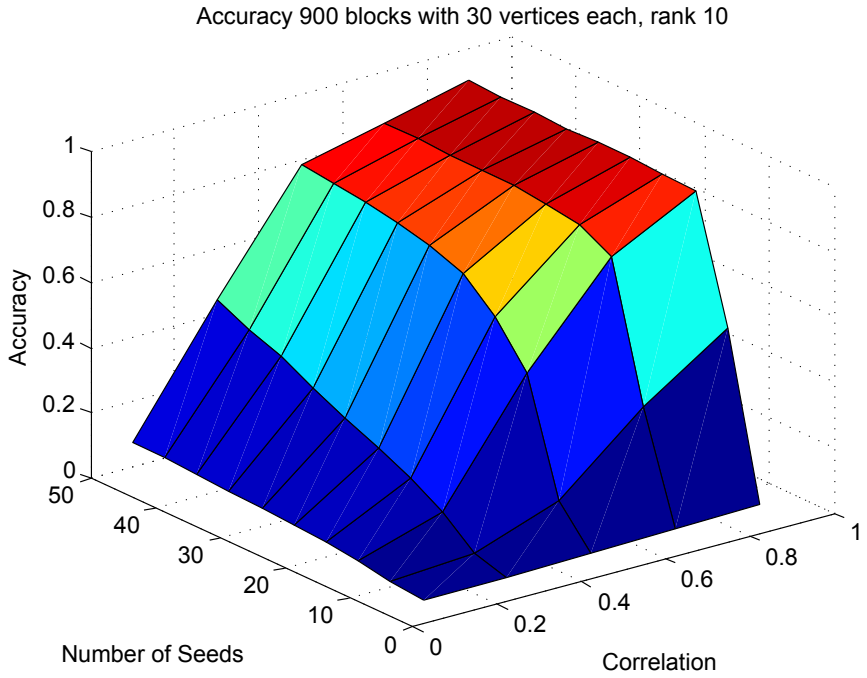

Figure 5.2: Monte Carlo LSGM simulation for 900 blocks with 30 vertices each, with $\text{rank}(\Lambda) = 10$ and SGM as the graph matching algorithm. With 50 seeds and $\rho = 0.5$ the accuracy is 78.75%.

$s\backslash\rho$	0.1	0.3	0.5	0.7	0.9
5	0.0003	0.0006	0.0030	0.0055	0.0106
10	0.0054	0.0196	0.0694	0.2325	0.1958
15	0.0101	0.0256	0.0281	0.0223	0.0074
20	0.0013	0.0056	0.0190	0.0120	0.0051
25	0.0017	0.0065	0.0094	0.0089	0.0044
30	0.0019	0.0069	0.0085	0.0082	0.0048
35	0.0016	0.0072	0.0070	0.0070	0.0070
40	0.0019	0.0106	0.0123	0.0109	0.0031
45	0.0020	0.0104	0.0108	0.0048	0.0036
50	0.0009	0.0076	0.0058	0.0077	0.0059

Table 5.9: LSGM standard deviation for simulated data.

5.3.2 Real Data

In this section we LSGM on brain graphs. The graph is generated from a 3-D brain scan. The vertices are voxels in the scan, and edges are neural connections between voxels. For more details on how the brain graphs were created see [3, 97]. The data is freely available at http://openconnecto.me/data/public/MR/MIGRAINE/. We binarized, symmetrized and down-sampled the graphs. Down-sampling is done by treating an $8 \times 8 \times 8$ voxel as a single voxel. After processing two graphs we want to match, we compute the intersection of the vertex sets. This results in a graph with approximately 18000 vertices.

When we apply the LSGM algorithm with SGM, we set the maximum cluster size to 800. We estimated the optimal embedding dimension to be 30 from the scree plot. Each experiment is repeated 30 times. In Figure 5.3 we compare the performance of matching two different subjects, graphs 8 and 29 against the performance of matching two brain scans of the same subject, graphs 1 and 8. For experiments with large numbers of seeds ($s = 1000, 2000, 5000$), we used the procedure is Section 5.1.1 to select a subset of seeds for each subgraph.

As one would expect, matching two brain scans of the same person performs better than of two different people. The low accuracy can be caused by noise in both the raw brain scans and pre-processing tools used to register and generate the graphs.

In Table 5.11 we present the effectiveness of LSGM with other graph matching algorithms on within subject graph matching. The maximum cluster size is again

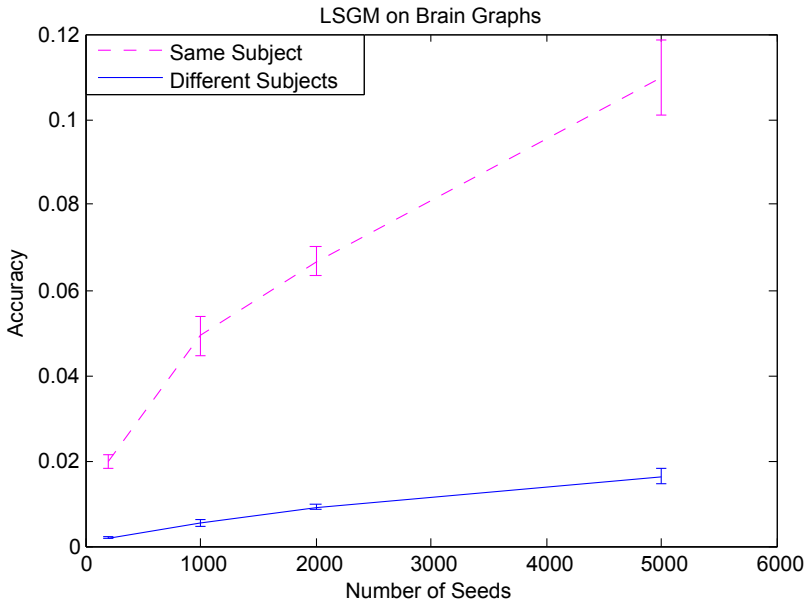

Figure 5.3: LSGM for matching brain connectome graphs $n \approx 18000$, $d = 30$, with 30 MC simulates and ± 2 standard error bars. The maximum cluster size is 800.

	across-subject		within-subject	
seeds	accuracy	std	accuracy	std
200	0.0019	0.0004	0.0198	0.0024
1000	0.0054	0.0013	0.0493	0.0074
2000	0.0092	0.0009	0.0667	0.0054
5000	0.0164	0.0028	0.1098	0.0140

Table 5.10: LSGM accuracy on brain graphs across-subject vs within-subject.

s	2500	5000
rand	0.0017	0.0020
U	0.0034	0.0040
rank	0.0039	0.0047
QCV	0.0091	0.0106
FAQ	0.0064	0.0066
SGM	0.0773	0.1074

Table 5.11: LSGM accuracy for brain graphs for 2500 and 5000 seed vertices.

800 vertices and 20 simulations were used for each experiment. In Table 5.12 are the corresponding runtimes per simulation in hours.

5.4 Discussion

The task of graph matching is important to many graph inference tasks, such as vertex nomination. In this chapter, we have presented a parallelizable graph match algorithm, whose computational complexity is $O(n^2 d)$. This is a large improvement over the complexity of state-of-the-art approximate graph matching algorithms $O(n^3)$. We have shown in simulated and real data examples the ability of LSGM to perform graph matching. Furthermore, we have shown, under mild conditions, theoretical results proving LSGM perfectly matches ρ-correlated stochastic block model graphs.

seeds	2500	5000
SGM	3.5669	4.4679
QAP	3.2825	4.2366
RAND	2.8425	3.6606
U	2.8496	3.5737
RANK	2.7859	3.2437
QCV	2.5952	2.9329

Table 5.12: LSGM runtime on brain graphs for 2500 and 5000 seed vertices in hours.

The choice of graph matching and clustering algorithms are left up to the user. We chose to use k-means for its simplicity and known theoretical results. The clustering procedure should be tailored to the data. For example, if the vertex embeddings resemble a mixture of Gaussians one should use `Mclust` instead. Choice of the graph matching algorithm should not be limited to SGM. Currently SGM is the only algorithm available able to use seed vertices. As other algorithms such as GLAG and PATH are able to utilize seed vertices, experiments should be used to determine their effectiveness in the LSGM algorithm.

Bibliography

[1] L. Euler, *The seven bridges of Konigsberg.* Wm. Benton, 1956.

[2] K. Haris, S. N. Efstratiadis, N. Maglaveras, C. Pappas, J. Gourassas, and G. Louridas, "Model-based morphological segmentation and labeling of coronary angiograms," *Medical Imaging, IEEE Transactions on*, vol. 18, no. 10, pp. 1003–1015, 1999.

[3] W. G. Roncal, Z. H. Koterba, D. Mhembere, D. M. Kleissas, J. T. Vogelstein, R. Burns, A. R. Bowles, D. K. Donavos, S. Ryman, R. E. Jung *et al.*, "Migraine: Mri graph reliability analysis and inference for connectomics," *arXiv preprint arXiv:1312.4875*, 2013.

[4] F. Zhou and F. De la Torre, "Factorized graph matching," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on.* IEEE, 2012, pp. 127–134.

[5] M. Cho and K. M. Lee, "Progressive graph matching: Making a move of graphs

via probabilistic voting," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 398–405.

[6] P. Boldi and S. Vigna, "The WebGraph framework I: Compression techniques," in *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*. Manhattan, USA: ACM Press, 2004, pp. 595–601.

[7] P. Boldi, M. Rosa, M. Santini, and S. Vigna, "Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks," in *Proceedings of the 20th international conference on World Wide Web*. ACM Press, 2011.

[8] C. E. Priebe, J. M. Conroy, D. J. Marchette, and Y. Park, "Scan statistics on enron graphs," *Computational & Mathematical Organization Theory*, vol. 11, no. 3, pp. 229–247, 2005.

[9] B. Bollobás, *Random Graphs*. Cambridge university press, 2001, vol. 73.

[10] A. Athreya, V. Lyzinski, D. J. Marchette, C. E. Priebe, D. L. Sussman, and M. Tang, "A limit theorem for scaled eigenvectors of random dot product graphs," *arXiv preprint arXiv:1305.7388*, 2013.

[11] A. Goldenberg, A. X. Zheng, S. E. Fienberg, and E. M. Airoldi, "A survey of statistical network models," *Foundations and Trends® in Machine Learning*, vol. 2, no. 2, pp. 129–233, 2010.

[12] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social networks*, vol. 5, no. 2, pp. 109–137, 1983.

[13] T. A. Snijders and K. Nowicki, "Estimation and prediction for stochastic blockmodels for graphs with latent block structure," *Journal of classification*, vol. 14, no. 1, pp. 75–100, 1997.

[14] 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*, vol. 106, no. 50, pp. 21 068–21 073, 2009.

[15] D. S. Choi, P. J. Wolfe, and E. M. Airoldi, "Stochastic blockmodels with a growing number of classes," *Biometrika*, p. asr053, 2012.

[16] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," in *Advances in Neural Information Processing Systems*, 2009, pp. 33–40.

[17] H. Zhang, B. Qiu, C. L. Giles, H. C. Foley, and J. Yen, "An lda-based community structure discovery approach for large-scale social networks," in *Intelligence and Security Informatics, 2007 IEEE*. IEEE, 2007, pp. 200–207.

[18] K. Rohe, S. Chatterjee, and B. Yu, "Spectral clustering and the high-dimensional stochastic blockmodel," *The Annals of Statistics*, vol. 39, no. 4, pp. 1878–1915, 2011.

[19] D. L. Sussman, M. Tang, D. E. Fishkind, and C. E. Priebe, "A consistent adjacency spectral embedding for stochastic blockmodel graphs," *Journal of the American Statistical Association*, vol. 107, no. 499, pp. 1119–1128, 2012.

[20] D. E. Fishkind, D. L. Sussman, M. Tang, J. T. Vogelstein, and C. E. Priebe, "Consistent adjacency-spectral partitioning for the stochastic block model when the model parameters are unknown," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 1, pp. 23–39, 2013.

[21] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," *International journal of pattern recognition and artificial intelligence*, vol. 18, no. 03, pp. 265–298, 2004.

[22] M. R. Garey and D. S. Johnson, *Computers and intractability.* wh freeman, 2002, vol. 29.

[23] S. Sahni and T. Gonzalez, "P-complete approximation problems," *Journal of the ACM (JACM)*, vol. 23, no. 3, pp. 555–565, 1976.

[24] R. C. Read and D. G. Corneil, "The graph isomorphism disease," *Journal of Graph Theory*, vol. 1, no. 4, pp. 339–363, 1977.

[25] A. Lubiw, "Some np-complete problems similar to graph isomorphism," *SIAM Journal on Computing*, vol. 10, no. 1, pp. 11–21, 1981.

BIBLIOGRAPHY

[26] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, "The design and analysis of computer algorithms, 1974," *Reading: Addison-Wesley*, pp. 207–209, 1987.

[27] J. E. Hopcroft and J.-K. Wong, "Linear time algorithm for isomorphism of planar graphs (preliminary report)," in *Proceedings of the sixth annual ACM symposium on Theory of computing*. ACM, 1974, pp. 172–184.

[28] E. M. Luks, "Isomorphism of graphs of bounded valence can be tested in polynomial time," *Journal of Computer and System Sciences*, vol. 25, no. 1, pp. 42–65, 1982.

[29] J. R. Ullmann, "An algorithm for subgraph isomorphism," *Journal of the ACM (JACM)*, vol. 23, no. 1, pp. 31–42, 1976.

[30] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach (International Edition)*. {Pearson US Imports & PHIPEs}, 2002.

[31] W.-H. Tsai and K.-S. Fu, "Error-correcting isomorphisms of attributed relational graphs for pattern analysis," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 9, no. 12, pp. 757–768, 1979.

[32] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 10, no. 5, pp. 695–703, 1988.

[33] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on Computers*, vol. 22, no. 1, pp. 67–92, 1973.

[34] J. T. Vogelstein, J. M. Conroy, L. J. Podrazik, S. G. Kratzer, D. E. Fishkind, R. J. Vogelstein, and C. E. Priebe, "Fast inexact graph matching with applications in statistical connectomics," *submitted for publication*, 2011.

[35] A. Rukhin, *Asymptotic Analysis of Various Statistics for Random Graph Inference*. Proquest, Umi Dissertation, 2011.

[36] Y. J. Wang and G. Y. Wong, "Stochastic blockmodels for directed graphs," *Journal of the American Statistical Association*, vol. 82, no. 397, pp. 8–19, 1987.

[37] S. Milgram, "The small world problem," *Psychology today*, vol. 2, no. 1, pp. 60–67, 1967.

[38] M. E. Newman, "The structure and function of complex networks," *SIAM review*, vol. 45, no. 2, pp. 167–256, 2003.

[39] G. Schwarz *et al.*, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[40] H. Akaike, "A new look at the statistical model identification," *Automatic Control, IEEE Transactions on*, vol. 19, no. 6, pp. 716–723, 1974.

[41] L. Chen, J. Vogelstein, and C. Priebe, "Robust vertex classification," *arXiv preprint arXiv:1311.5954*, 2013.

[42] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[43] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Applied statistics*, pp. 100–108, 1979.

[44] C. Fraley and A. E. Raftery, "Mclust: Software for model-based cluster analysis," *Journal of Classification*, vol. 16, no. 2, pp. 297–306, 1999.

[45] ——, "Enhanced model-based clustering, density estimation, and discriminant analysis software: Mclust," *Journal of Classification*, vol. 20, no. 2, pp. 263–286, 2003.

[46] ——, "Mclust version 3: an r package for normal mixture modeling and model-based clustering," DTIC Document, Tech. Rep., 2006.

[47] V. Lyzinski, D. Sussman, M. Tang, A. Athreya, and C. Priebe, "Perfect clustering for stochastic blockmodel graphs via adjacency spectral embedding," *arXiv preprint arXiv:1310.0532*, 2013.

[48] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.

[49] K. Nowicki and T. A. B. Snijders, "Estimation and prediction for stochastic blockstructures," *Journal of the American Statistical Association*, vol. 96, no. 455, pp. 1077–1087, 2001.

[50] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[51] T. Caelli and S. Kosinov, "An eigenspace projection clustering method for inexact graph matching," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 4, pp. 515–519, 2004.

[52] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 2, pp. 224–227, 1979.

[53] D. Knossow, A. Sharma, D. Mateus, and R. Horaud, "Inexact matching of large and sparse graphs using laplacian eigenvectors," in *Graph-Based Representations in Pattern Recognition*. Springer, 2009, pp. 144–153.

[54] M. Zaslavskiy, F. Bach, and J.-P. Vert, "A path following algorithm for the graph matching problem," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 12, pp. 2227–2242, 2009.

[55] V. Lyzinski, D. Fishkind, M. Fiori, J. T. Vogelstein, C. E. Priebe, and G. Sapiro, "Graph matching: Relax at your own risk," *arXiv preprint arXiv:1405.3133*, 2014.

[56] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[57] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*. Courier Dover Publications, 1998.

[58] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[59] R. E. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems, Revised Reprint*. Siam, 2009.

[60] D. E. Fishkind, S. Adali, and C. E. Priebe, "Seeded graph matching," *arXiv preprint arXiv:1209.0367*, 2012.

[61] V. Lyzinski, D. E. Fishkind, and C. E. Priebe, "Seeded graph matching for correlated erdos-renyi graphs," *arXiv preprint arXiv:1304.7844*, 2013.

[62] R. Singh, J. Xu, and B. Berger, "Pairwise global alignment of protein interaction networks by matching neighborhood topology," in *Research in computational molecular biology*. Springer, 2007, pp. 16–31.

[63] M. Fiori, P. Sprechmann, J. Vogelstein, P. Musé, and G. Sapiro, "Robust multimodal graph matching: Sparse coding meets graph matching," in *Advances in Neural Information Processing Systems*, 2013, pp. 127–135.

BIBLIOGRAPHY

[64] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[65] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *Proceedings of the National academy of Sciences of the United States of America*, vol. 101, no. Suppl 1, pp. 5228–5235, 2004.

[66] T. Minka and J. Lafferty, "Expectation-propagation for the generative aspect model," in *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2002, pp. 352–359.

[67] H. Pao, G. A. Coppersmith, and C. E. Priebe, "Statistical inference on random graphs: Comparative power analyses via monte carlo," *Journal of Computational and Graphical Statistics*, vol. 20, no. 2, pp. 395–416, 2011.

[68] P. J. Bickel and K. A. Doksum, "Mathematical statistics, volume i," 2001.

[69] A. Rukhin and C. E. Priebe, "A comparative power analysis of the maximum degree and size invariants for random graph inference," *Journal of Statistical Planning and Inference*, vol. 141, no. 2, pp. 1041–1046, 2011.

[70] A. V. Goldberg, *Finding a maximum density subgraph*. University of California Berkeley, CA, 1984.

[71] E. R. Scheinerman and D. H. Ullman, *Fractional graph theory: a rational approach to the theory of graphs*. Courier Dover Publications, 2011.

[72] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[73] A. Rukhin and C. E. Priebe, "On the limiting distribution of a graph scan statistic," *Communications in Statistics-Theory and Methods*, vol. 41, no. 7, pp. 1151–1170, 2012.

[74] K. Nowicki and J. C. Wierman, "Subgraph counts in random graphs using incomplete u-statistics methods," *Discrete Mathematics*, vol. 72, no. 1, pp. 299–310, 1988.

[75] J. M. Keil and T. B. Brecht, "The complexity of clustering in planar graphs," *J. Combinatorial Mathematics and Combinatorial Computing*, vol. 9, pp. 155–159, 1991.

[76] S. Khot, "Ruling out ptas for graph min-bisection, dense k-subgraph, and bipartite clique," *SIAM Journal on Computing*, vol. 36, no. 4, pp. 1025–1071, 2006.

[77] U. Feige, D. Peleg, and G. Kortsarz, "The dense k-subgraph problem," *Algorithmica*, vol. 29, no. 3, pp. 410–421, 2001.

[78] D. E. Fishkind, V. Lyzinski, H. Pao, L. Chen, and C. E. Priebe, "Vertex nomination schemes for membership prediction," *arXiv preprint arXiv:1312.2638*, 2013.

[79] G. A. Coppersmith and C. E. Priebe, "Vertex nomination via content and context," *arXiv preprint arXiv:1201.4118*, 2012.

[80] D. S. Lee and C. E. Priebe, "Bayesian vertex nomination," *arXiv preprint arXiv:1205.5082*, 2012.

[81] L. Devroye, *A probabilistic theory of pattern recognition*. springer, 1996, vol. 31.

[82] P. Erdős and A. Rényi, "Asymmetric graphs," *Acta Mathematica Hungarica*, vol. 14, no. 3, pp. 295–315, 1963.

[83] G. Pólya, "Kombinatorische anzahlbestimmungen für gruppen, graphen und chemische verbindungen," *Acta mathematica*, vol. 68, no. 1, pp. 145–254, 1937.

[84] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[85] J. G. Propp and D. B. Wilson, "Exact sampling with coupled markov chains and applications to statistical mechanics," *Random structures and Algorithms*, vol. 9, no. 1-2, pp. 223–252, 1996.

[86] J. A. Fill, "An interruptible algorithm for perfect sampling via markov chains," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. ACM, 1997, pp. 688–695.

[87] S. Chib and E. Greenberg, "Understanding the metropolis-hastings algorithm," *The American Statistician*, vol. 49, no. 4, pp. 327–335, 1995.

[88] G. O. Roberts and A. F. Smith, "Simple conditions for the convergence of the gibbs sampler and metropolis-hastings algorithms," *Stochastic processes and their applications*, vol. 49, no. 2, pp. 207–216, 1994.

[89] K. L. Mengersen, R. L. Tweedie *et al.*, "Rates of convergence of the hastings and metropolis algorithms," *The Annals of Statistics*, vol. 24, no. 1, pp. 101–121, 1996.

[90] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 us election: divided they blog," in *Proceedings of the 3rd international workshop on Link discovery*. ACM, 2005, pp. 36–43.

[91] V. Lyzinski, D. L. Sussman, D. E. Fishkind, H. Pao, and C. E. Priebe, "Seeded graph matching for large stochastic block model graphs," *arXiv preprint arXiv:1310.1297*, 2013.

[92] M. Cho and J. Lee, "Feature correspondence and deformable object matching via agglomerative correspondence clustering," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1280–1287.

[93] A. Armiti and M. Gertz, "Efficient geometric graph matching using vertex em-

bedding," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2013, pp. 224–233.

[94] P. H. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, 1966.

[95] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[96] M. Brand, "Fast low-rank modifications of the thin singular value decomposition," *Linear algebra and its applications*, vol. 415, no. 1, pp. 20–30, 2006.

[97] W. R. Gray, J. A. Bogovic, J. T. Vogelstein, B. A. Landman, J. L. Prince, and R. Vogelstein, "Magnetic resonance connectome automated pipeline: an overview," *Pulse, IEEE*, vol. 3, no. 2, pp. 42–48, 2012.

Vita

Henry Pao was born in Baltimore, MD. He attended Centennial High school and graduated in 2004. Then he attended Johns Hopkins University and graduated with a Bachelor's Degree in 2007, and a Master's Degree in 2008. Both degrees are in Applied Mathematics and Statistics. He continued to study at Johns Hopkins University to receive a Master's in Computer Science in 2011.