Pattern Recognition on Random Graphs

by

Li Chen

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

Baltimore, Maryland

February, 2015

Abstract

The field of pattern recognition developed significantly in the 1960s, and the field of random graph inference has enjoyed much recent progress in both theory and application. This dissertation focuses on pattern recognition in the context of a particular family of random graphs, namely the stochastic blockmodels, from the two main perspectives of single graph inference and joint graph inference.

Single graph inference is the performance of statistical inference on one single observed graph. Given a single graph realized from a stochastic blockmodel, we here consider the specific exploitation tasks of vertex classification, clustering, and nomination.

Given an observed graph, vertex classification is the identification of the block labels of test vertices after learning from the training vertices. We propose a robust vertex classifier, which utilizes a representation of a test vertex as a sparse combination of the training vertices [17]. Our proposed classifier is demonstrated to be robust against data contamination, and has superior performance over classical spectral-embedding classifiers in simulation and real data experiments.

ABSTRACT

Vertex clustering groups vertices based on their similarities. We present a model-based clustering algorithm for graphs drawn from a stochastic blockmodel, and illustrate its usefulness on a case study in online advertising [16]. We demonstrate the practical value of our vertex clustering method for efficiently delivering internet advertisements.

Under the stochastic blockmodel framework, suppose one block is of particular interest. The task of vertex nomination is to create a nomination list so that vertices from the group of interest are concentrated abundantly near the top of the list. We present several vertex nomination schemes [40], and propose a vertex nomination scheme that is scalable for large graphs [19]. We demonstrate the effectiveness of our methodology on simulation and real datasets.

Next, we consider joint graph inference, which involves the joint space of multiple graphs; in this dissertation, we specifically consider joint graph inference on two graphs. Given two graphs, we consider the tasks of seeded graph matching for large graphs and joint vertex classification.

Graph matching is the task of aligning two graphs so as to minimize the number of edge disagreements between them. We propose a scalable graph matching algorithm, which uses a divide-and-conquer approach to scale the state-of-the-art seeded graph matching algorithm to big graph data [59]. We prove theoretical performance guarantees, and demonstrate desired properties such as scalability, robustness, accuracy and runtime in both simulated data and human brain connectome data.

ABSTRACT

Within the joint graph inference framework, we present a case study on the paired chemical and electrical *Caenorhabditis elegans* neural connectomes [18]. Motivated by the success of seeded graph matching on the paired connectomes, we propose joint vertex classification on the paired connectomes. We demonstrate that joint inference on the paired connectomes yields more accurate results than single inference on either connectome. This serves as a first step for providing a methodological and quantitative approach for understanding the coexistent significance of the chemical and electrical connectomes.

Primary Reader: Carey Priebe

Secondary Reader: Donniell Fishkind

Acknowledgments

First, I would like to thank my parents, who support me to study in the US, and always encourage me to overcome difficulties. Their love and inspiration always give me courage and positive energy to keep doing my best. I am very lucky and grateful to have them in my life.

I offer my sincerest gratitude to my advisor Prof. Carey Priebe, who is both a patient teacher and an amazing friend. He introduced me to the interesting topics in random graph inference and data science. He is constantly willing to help and urge me to produce good quality research. Discussions with him are always great pleasure. He is the funniest and most energetic advisor, and I could not wish for a better advisor.

I would like to express my sincere thanks to my defense committee members: Carey Priebe, Donniell Fishkind, Vince Lyzinki, Joshua Vogelstein and Avanti Athreya for their support. I cannot express enough thanks to Donniell Fishkind, Vince Lyzinki, Henry Pao, Avanti Athreya, Minh Tang, Nam Lee and Youngser Park for proofreading my thesis and presentation slides.

ACKNOWLEDGMENTS

My many thanks go to my collaborators: Carey Priebe, Donniell Fishkind, Vince Lyzinki, Daniel Sussman, Henry Pao, Youngser Park, Joshua Vogelstein, Bruno Jedynak, Cencheng Shen and many more. My research would not have been possible without their help. I would also like to thank my friends Theodore Drivas, Allen Cheng, Nam Lee, Minh Tang, Runze Tang, Jordan Yoder, and many more, who make my time at Hopkins very enjoyable.

I cannot quantify my appreciation to the wonderful Henry for his care, kindness and encouragement. His faithful support and unconditional love to me is invaluable. My heartfelt thanks also go to his parents for helping me whenever and wherever.

I would like to offer my deepest gratitude to my grandmother, who introduced me to the fascinating world of mathematics, when I was a little child. She was so kind, caring and loving. I will always remember her smile and encouragement, and she will always live in my heart.

Dedication

This thesis is dedicated to my beloved parents and grandmother, without whom this dissertation would not have existed.

Contents

CONTENTS

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction and Overview

Pattern recognition is a field in machine learning, in which we learn and discover patterns from data. Graphs describe a structure that contains interacting objects; the objects are called vertices, where some pairs of the vertices are adjacent by links or edges. The graph data structure is useful for representing diverse phenomena. For example, in a social network, the vertices are people, where some pairs of people are adjacent by their friendships. In a neural connectome, the vertices are neurons, where some pairs of neurons are adjacent by the synaptic interactions. In a webpage network, the vertices are websites, where some pairs of the webpages are linked via URL links. The term "random graph" here will refer to graphs which have a fixed vertex set and a random edge set. There has been a surge of interest in studying graphs in the fields of statistics, machine learning, computer vision, neuroscience, social science, and so on. The recent advances in random graph research has proven

to be very valuable in both theory and applications.

In this dissertation, we are concerned with two main topics in pattern recognition for random graphs namely: single graph inference – inference performed on one single graph – and joint graph inference – inference performed on the joint space of multiple graphs (here we particularly consider two graphs). We formulate our inference methodologies in the context of a stochastic blockmodel.

In Chapter 2, we present the random graph models considered throughout this dissertation. The latent position graph is a very general random graph model, where we assume that each vertex is associated with a latent random vector, which stochastically generates the adjacency matrix. The random dot product model is a special case of the latent position model. The stochastic blockmodel is a special case of the random dot product model, and is widely used for exploratory graph analysis; it enables the development of many principled algorithms.

Chapter 3–5 present our proposed inference methodologies within single graph inference framework. In Chapter 3, we consider the task of vertex classification, where a subset of vertices have known labels, and we want to predict the labels of the remaining vertices. Here the class labels for classification are the block memberships in the stochastic blockmodel. We propose a sparse representation vertex classifier, which represents the test vertex as a sparse combination of the training vertices, and uses the recovered coefficients to classify the test vertex [17]. This classifier does not critically rely on the model dimension of the stochastic blockmodel, while the success

of classifiers combined with adjacency spectral embedding relies heavily on the model dimension. We propose two contamination models, where both contamination procedures result in a change of the model dimension. We compare the performance of our proposed vertex classifier with two other classifiers: adjacency spectral embedding followed by the nearest neighbor classifier and adjacency spectral embedding followed by linear discriminant analysis on both simulated and real data. Our proposed classifier demonstrates superior or competitive classification performance.

In Chapter 4, we consider the task of vertex clustering, which is the task to group vertices from the same block of a stochastic blockmodel into the same cluster. We present a model-based clustering approach using the Bayesian information criterion [16]. We compare its performance with a goodness-of-fit likelihood method and a deterministic partitioning method on simulated and on real data. Our case study examines our clustering algorithm's applications in online advertising, where we are concerned with the business problem of targeting a wider variety of online users based on the relational events of users visiting websites. The business motivation of applying our approach is to reduce the cost of buying web pages for ads posts while still reaching a broad variety of the target audience. We demonstrate the effectiveness of our approach for efficient online advertisement delivery.

In Chapter 5, we consider the task of vertex nomination, where the task is to create a nomination list such that an abundance of "interesting" vertices are at the top of the list. We have previously proposed several vertex nomination schemes:

canonical vertex nomination, likelihood maximization vertex nomination, and spectral vertex nomination scheme [40]. The canonical vertex nomination is proven to be the best possible vertex nomination scheme using certain metric, but it is only computationally feasible for graphs of very few vertices. In this chapter, we propose a canonical sampling vertex nomination scheme which not only approximates the optimal performance quality of the canonical vertex nomination scheme, but also scales to large graphs [19]. We prove theoretical guarantees for our proposed scheme, and demonstrate its performance in simulated and real data.

Joint graph inference framework focuses on inference in the joint space of multiple graphs. In this dissertation, the joint inference is performed on two graphs.

In Chapter 6, we are concerned with the problem of seeded graph matching for large graphs. Graph matching is the task to find an alignment between the vertices across two graphs such that it minimizes the number of edge disagreements. The problem of seeded graph matching leverages information of a partially known vertex correspondence, and finds an alignment between the remaining vertices. The state-of-the-art seeded graph matching algorithm is limited to thousands of vertices. We propose a divide-and-conquer approach to scale the seeded graph matching algorithm to big graph data. We present the theoretical performance guarantee of our algorithm under the stochastic blockmodel assumption, and demonstrate its effectiveness in scalability, accuracy, run time and robustness via simulation and a human brain connectome experiment.

In Chapter 7, we present a joint graph inference case study for the pair of neural connectomes of the *Caenorhabditis elegans*. We formulate our joint graph inference from the perspectives of seeded graph matching and joint vertex classification. Our analysis results indicate that we should perform inference in the joint space of the neural connectomes. Our proposed joint graph inference provides a methodological and quantitative framework for understanding the significance of the coexistence of the chemical and the electrical synapses.

Chapter 8 concludes this dissertation by summarizing our work in single and joint graph inference.

Chapter 2

Statistical Models of Random Graphs

A graph $G = (V, E)$ contains a vertex set $V = [n] := \{1, 2, \ldots, n\}$, where we use integers to denote the vertices, and edge set $E \subset \binom{[n]}{2}$. In this dissertation, we assume the graph is undirected, unweighted and non-loopy. The adjacency matrix A is of order n, binary, symmetric and hollow, i.e., the diagonal entries a_{ii} of A are 0. Each entry $a_{ij} \in \{0, 1\}$ for $i \neq j$ denotes the edge existence between vertex i and j, where 1 means edge existence and 0 otherwise.

A random graph is a graph-valued random variable: $\mathcal{G} : \Omega \to \mathcal{G}_n$, where $\mathcal{G}_n$ represents the collection of all $2^{\binom{[n]}{2}}$ possible graphs on the vertex set $V = [n]$, and Ω is a probability space. Associated with the adjacency matrix $A \in \{0, 1\}^{n \times n}$, there exists a communication probability matrix $P \in [0, 1]^{n \times n}$, where each entry P_{ij} denotes

the probability of edge existence between vertex i and vertex j. Below, we introduce several random graph models studied in this dissertation.

2.1 Latent Position Models

The latent position model (LPM) was proposed in [49]. In this model, each vertex i is associated with a latent random variable $X_i \in \mathbb{R}^d$ drawn independently from a specified distribution F on $\mathbb{R}^d$. These latent variables determine the probabilities of edge existence. We formally define the latent position model as the following:

Definition 2.1. ***Latent Position Model (LPM)*** Let F be a distribution on $\mathbb{R}^d$. Let $X_1, ..., X_n \overset{iid}{\sim} F$ and define $\mathcal{X} := [X_1, ..., X_n]^T \in \mathbb{R}^{n \times d}$. Let $P \in [0,1]^{n \times n}$ be the communication probability matrix, where each entry P_{ij} is the probability that there is an edge between vertices i, j conditioned on X_i and X_j for $\forall i, j$. Let $A \in \{0,1\}^{n \times n}$ be the random adjacency matrix. Then the graph G is realized from a latent position model $G \sim LPM(F)$ if there is a link function $l : \mathbb{R}^d \times \mathbb{R}^d \to [0,1]$ such that, for $\forall i, j$

$$\mathbb{P}(A|X_1, ..., X_n) = \Pi_{i<j} P_{ij}^{A_{ij}} (1 - P_{ij})^{1 - A_{ij}}, \tag{2.1}$$

$$P_{ij} = \mathbb{P}(A_{ij} = 1 | X_i, X_j) = l(X_i, X_j). \tag{2.2}$$

The random variables X_is are the latent positions for the model.

2.2 Random Dot Product Graphs

The random dot product graph model proposed in [98] is a special case of the latent position model, where the link function $l(X_i, X_j)$ is the inner product of latent positions, $l(X_i, X_j) = \langle X_i, X_j \rangle$ with the restriction $\langle X_i, X_j \rangle \in [0, 1]$ for $\forall i, j$. The random dot product model is defined as follows:

Definition 2.2. ***Random Dot Product Graph (RDPG)*** Let F, $\mathcal{X}$, A be defined as in Section 2.1. Let $\mathcal{H}$ be the hollowing function that puts zeros along the diagonals of a matrix. Let P denote the symmetric communication probability matrix such that $P = \mathcal{H}(\mathcal{X}\mathcal{X}^T) \in [0, 1]^{n \times n}$. Then the graph is realized from a random dot product graph $G \sim RDPG(F)$ if

$$\mathbb{P}(A|X_1, ..., X_n) = \Pi_{i<j} P_{ij}^{A_{ij}} (1 - P_{ij})^{1-A_{ij}}, \tag{2.3}$$

$$P_{ij} = \mathbb{P}(A_{ij} = 1|X_i, X_j) = \langle X_i, X_j \rangle. \tag{2.4}$$

We say the RDPG is d-dimensional, if the rank of the communication probability matrix P is $d = \text{rank}(P)$.

2.3 Stochastic Blockmodels

In this section, we present the stochastic blockmodel, which is the underlying framework where we develop several inference methodologies. The stochastic blockmodel was introduced in [50]. It is a family of random graph models such that a set

of n vertices randomly belong to K blocks. Conditioned on the K-partition, edges between all the pairs of vertices are independent Bernoulli trials with parameters determined by the block memberships of the two vertices.

We introduce three different parametrizations of the stochastic blockmodels in Definitions 2.4, 2.5 and 2.6 useful for the task of vertex classification, graph matching and vertex nomination respectively. Even though the parametrizations are slightly different, the theoretical results of [41], [84], [85], [86], [63] and [6] hold for all three parametrizations.

Before proceeding with the definitions of stochastic blockmodels, we first review a related definition: the unit simplex.

Definition 2.3. ***Unit N-Simplex*** The unit N-simplex $\Delta^N \subset \mathbb{R}^{N+1}$ is a collection of points $t = (t_1, t_2, \ldots, t_{N+1}) \in \mathbb{R}^{N+1}$ such that

$$\Delta^N := \{t = (t_1, t_2, \ldots, t_{N+1}) \in \mathbb{R}^{N+1} | \sum_{i=0}^{N} t_i = 1, t_i \geq 0, \text{ for } \forall i\}. \tag{2.5}$$

We first present the stochastic blockmodel parametrized by the vertex set $[n] := \{1, 2, \ldots, n\}$, a symmetric matrix $B \in [0,1]^{K \times K}$, and a unit-length vector $\pi \in [0,1]^K$.

Definition 2.4. ***Stochastic Blockmodel SBM***$([n], B, \pi)$ Let K be the number of blocks. Let π be a length K vector in the unit simplex Δ^{K-1} specifying the block membership probabilities. The block membership of the vertex i is given by $Y_i \overset{iid}{\sim}$ Multinomial$([K], \pi)$. Let B be a $K \times K$ symmetric block communication probability matrix. Then the graph G is realized from a stochastic blockmodel

$G \sim SBM([n], B, \pi)$ if

$$\mathbb{P}(A|Y_1, ..., Y_n) = \Pi_{i<j} P_{ij}^{A_{ij}} (1 - P_{ij})^{1-A_{ij}}, \tag{2.6}$$

$$P_{ij} = \mathbb{P}(A_{ij} = 1|X_i, X_j) = \mathbb{P}(A_{ij} = 1|Y_i, Y_j) = B_{Y_i, Y_j}. \tag{2.7}$$

The following parametrization of SBM assumes the parameter of block sizes are fixed. We will use this definition of SBM for graph matching tasks in Chapter 6.

Definition 2.5. ***Stochastic Blockmodel SBM***$(\vec{n}, b, B)$ Let $K \in \mathbb{N}^+$ denote the number of blocks. Let $V_1, V_2, \ldots, V_K$ denote the K blocks. Let $\vec{n} = (n_1, n_2, \ldots, n_K) \in \mathbb{N}^K$ denote the size of the K blocks with $\sum_{k\in[K]} n_k = n$. Let b denote the unknown block assignment function $b : V \to \{1, 2, \ldots, K\}$, such that the cardinality $|V_k| = \{v \in V : b(v) = k\} = n_k$ for $k \in [K]$. Let $B \in [0, 1]^{K\times K}$ be a symmetric block communication matrix. We say that $G \sim \text{SBM}(\vec{n}, b, B)$ if and only if the conditional probability of edge existence is a Bernoulli random variable for any unordered pair of distinct vertices $\{i, j\} \in V$, i.e.,

$$\mathbb{P}(A_{ij} = 1|b(i), b(j)) = B_{b(i)b(j)}. \tag{2.8}$$

The following parametrization of SBM assumes the block assignment function is discretely uniformly distributed in the space of all possible block assignments. This definition is useful for vertex nomination in Chapter 5.

Definition 2.6. ***Stochastic Blockmodel SBM***$(\vec{n}, \mathcal{L}, B)$ Let $K \in \mathbb{N}^+$ denote the number of blocks. Let $V_1, V_2, \ldots, V_K$ denote the K blocks. Let $\vec{n} = (n_1, n_2, \ldots, n_K) \in \mathbb{N}^K$ denote the size of the K blocks with $\sum_{k\in[K]} n_k = n$. Let $B \in [0,1]^{K\times K}$ be a symmetric block communication matrix. Let $\mathcal{L}$ consist of all block assignment function $l : V \to \{1, 2, \ldots, K\}$, such that the cardinality $|V_k| = \{v \in V : l(v) = k\} = n_k$ for $k \in [K]$. A graph $G \sim \text{SBM}(\vec{n}, \mathcal{L}, B)$ may be realized via the following procedure. Let l be discrete-uniformly selected from $\mathcal{L}$. Conditioned on l, the probability of edge existence is a Bernoulli random variable for any unordered pair of distinct vertices $\{i, j\} \in V$, i.e.,

$$\mathbb{P}(A_{ij} = 1 | l(i), l(j)) = B_{l(i)l(j)}. \tag{2.9}$$

The main difference between $\text{SBM}(\vec{n}, b, B)$ and $\text{SBM}([n], \pi, B)$ is that $\text{SBM}(\vec{n}, b, B)$ assumes that the block sizes $\vec{n} = (n_1, n_2, \ldots, n_K)$ are fixed, while $\text{SBM}([n], \pi, B)$ has expected block sizes πn. The main difference between $\text{SBM}(\vec{n}, b, B)$ and $\text{SBM}(\vec{n}, \mathcal{L}, B)$ is that $\text{SBM}(\vec{n}, \mathcal{L}, B)$ assumes a uniform prior for the block assignment functions, while $\text{SBM}(\vec{n}, b, B)$ considers the block assignment function fixed but unknown. The parametrization $\text{SBM}([n], \pi, B)$ is useful for the mechanism of vertex classification and clustering, $\text{SBM}(\vec{n}, \mathcal{L}, B)$ is appropriate for explaining vertex nomination, and $\text{SBM}(\vec{n}, b, B)$ is useful for constructing correlated SBMs (see Section 2.4) for graph matching.

When the block communication probability matrix B is symmetric and diago-

nally dominant, the stochastic blockmodel is called positive semidefinite. The eigenstructure of the stochastic blockmodel is often low-rank, i.e., $d \ll n$. Let $\delta_1(P) \geq \delta_2(P) \geq \cdots \geq \delta_n(P)$ denote the singular values of P sorted in non-increasing order. Since the rank of the matrix P is $\text{rank}(P) = d$, then any singular values after the d-th position vanishes $\delta_j(P) = 0$ for all $j \geq d$.

Definition 2.7. ***Model Dimension*** For stochastic blockmodels, the model dimension refers to the rank of the block communication probability matrix, $\text{rank}(B) = d$. We say such a stochastic blockmodel is d-dimensional.

The stochastic blockmodel can be parametrized as a random dot product graph. Suppose $\text{rank}(B) = d$. There exists a unique (up to rotation) matrix $v \in \mathbb{R}^{K \times d}$ such that $B = vv^T$. By definition, $B_{ij} = \langle v_i, v_j \rangle$. Define $\mathcal{X} = [X_1, X_2, \ldots, X_n]^T$ to be the matrix containing the latent positions. Then each row $i \in [n]$ is given by $X_i^T = v_{Y_i}^T$. Then $\mathbb{P}(A_{ij} = 1 | Y_i, Y_j) = B_{Y_i, Y_j} = \langle v_{Y_i}, v_{Y_j} \rangle = \langle X_i, X_j \rangle$, which is precisely the random dot product model.

This RDPG parametrization of SBM is a convenient theoretical tool, and the spectral properties of RDPG are well-understood (see [41], [84], [85], [86], [63] and [6]).

2.4 Correlated Stochastic Blockmodels

We introduce correlation between two graphs G_1 and G_2 realized from the stochastic blockmodel $\mathrm{SBM}(\vec{n}, b, B)$ in Definition 2.5. Recall the definition of the Pearson product-moment correlation coefficient ρ between two random variables $\mathcal{V}_1$ and $\mathcal{V}_2$ is defined as

$$\rho = \frac{\mathrm{cov}(\mathcal{V}_1, \mathcal{V}_2)}{\sqrt{\mathrm{Var}(\mathcal{V}_1)}\sqrt{\mathrm{Var}(\mathcal{V}_2)}}, \tag{2.10}$$

where Var denotes the variance of the random variable.

Definition 2.8. ***ρ-Correlated Stochastic Blockmodels*** Let $v \sim_{G_1} v'$ denote the event that two vertices in G_1 are adjacent, and $\mathbb{1}_{v \sim_{G_2} v'}$ for two vertices in G_2 are adjacent. Two graphs G_1 and G_2 realized from $\mathrm{SBM}(\vec{n}, b, B)$ are ρ-correlated, if the set of all indicator random variables $\{\mathbb{1}_{v \sim_{G_i} v'}\}_{\{v,v'\},\{w,w'\} \in \binom{V}{2}, i \in \{1,2\}}$ are mutually independent, except that for each $\{v, v'\} \in \binom{V}{2}$, the indicator random variables $\mathbb{1}_{v \sim_{G_1} v'}$ and $\mathbb{1}_{v \sim_{G_2} v'}$ have Pearson product-moment correlation coefficient ρ.

Introducing a correlation between two SBMs as defined in Definition 2.4 naturally assumes there is a bijective alignment between the vertices across two graphs, thus making it suitable for the task of graph matching in Chapter 6. Note that the notion of correlated SBMs may not be suitable for the SBM in Definition 2.4, since the vertices will not be exactly aligned.

Simulation of the ρ-correlated stochastic blockmodels can be done in the following manner. We first realize G_1 from the underlying model $\mathrm{SBM}(\vec{n}, b, B)$ as in Definition

2.5. Given G_1, if v and v' are adjacent in G_1, then the probability that an edge exists between a pair of vertices in G_2, i.e., $\mathbb{P}(v \sim_{G_2} v')$ is $B(v, v') + \rho(1 - B(v, v'))$. If v and v' are not adjacent in G_1, then the probability that an edge exists between a pair of vertices in G_2 is $B(v, v')(1-\rho)$. In short, given the adjacency matrix A_1 of G_1, for each $\{v, v'\} \in \binom{V}{2}$, the indicator random variable $\mathbb{1}v \sim_{G_2} v'$ is an independent Bernoulli trial with probability of success $(1-\rho)B(v, v') + \rho\,[A_1]_{b(v)b(v')}$, where $[A_1]_{b(v)b(v')}$ is the $b(v)b(v')$-th entry of A_1.

2.5 Adjacency Spectral Embedding

In this section, we present the method of adjacency spectral embedding (ASE) for estimating the latent positions of the stochastic blockmodel [84]. This embedding technique uses a decomposition of a low rank approximation of the adjacency matrix. The embedded vertices are represented in a low dimensional space. This method is presented in Algorithm 1.

Algorithm 1 The adjacency spectral embedding method for undirected graphs

Input: The adjacency matrix $A \in \{0,1\}^{n\times n}$. The embedding dimension $\hat{d}$.

Output: Approximated latent positions $\hat{\mathcal{X}} \in \mathbb{R}^{n\times\hat{d}}$

Step 1: Spectral decomposition on A. Compute the first $\hat{d}$ eigen-pairs of A, denoted by $(U_A, S_A) \in \mathbb{R}^{n\times\hat{d}} \times \mathbb{R}^{\hat{d}}$, where S_A has $\hat{d}$ largest eigenvalues in magnitude sorted in non-increasing order.

Step 2: Denote the $\hat{d}$-dimensional coordinate-scaled singular vector matrix of A be $\hat{\mathcal{X}} = U_A S_A^{1/2} \in \mathbb{R}^{n\times\hat{d}}$.

For stochastic blockmodels with K blocks and a known model dimension d, Sussman et al. [84] show that partitioning on adjacency spectral embedding consistently estimates the block memberships of the stochastic blockmodel. When the model dimension d is not known, Fishkind et al. [41] show that embedding the adjacency matrix to $\hat{d} > d$-dimension is also consistent.

An example of adjacency spectral embedding is shown in Figure 2.1. Given the model dimension d, the resulting embedding $\hat{\mathcal{X}} \in \mathbb{R}^{n\times d}$ consistently estimates the latent positions X.

Besides embedding the adjacency matrix A for consistently estimating the block memberships, in [78], the consistency property of Laplacian spectral embedding is proved, and this yields the same algorithm as in Algorithm 1 except that the input matrix A becomes the Laplacian matrix $L := D - A$, where D is the degree matrix. Both methods require the knowledge of the embedding dimension d. However, in

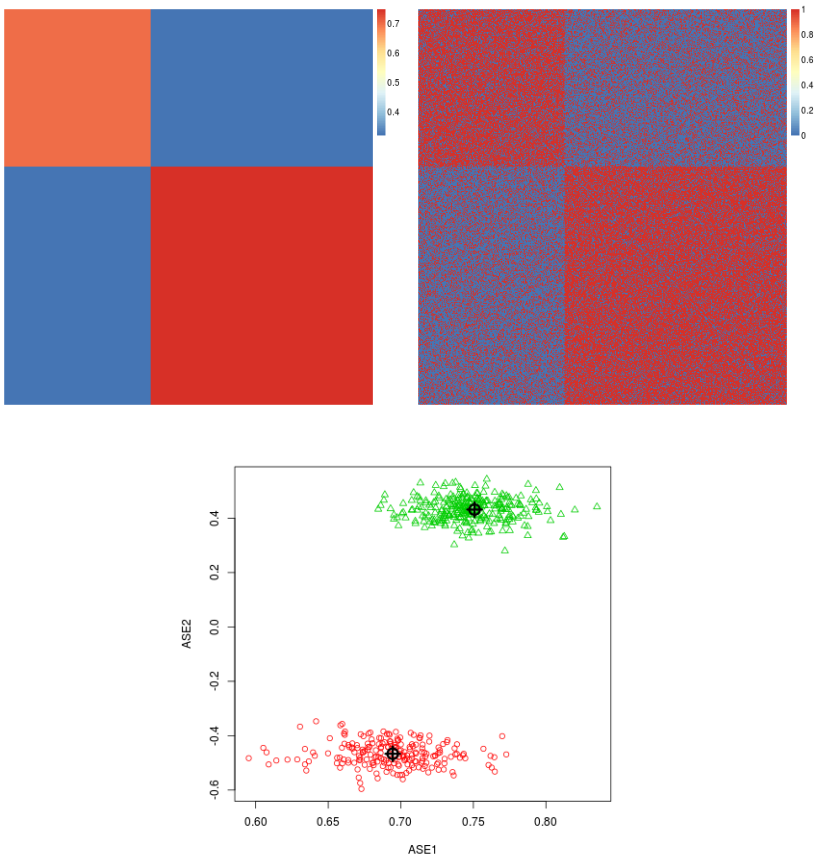


Figure 2.1: An example of adjacency spectral embedding. We simulate a stochastic blockmodel with 200 vertices and parameters specified in Equation 3.37. (Top left): The communication probability matrix P of a stochastic blockmodel. (Top right): The adjacency matrix drawn from the P. (Bottom middle): Adjacency spectral embedding represents the vertices into 2-dimensional Euclidean space. The vertices from two different blocks are well-separated in the Euclidean space. The black dots are the two eigen-vectors of B. The latent positions of the stochastic blockmodel here are mixture of these two points.

practice, the model dimension is often unknown. Indeed, for finite samples, when d is unknown and under data contamination, adjacency spectral embedding fails to consistently estimates the latent positions. ASE is widely used in graph inference. In Chapter 3, we will present a vertex classification framework, which does not require the knowledge of the model dimensions. This framework is robust to data contamination whose effect results in a changed model dimension. In Chapter 4, our proposed vertex clustering algorithms are motivated by adjacency spectral embedding techniques. In Chapter 5, one of our proposed vertex nomination schemes – the spectral partitioning vertex nomination scheme – also incorporates adjacency spectral embedding in its procedure.

Chapter 3

Robust Vertex Classification

One interesting graph inference task is vertex classification, which is to determine the class label of the vertices. For example, we may wish to classify whether a neuron is a motor neuron or an inter-neuron or whether a person in a social network is liberal or conservative. In many applications, measured edge activity can be inaccurate. This inaccuracy may be caused by either missing edges or wrongly-recorded edges, which leads to contaminated datasets. When the edge activities among a collection of vertices are not visible, occlusion contamination occurs. When the edge activities among a collection of vertices are wrongly observed, linkage reversion contamination occurs. In this chapter, we propose a sparse representation vertex classifier for stochastic blockmodels. Our proposed classifier does not require knowledge of the embedding dimension, is more robust to model misspecification than spectral embedding-based vertex classifiers, and maintains good performance for empirical graph inference [17].

The adjacency spectral embedding introduced in Chapter 2.5 has been shown to be a valuable tool for performing inference on stochastic blockmodels ([85], [41], [84], [86]). One major issue is that the method assumes knowledge of model dimension d, which is often unknown in practice. Occlusion in graphs degrades performance of spectral embedding methods. Other contamination such as linkage reversion in graphs may also degrade the inference performance.

The sparse representation classifier was originally developed for face recognition ([95], [96]), and has exhibited robustness to occlusion contamination. We extend and modify the sparse representation classifier for vertex classification. Our classifier maintains low misclassification error under both occlusion and linkage reversion contamination. We compare its performance with two spectral embedding-based vertex classifiers: applying classifiers (nearest neighbor and linear discriminant analysis) following adjacency spectral embedding. Our proposed method outperform the other two methods in real data inference.

This chapter is organized as follows. In Section 3.1, we introduce the framework of vertex classification. In Section 3.2, we describe the motivation for proposing a robust vertex classifier. In Section 3.3, we propose two contamination scenarios and prove several theoretical properties of the contaminated models. In Section 3.4, we examine the contamination effect on adjacency spectral embedding methodology. In Section 3.5, we present the sparse representation vertex classification algorithm. In Section 3.7, we demonstrate the effectiveness and robustness of the proposed classifier

on simulated and real data.

3.1 The Problem of Vertex Classification

In this section, we describe the classical setting of classification and extend this setting to vertex classification. Define the notation $[K] := \{1, ..., K\}$ for any positive integer K. Let $(X, Y) \sim F_{XY}$, where the feature vector X is an $\mathbb{R}^d$-valued random variable, Y is a $[K]$-valued class label, and F_{XY} is the unknown joint distribution of X and Y. Denote $\pi_k = \mathbb{P}(Y = k)$ as the class priors. Let $g : \mathbb{R}^d \to [K]$ be a classifier which maps d-dimensional data point X to the assigned label Y. The mapping g provides one's guess of Y given X. The task of classification intends to estimate the label Y of a test observation X via $g(X)$.

The probability of error is denoted by $L(g) = \mathbb{P}(g(X) \neq Y)$. The *Bayes* classifier g^*, is defined by $g^* = \arg\min_{g:\mathbb{R}^d \to [K]} \mathbb{P}(g(X) \neq Y)$. The *Bayes* classifier is optimal, because it has the smallest possible probability of error L^*. In the classical setting of classification, we observe training data $\mathcal{T}_n = \{(X_1, Y_1), ..., (X_n, Y_n)\} \overset{iid}{\sim} F_{XY}$. Denote the classifier on the training data by g_n. The performance of g_n is measured by the conditional probability of error defined by

$$L_n = L(g_n) = \mathbb{P}(g_n(X; \mathcal{T}_n) \neq Y | \mathcal{T}_n).$$

A sequence of classifiers $\{g_n, n \geq 1\}$ is universally consistent if $\lim_{n\to\infty} L_n = L^*$ with probability 1 for any distribution F_{XY} [29].

In the setting of latent position graphs, we observe the adjacency matrix A of $\mathbb{G} = \mathbb{G}(X_1, ..., X_{n-1}, X)$ on n vertices, and the first $n-1$ vertices have known labels $(Y_1, ..., Y_{n-1})$. We do not and cannot observe the latent positions $X_1, ..., X_{n-1}, X$; otherwise, we are back in the classical setting of classification. Let v_X denote the test vertex associated with latent position vector X and unobserved class label Y. We desire to construct a classifier g so that misclassification error $\mathbb{P}(g(X; X_1, \ldots, X_n) \neq Y)$ is small.

Recent developments in vertex classification for latent position graphs have focused on the universal consistency of vertex classifiers. Sussman et al. [85] show that estimating $(X_1, ..., X_{n-1}, X)$ via adjacency spectral embedding, and then employing a k-nearest neighbor (kNN) classifier on the represented data is universally consistent. Tang et al. [86] show that a class of linear classifiers applied after adjacency spectral embedding are universally consistent for latent position graphs. Athreya et al. [6] show that the adjacency spectral embedding of stochastic blockmodels is distributed as a mixture of multivariate normal distributions. This would imply that a subsequent linear discriminant analysis (LDA) or quadratic discriminant analysis (QDA) [32] on the represented data of stochastic blockmodels are asymptotic Bayes plug-in classifiers.

3.2 Motivation

Inference methodologies, which are unduly affected by outliers, or rely heavily on model assumptions, often have poor performance in practice, as real data do not follow the same model assumptions. When the model dimension d is known, ASE_d consistently estimates the latent positions for RDPG [84]. When model assumptions do not hold, how well can vertex classifiers perform? An example of ASE_d, where vertices from two classes are well separated in the embedded space is seen in Figure 3.1. A subsequent nearest neighbor (NN) classifier on ASE_d is universally consistent for RDPG [85]. That means regardless of what distribution the latent positions are drawn from, $\text{NN}\circ\text{ASE}_d$ achieves the Bayes error L^* asymptotically. In particular, for stochastic blockmodels, $\text{1NN}\circ\text{ASE}_d$ is asymptotically Bayes optimal.

Athreya et al. [6] proved for $d = 1$-dimensional RDPG, $\hat{\mathcal{X}}_1$ via ASE_1 is distributed as a mixture of normal distributions as the number of vertices goes to infinity. They also proved that for a K-block and d-dimensional ($d \geq 2$) SBM, $\hat{\mathcal{X}}_d$ via ASE_d is distributed as a K mixture of d-variate normals in the limit. This implies that quadratic discriminant analysis (QDA) on $\hat{\mathcal{X}}_d$ achieves the Bayes error L^* asymptotically. If the covariance matrices of each of the d-variate normals are the same, linear discriminant analysis (LDA) achieves the Bayes error L^* asymptotically [33]. In the case of SBMs, the covariance matrices are of order $\frac{1}{n}$, so both QDA and LDA are asymptotically Bayes optimal, while LDA requires a fewer number of parameters to fit [1]. Hence in our analysis, we employ two classifiers $\text{1NN}\circ\text{ASE}_d$ and $\text{LDA}\circ\text{ASE}_d$ for vertex classi-

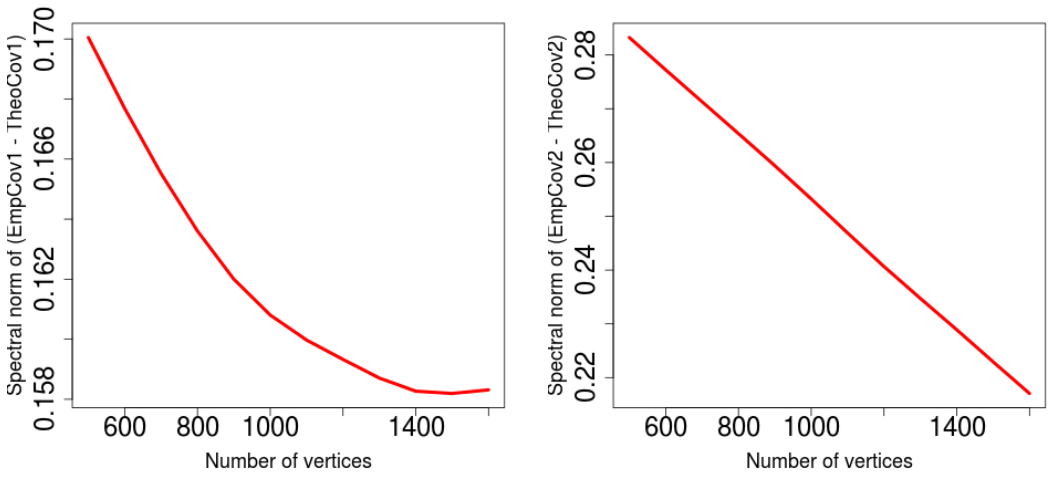


Figure 3.1: We simulate a stochastic blockmodel, whose parameters B and π are chosen the same as in Eq 3.37 with 500 vertices. (Left) Spectral norm difference between the theoretical covariance matrix and the empirical covariance matrix for Block 1. (Right) Spectral norm difference between the theoretical covariance matrix and the empirical covariance matrix for Block 2. We see that as the number of vertices n increases, the spectral norm differences between the empirical and the theoretical covariance matrix become smaller.

fication of stochastic blockmodels.

Importantly, having information on the model dimension d is critical to adjacency spectral approaches. When d is given, ASE_d is consistent, and $\text{1NN}\circ\text{ASE}_d$, $\text{LDA}\circ\text{ASE}_d$ asymptotically achieve the Bayes error. When d is not known, Sussman et al. [84] estimate d via a consistent estimator

$$\hat{d} = \max\{i : \sigma_i(A) > 3^{1/4} n^{3/4} \log^{1/4} n\}, \tag{3.1}$$

where $\sigma(A)$ is the singular value of A. However, the number of vertices necessary for the consistent estimator to be usable in practice will depend on the the sparsity of the graph. The number of vertices increases rapidly as the expected graph density decreases. Fishkind et al. [41] showed that ASE_R is consistent as $n \to \infty$, if a positive integer R is known with $d \leq R$. However, for a finite number of samples, $\text{1NN}\circ\text{ASE}_R$ and $\text{LDA}\circ\text{ASE}_R$ degrade significantly in performance compared to $\text{1NN}\circ\text{ASE}_d$ and $\text{LDA}\circ\text{ASE}_d$. Moreover, embedding to the wrong dimension, especially $\hat{d} < d$ degrades the classification performance greatly. Here we focus on developing a method which does not critically rely on the model dimension d compared to adjacency spectral embedding approaches, while still maintaining low misclassification error. Such a vertex classification procedure is robust to model misspecification and suitable for real graph data inference.

3.3 Two Contamination Models

We propose two scenarios of contamination procedures that change the dimension of the uncontaminated model. For the rest of the paper, we assume the uncontaminated graph model $\mathbb{G}_{\text{un}}$ is an element of the family of stochastic blockmodels $\mathcal{G} = \{\mathbb{G} \sim SBM([n], B, \pi)\}$. Hence, $\mathbb{G}_{\text{un}} \triangleq (B_{\text{un}}, \pi_{\text{un}}) \sim SBM([n], B_{\text{un}}, \pi_{\text{un}})$.

3.3.1 Contamination I: the Occlusion Model

Let $p_o \in [0, 1]$ denote the occlusion rate. We randomly select $100\%p_o$ vertices out of the n vertices and set the probability of connectivity among the selected vertices to be 0. In this scenario, the probability of connectivity between the contaminated vertices and the uncontaminated vertices remains the same as in $\mathbb{G}_{\text{un}}$.

3.3.1.1 The Contamination Procedure

Let $\epsilon = (1 - p_o, p_o)^T \in \mathbb{R}^2$ be the contamination proportion vector and p_o the contamination proportion, in this case, the occlusion rate. We randomly select $100\%p_o$ vertices out of the n vertices and set probability of connectivities among the selected vertices to be 0. In this scenario, the probability of connectivity between the contaminated vertices and the uncontaminated vertices remains the same as in $\mathbb{G}_{\text{un}}$. Figure

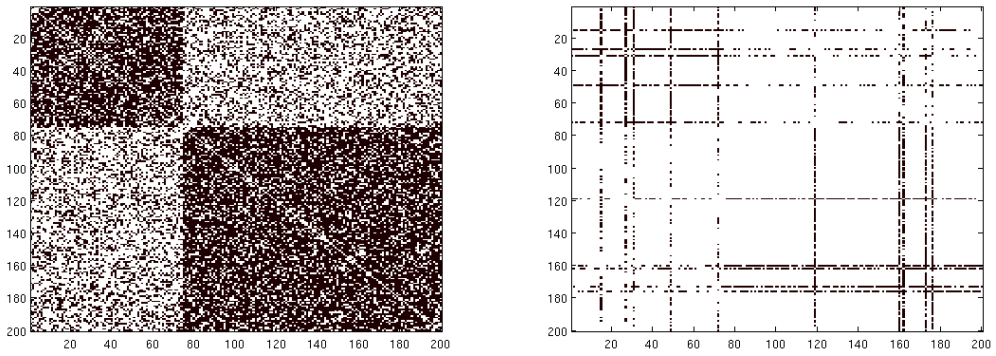


Figure 3.2: An example of the occlusion contamination. (Left) Original model. (Right) Occlusion contamination.

3.13 presents an example of the occlusion contamination. This occlusion procedure

can be formulated as a stochastic block model $\mathbb{G}_{\text{occ}}$ with the following parameters:

$$B_{\text{occ}} = \begin{pmatrix} B_{\text{un}} & B_{\text{un}} \\ B_{\text{un}} & 0_{K\times K} \end{pmatrix} \in \mathbb{R}^{2K\times 2K}, \tag{3.2}$$

$$\pi_{\text{occ}} = [(1-p_o)\pi_{\text{un}}^T,\ p_o\pi_{\text{un}}^T]^T \in \mathbb{R}^{2K}. \tag{3.3}$$

The operator $\otimes$ is the Kronecker product. The number of memberships in the contaminated model $\mathbb{G}_{\mathbf{occ}}$ rises to $2K$, where K memberships correspond to the class prior $(1-p_o)\pi_{\text{un}}$ and K memberships correspond to the class prior $p_o\pi_{\text{un}}$. If $\text{rank}(B_{\text{un}}) = d$, then $\text{rank}(B_{\text{occ}}) = 2d$. This is due to its Gaussian elimination form

$$B_{\text{occ}} \sim \begin{pmatrix} B_{\text{un}} & B_{\text{un}} \\ 0_{K\times K} & B_{\text{un}} \end{pmatrix}. \tag{3.4}$$

Furthermore,

$$\begin{pmatrix} B_{\text{un}} & B_{\text{un}} \\ B_{\text{un}} & 0_{K\times K} \end{pmatrix} = \begin{pmatrix} 1_{K\times K} & 0_{K\times K} \\ 1_{K\times K} & 1_{K\times K} \end{pmatrix} \begin{pmatrix} B_{\text{un}} & 0_{K\times K} \\ 0_{K\times K} & -B_{\text{un}} \end{pmatrix} \begin{pmatrix} 1_{K\times K} & 0_{K\times K} \\ 1_{K\times K} & 1_{K\times K} \end{pmatrix}^T. \tag{3.5}$$

This shows that B_{un} is congruent to the symmetric matrix $S := \begin{pmatrix} B_{\text{un}} & 0_{K\times K} \\ 0_{K\times K} & -B_{\text{un}} \end{pmatrix}$. In the next section, we show these two real congruent symmetric matrices have the same numbers of positive, negative and zero eigenvalues. If B is positive semi-definite, then B has d positive eigenvalues. Hence, B_{occ} has d positive eigenvalues and d negative eigenvalues, where the negative eigenvalues are due to contamination. Then the largest $2d$ eigenvalues in magnitude of the probability matrix $P_{\text{occ}} \in \mathbb{R}^{n\times n}$ has

exactly d positive eigenvalues and d negative eigenvalues. For small choices of p_o, as p_o increases, the d negative eigenvalues grow in magnitude. As p_o continues to increase to 1, all eigenvalues get closer to 0, and the size of contaminated vertices approaches n, indicating that the majority of the edges are sampled from the contamination source $0_{K\times K}$. As a result, the adjacency matrix A becomes sparser and sparser and eventually all zeros.

3.3.1.2 Theoretical Results on the Occlusion Stochastic Block-model

We now present several theoretical properties of the occlusion stochastic block-model. Suppose $\text{rank}(B_{\text{un}}) = d$. Let $\nu \in \mathbb{R}^{K\times d}$ such that $B_{\text{un}} = \nu\nu^T$. Let $\sigma_i(M)$ be the i-th largest singular value of a matrix M and let $\lambda_i(\tilde{M})$ be the i-th largest eigenvalue of a square matrix $\tilde{M}$. Let n_i be the size of Block i. We define the following constants which are not dependent on n:

- $\alpha > 0$ such that $\alpha \leq \min \lambda_i(\nu\nu^T)$
- $\beta > 0$ such that $\beta \leq \min_{i\neq j} \|\nu_i - \nu_j\|$
- $\gamma > 0$ such that $\gamma \leq \min_{i\in[K]} \pi_{\text{un},i}$.

Note that an event occurs "almost always", if with probability 1, the event occurs for all but finitely many $n \in \{1, 2, \ldots, n\}$.

Theorem 3.1. It always holds that $\sigma_1(P_{occ}) \leq \sigma_1(P_{\text{un}}) \leq n$.

Proof. Suppose the set of the contaminated vertices is $\mathcal{I} := \{i_1, i_2, \ldots, i_l\}$. Let P_s' denote the principal submatrix of $P_{\text{un}} \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|}$ obtained by deleting the $V\backslash\mathcal{I}$ columns and the corresponding $V \setminus \mathcal{I}$ rows. P_s' is symmetric.

Note that $P_{\text{un}} = P_{\text{occ}} + P_s$, where P_s is symmetric, $P_s = P_s'$ at $\{i_1, i_2, \ldots, i_l\}$-th columns and $\{i_1, i_2, \ldots, i_l\}$-th rows, and $P_s = 0$ everywhere else. By Weyl's Theorem [51], $\sigma_1(P_{\text{occ}}) + \min_\sigma \sigma(P_s) \leq \sigma_1(P_{\text{occ}} + P_s) = \sigma_1(P_{\text{un}})$. Thus, $\sigma_1(P_{\text{occ}}) \leq \sigma_1(P_{\text{un}})$.

Since $P_{\text{un}} \in [0,1]^n$, $P_{\text{un}}P_{\text{un}}^T = P_{\text{un}}P_{\text{un}}$ is a non-negative and symmetric matrix with entries bounded by n. Then each row sum is bounded by n^2. Thus, $\sigma_1^2(P_{\text{un}}) = \sigma_1(P_{\text{un}}^2) = \sigma(P_{\text{un}}P_{\text{un}}^T) \leq n^2$, giving $\sigma_1(P_{\text{un}}) \leq n$.

□

Theorem 3.2. It always holds that $\sigma_{2d+1}(P_{occ}) = 0$. It almost always holds that $\sigma_{2d}(P_{occ}) \geq \min(p_0, 1-p_0)\alpha\gamma n$. $\text{rank}(P_{occ}) = 2d$.

Proof. The Guassian elimination of B_{occ} is given by

$$B_{\text{occ}} \sim \begin{pmatrix} B_{\text{un}} & B_{\text{un}} \\ 0_{K\times K} & B_{\text{un}} \end{pmatrix}. \tag{3.6}$$

Since $\text{rank}(B_{\text{un}}) = d$, $\text{rank}(B_{\text{occ}}) = 2d$. Then there exist $\mu = \begin{pmatrix} \nu & 0_{K\times K} \\ \nu & -\nu \end{pmatrix} \in \mathbb{R}^{2K\times 2d}$ and $\tilde{\mu} = \begin{pmatrix} \nu & 0_{K\times K} \\ \nu & \nu \end{pmatrix} \in \mathbb{R}^{2K\times 2d}$ such that $B_{\text{occ}} = \mu\tilde{\mu}^T$. Let $\mathcal{X}_{\text{occ}} \in \mathbb{R}^{n\times 2d}$ and $\tilde{\mathcal{X}}_{\text{occ}} \in \mathbb{R}^{n\times 2d}$ with row u given by $\tilde{\mathcal{X}}_{\text{occ},u} = \tilde{\mu}_{Y_u}$. By the parametrization of SBM as RDPG model, $P_{\text{occ}} = \mathcal{X}_{\text{occ}}\tilde{\mathcal{X}}_{\text{occ}}^T$. Since $\mathcal{X}_{\text{occ}}, \tilde{\mathcal{X}}_{\text{occ}}$ are at most rank $2d$, then $\sigma_{2d+1}(P_{\text{occ}}) = 0$.

Since the following holds:

$$\mu\mu^T = \tilde{\mu}\tilde{\mu}^T = \begin{pmatrix} \nu\nu^T & \nu\nu^T \\ \nu\nu^T & 2\nu\nu^T \end{pmatrix} = \begin{pmatrix} \nu\nu^T & \nu\nu^T \\ \nu\nu^T & \nu\nu^T \end{pmatrix} + \begin{pmatrix} 0_{K\times K} & 0_{K\times K} \\ 0_{K\times K} & \nu\nu^T \end{pmatrix}, \quad (3.7)$$

by Weyl's theorem [51],

$$\min \lambda_i(\mu\mu^T) = \min \lambda_i(\tilde{\mu}\tilde{\mu}^T) \geq \min \lambda_i \begin{pmatrix} \nu\nu^T & \nu\nu^T \\ \nu\nu^T & \nu\nu^T \end{pmatrix} + \min \lambda_i \begin{pmatrix} 0_{K\times K} & 0_{K\times K} \\ 0_{K\times K} & \nu\nu^T \end{pmatrix} \geq \gamma + 0 = \gamma. \quad (3.8)$$

Moreover, we have

$$\min_{i\in[2K]} (\pi_{\text{occ},i}) = \min(p_o\pi_{\text{un},i}, (1-p_o)\pi_{\text{un},i}) \geq \min(p_o, 1-p_o)\gamma. \quad (3.9)$$

The eigenvalues of $P_{\text{occ}}P_{\text{occ}}^T$ are the same as the nonzero eigenvalues of $\tilde{\mathcal{X}}_{\text{occ}}^T\tilde{\mathcal{X}}_{\text{occ}}\mathcal{X}_{\text{occ}}^T\mathcal{X}_{\text{occ}}$.

it almost always holds that $n_i \geq \min(p_o, 1-p_0)\gamma n$ for all $i \in [2K]$ so that

$$\mathcal{X}_{\text{occ}}^T\mathcal{X}_{\text{occ}} = \sum_{i=1}^{2K} n_i\mu_i\mu_i^T = \min(p_o, 1-p_o)\gamma n\mu^T\mu + \sum_{i=1}^{2K}(n_i - \min(p_o, 1-p_o)\gamma n)\mu_i\mu_i^T. \quad (3.10)$$

The first term has $\min \lambda_i$ bounded below by $\alpha \min(p_o, 1-p_o)\gamma$. This means $\lambda_{2d}(\mathcal{X}_{\text{occ}}^T\mathcal{X}_{\text{occ}}) \geq \alpha \min(p_o, 1-p_o)\gamma$. For the exact same argument, $\lambda_{2d}(\tilde{\mathcal{X}}_{\text{occ}}^T\tilde{\mathcal{X}}_{\text{occ}}) \geq \alpha \min(p_o, 1-p_o)\gamma$.

$\tilde{\mathcal{X}}_{\text{occ}}^T\tilde{\mathcal{X}}_{\text{occ}}\mathcal{X}_{\text{occ}}^T\mathcal{X}_{\text{occ}}$ is the product of two positive semi-definite matrices. Then,

$$\lambda_{2d}(\tilde{\mathcal{X}}_{\text{occ}}^T\tilde{\mathcal{X}}_{\text{occ}}Z_{\text{occ}}^TZ_{\text{occ}}) \geq \lambda_{2d}(\tilde{\mathcal{X}}_{\text{occ}}^T\tilde{\mathcal{X}}_{\text{occ}})\lambda_{2d}(\mathcal{X}_{\text{occ}}^T\mathcal{X}_{\text{occ}}) \geq (\alpha \min(p_0, 1-p_0)\gamma n)^2. \quad (3.11)$$

This gives

$$\lambda_{2d}(\mathbf{P}_{\text{occ}}) \geq \alpha \min(p_0, 1-p_0)\gamma n = \min(p_0, 1-p_0)\alpha\gamma n. \quad (3.12)$$

Since $\lambda_{2d}(P_{\text{occ}}) \geq 0$ almost always and $\sigma_{2d+1}(P_{\text{occ}}) = 0$ always, then $\text{rank}(P_{\text{occ}}) = d$. □

Theorem 3.3. B_{occ} has d positive eigenvalues and d negative eigenvalues.

Proof. Let the consider the eigen-decomposition of B_{un} given by $\Xi\Psi\Xi^T$, where $\Xi \in \mathbb{R}^{K\times K}$ is orthogonal and $\Psi = \text{Diag}(\psi_1, ..., \psi_k) \in \mathbb{R}^{K\times K}$ is diagonal. We have the following congruent relation:

$$\begin{aligned}\begin{pmatrix} B_{\text{un}} & B_{\text{un}} \\ B_{\text{un}} & 0_{K\times K} \end{pmatrix} &= \begin{pmatrix} I_{K\times K} & 0_{K\times K} \\ I_{K\times K} & I_{K\times K} \end{pmatrix} \begin{pmatrix} B_{\text{un}} & 0_{K\times K} \\ 0_{K\times K} & -B_{\text{un}} \end{pmatrix} \begin{pmatrix} I_{K\times K} & 0_{K\times K} \\ I_{K\times K} & I_{K\times K} \end{pmatrix}^T \\ &= \begin{pmatrix} \Xi & 0_{K\times K} \\ \Xi & \Xi \end{pmatrix} \begin{pmatrix} \Psi & 0_{K\times K} \\ 0_{K\times K} & -\Psi \end{pmatrix} \begin{pmatrix} \Xi & 0_{K\times K} \\ \Xi & \Xi \end{pmatrix}^T . \end{aligned} \quad (3.13)$$

Hence, B_{occ} and $\begin{pmatrix} \Psi & 0_{K\times K} \\ 0_{K\times K} & -\Psi \end{pmatrix}$ are congruent. By Sylvester's law of Inertia [51], they have the same number of positive, negative and zero eigenvalues. Ψ has d positive diagonal entries since $\text{rank}(B_{\text{un}}) = d$. Similarly, $-\Psi$ has d negative diagonal entries. Hence, B_{occ} has d positive eigenvalues and d negative eigenvalues. □

Lemma 3.1. Let $M_1 \in \mathbb{R}^{n\times m}$ and $M_2 \in \mathbb{R}^{m\times n}$ be two matrices. The number of nonzero eigenvalues of M_1M_2 equals the number of nonzero eigenvalues of M_2M_1.

Proof. Let λ be a nonzero eigenvalue of M_1M_2. If v is an eigenvector of M_1M_2 then $M_1M_2v = \lambda v \neq 0$. Then $M_2M_1M_2v = \lambda M_2v \neq 0$. Thus, λ is also a nonzero eigenvalue for M_2M_1. By symmetry, if λ is a nonzero eigenvalue of M_2M_1, then λ

is also a nonzero eigenvalue of M_1M_2. Thus, the number of nonzero eigenvalues of M_1M_2 equals the number of nonzero eigenvalues of M_2M_1. □

Theorem 3.4. Assuming $|\lambda_1(P_{\text{occ}})| \geq |\lambda_2(P_{\text{occ}})| \geq \cdots \geq |\lambda_{2d}(P_{\text{occ}})|$, then $|\{i : \lambda_i(P_{\text{occ}}) < 0\}| = |\{i : \lambda_i(P_{\text{occ}}) < 0\}| = d$. That is, the number of positive eigenvalues of P_{occ} is the same as the number of negative eigenvalues of P_{occ}, and it equals d.

Proof. Let $Z \in \{0,1\}^{n\times 2K}$ denote the matrix, where each row i is of the form $(0,\ldots,1,0,\ldots,0)$, where 1 indicates the block membership of vertex i in the occlusion stochastic blockmodel. Then $P_{\text{occ}} = ZB_{\text{occ}}Z^T$. By Lemma 3.1, P_{occ} has the same number of nonzero eigenvalues as Z^TZB_{occ}. Let $D_Z := Z^TZ \in \mathbb{N}^{2K\times 2K}$ and note that D_Z is a diagonal matrix with nonnegative diagonal entries, where each diagonal entry denotes the number of vertices belonging to block $k \in [K]$. With high probability, D is positive definite, as the number of vertices in each block is positive. Then the number of nonzero eigenvalues of P_{occ} is the same as the number of nonzero eigenvalues of $Z^TZB_{\text{occ}} = DB_{\text{occ}} = \sqrt{D_Z}\sqrt{D_Z}B_{\text{occ}} = \sqrt{D_Z}B_{\text{occ}}\sqrt{D_Z}$. By Sylvester's law of Inertia [51], the number of positive eigenvalues of $\sqrt{D_Z}B_{\text{occ}}\sqrt{D_Z}$ is the same as the number of positive eigenvalues of B_{occ}, and the number of negative eigenvalues of $\sqrt{D_Z}B_{\text{occ}}\sqrt{D_Z}$ is the same as the number of negative eigenvalues of B_{occ}, thus proving our claim.

□

Next we examine, in the spectral embedded space, how distinct the rows of the latent positions are. Since P_{occ} has d negative eigenvalues that are among the $2d$ largest eigenvalues in magnitude, we consider the singular value decomposition (SVD) of P_{occ}, so that the eigenvalues are ordered in their absolute values. The SVD of P_{occ} is given by $U_{P_{\text{occ}}}\Sigma\tilde{U}^T_{P_{\text{occ}}}$. Denote the matrix consisting of the first $2d$ columns of $U_{P_{\text{occ}}}$ by $U_{P_{\text{occ}},2d}$. For a given matrix M, denote M_u as the u-th row of M.

Theorem 3.5. For all u, v such that $\mathcal{X}_{\text{occ},u} \neq \mathcal{X}_{\text{occ},v}$, it almost always holds that

$$\|U_{P_{\text{occ}},2d,u} - U_{P_{\text{occ}},2d,v}\| \geq \min(p_o, 1-p_o)^{3/2}\beta\sqrt{\alpha\gamma}/\sqrt{n}, \tag{3.14}$$

$$\|\tilde{U}_{P_{\text{occ}},2d,u} - \tilde{U}_{P_{\text{occ}},2d,v}\| \geq \min(p_o, 1-p_o)^{3/2}\beta\sqrt{\alpha\gamma}/\sqrt{n}. \tag{3.15}$$

Proof. Let $\tilde{\mathcal{X}}^T_{occ}\tilde{\mathcal{X}}_{occ} = M\Lambda M^2$, where $M \in \mathbb{R}^{2d\times 2d}$ is orthogonal and $\Lambda \in \mathbb{R}^{2d\times 2d}$ is diagonal. Define $\Phi = \mathcal{X}_{\text{occ}}M$, $\Theta = \Phi\Lambda$, $U' = U_{\text{occ}}\Sigma$. Let u, v be such that $\mathcal{X}_{\text{occ},u} \neq \mathcal{X}_{\text{occ},v}$.

We have the following equalities and inequalities,

$$\begin{aligned}
\|\Phi_u - \Phi_v\| &= \|\mathcal{X}_{\text{occ},u}M - \mathcal{X}_{\text{occ},v}M\| \\
&= \|(\mathcal{X}_{\text{occ},u} - \mathcal{X}_{\text{occ},v})M\| \\
&= \|\mathcal{X}_{\text{occ},u} - \mathcal{X}_{\text{occ},v}\|. \qquad (3.16) \\
\|U'_u - U'_v\| &= \|U_{P_{\text{occ}},2d,u}\Sigma - U_{P_{\text{occ}},2d,v}\Sigma\|
\end{aligned}$$

By Theorem 3.1,

$$\begin{aligned} \sigma_1(P_{\text{occ}}) &\leq n \\ \|U'_u - U'_v\| &\leq n\|U_{P_{\text{occ}},2d,u} - U_{P_{\text{occ}},2d,v}\|. \qquad (3.17) \\ \|\Theta_u - \Theta_v\| &= \|\Phi_u\Lambda - \Phi_v\Lambda\| \\ &= \|(\Phi_u - \Phi_v)\Lambda\| \qquad (3.18) \end{aligned}$$

By Theorem 3.2,

$$\begin{aligned} \min_i |\Lambda_{ii}| &\geq \alpha \min(p_o, 1-p_o)\gamma n, \\ \|\Theta_u - \Theta_v\| &\geq \sqrt{\alpha \min(p_o, 1-p_o)\gamma n}\|\Phi_u - \Phi_v\| \\ \Theta\Theta^T &= \Phi M^2 \Phi^T = \mathcal{X}_{\text{occ}} M \Lambda^2 M^T Z_{\text{occ}}^T = \mathcal{X}_{\text{occ}} \tilde{\mathcal{X}}_{\text{occ}}^T \tilde{\mathcal{X}}_{\text{occ}} \mathcal{X}_{\text{occ}} \\ &= U_{P_{\text{occ}},2d} \Sigma \tilde{U}_{P_{\text{occ}},2d}^T \tilde{U}_{P_{\text{occ}},2d} \Sigma U_{P_{\text{occ}},2d}^T = U_{P_{\text{occ}},2d} \Sigma^2 U_{P_{\text{occ}},2d}^T \\ &= U' U'^T. \qquad (3.19) \end{aligned}$$

Let $e = [0, 0, \ldots, -1, \ldots, 1, \ldots, 0] \in \mathbb{R}^n$ be a vector of all zeros except 1 at the u-th entry and -1 at the v-th entry. Then the following holds,

$$\begin{aligned} \|\Theta_u - \Theta_v\|^2 &= e^T \Theta\Theta^T e = e^T U' U'^T e = \|U'_u - U'_v\|^2 \\ \|\Theta_u - \Theta_v\| &= \|U'_u - U'_v\|. \qquad (3.20) \end{aligned}$$

Combining Eq 3.16 - 3.20, the following holds,

$$\begin{aligned}
\|\mathcal{X}_{\text{occ},u} - \mathcal{X}_{\text{occ},v}\| &= \|\Phi_u - \Phi_v\| \\
&\leq \frac{1}{\sqrt{\min(p_o, 1-p_o)\alpha\gamma n}}\|\Theta_u - \Theta_v\| \\
&= \frac{1}{\sqrt{\min(p_o, 1-p_o)\alpha\gamma n}}\|U'_u - U'_v\| \\
&\leq \frac{\sqrt{n}}{\sqrt{\min(p_o, 1-p_o)\alpha\gamma}}\|U_{P_{\text{occ}},2d,u} - U_{P_{\text{occ}},2d,v}\|. \qquad (3.21)
\end{aligned}$$

Since $\beta \leq \min_{i\neq j}\|\nu_i - \nu_j\|$, then $\min_{u\neq v}\|\mathcal{X}_u - \mathcal{X}_v\| \geq \beta$ and $\min_{u\neq v}\|\mathcal{X}_{\text{occ},u} - \mathcal{X}_{\text{occ},v}\| \geq \min(p_o, 1-p_o)\beta$. Hence, we have shown

$$\|U_{P_{\text{occ}},2d,u} - U_{P_{\text{occ}},2d,v}\| \geq \min(p_o, 1-p_o)^{3/2}\beta\sqrt{\alpha\gamma}/\sqrt{n}. \qquad (3.22)$$

For a symmetric argument, we can show

$$\|\tilde{U}_{P_{\text{occ}},2d,u} - \tilde{U}_{P_{\text{occ}},2d,v}\| \geq \min(p_o, 1-p_o)^{3/2}\beta\sqrt{\alpha\gamma}/\sqrt{n}. \qquad (3.23)$$

□

When there is no contamination, it holds that $\|U_{P,d,u} - U_{P,d,v}\| \geq \beta\sqrt{\alpha\gamma}/\sqrt{n}$, where U is the matrix of the eigenvectors of P [84]. Since $\min(p_o, 1-p_o) \leq 1$, after contamination, the rows of the spectral embedding of the occluded communication matrix are not as distinguishable, because the lower bound of the row difference becomes smaller.

Recall that the adjacency spectral embedding with model dimension d is given by $U_{A,d}S_{A,d}^{\frac{1}{2}}$. We prove an upper bound of how different the occluded adjacency

matrix A_{occ} is from the occluded probability matrix P_{occ} via spectral embedding with specified embedding dimension $2d$. We first cite a proposition from [84].

Proposition 3.1. It almost always holds that $\|A^2 - P^2\|_F \leq \sqrt{2n^3 \log n}$.

It follows immediately that $\|A^2_{\text{occ}} - P^2_{\text{occ}}\|_F \leq \sqrt{2n^3 \log n}$ holds almost always.

We first prove that the spectral embedding of the occluded adjacency matrix $U_{A_{\text{occ}},2d}$ is close to that of the occluded probability matrix $U_{P_{\text{occ}},2d}$.

Theorem 3.6. It almost always holds that there exists an orthogonal matrix $R \in \mathbb{R}^{2d \times 2d}$ such that

$$\|U_{A_{\text{occ}},2d}R - U_{P_{\text{occ}},2d}\|_F \leq \frac{\sqrt{6 \log n}}{\min(p_o, 1-p_o)^2 \alpha^2 \gamma^2 \sqrt{n}}. \tag{3.24}$$

Proof. Theorem 3.2 showed that the eigengap δ for $\mathbf{P}^2_{\text{occ}}$ is greater than $\min(p_o, 1 - p_o)^2 \alpha^2 \gamma^2 n^2$. The eigenvectors of A_{occ} and P_{occ} equal to the eigenvectors of A^2_{occ} and $\mathbf{P}^2_{\text{occ}}$. Using Proposition 3.1 and applying Davis-Kahan Theorem [28],

$$\begin{aligned}
\|U_{A_{\text{occ}},2d}R - U_{P_{\text{occ}},2d}\| &\leq \frac{\sqrt{2}}{\delta}\|A^2_{\text{occ}} - P^2_{\text{occ}}\|_F && (3.25)\\
&\leq \frac{\sqrt{2}}{\delta}\sqrt{3n^3 \log n} && (3.26)\\
&\leq \frac{\sqrt{6 \log n}}{\min(p_o, 1-p_o)^2 \alpha^2 \gamma^2 \sqrt{n}}. && (3.27)
\end{aligned}$$

□

It follows that the spectral embedding of the occluded adjacency matrix $U_{A_{\text{occ}},2d} S^{\frac{1}{2}}_{A_{\text{occ}},2d}$ is close to the spectral embedding of the occluded probability matrix $\mathcal{X}_{\text{occ}} = U_{P_{\text{occ}},2d} S^{\frac{1}{2}}_{P_{\text{occ}},2d}$.

That is, ASE_{2d} is close to the latent positions under occlusion. An easy adaption from Theorem 1 in [84] shows that it almost always holds that there an orthogonal matrix $R \in \mathbb{R}^{2d \times 2d}$ such that

$$\|U_{A_{\text{occ}},2d} S^{\frac{1}{2}}_{A_{\text{occ}},2d} R - \mathcal{X}_{\text{occ}}\|_F \leq \frac{4d}{\sqrt{\min(p_o, 1-p_o)}} \sqrt{\frac{3 \log n}{\alpha^3 \gamma^3 n}}. \tag{3.28}$$

Note that this bound in Equation 3.28 is weaker than the bound in [84] due to our add-in occlusion contamination.

3.3.2 Contamination II: The Linkage Reversion Model

Besides missing edge information, we could also be handling absolutely wrong edge information. This leads us to design a linkage reversion contamination procedure in order to assess whether error rate is maintained low when edge information is wrong.

Let $p_l \in [0, 1]$ denote the linkage reversion rate. We randomly select $100\% p_l$ vertices out of the n vertices and reverse the connectivity among all the selected vertices. The probability of connectivity between the contaminated vertices and the uncontaminated vertices remains the same as in $\mathbb{G}_{\text{un}}$. The linkage reversion contamination can be formulated as a stochastic blockmodel $\mathbb{G}_{\text{rev}}$ with the following parameters:

$$B_{\text{rev}} = \begin{pmatrix} B_{\text{un}} & B_{\text{un}} \\ B_{\text{un}} & J_{K \times K} - B_{\text{un}} \end{pmatrix} \in \mathbb{R}^{2K \times 2K}, \tag{3.29}$$

$$\pi_{\text{rev}} = [(1 - p_l)\pi^T_{\text{un}}, \ p_l \pi^T_{\text{un}}]^T \in \mathbb{R}^{2K}. \tag{3.30}$$

The matrix $J_{K\times K} \in \mathbb{R}^{K\times K}$ is of all ones. Denote the communication probability matrix of $\mathbb{G}_{\text{rev}}$ by P_{rev}. To see that the contamination source is $J_{K\times K} - B_{\text{un}}$, let E_1 denote the event that an edge exists between vertices v and w, E_2 the event that v and w are selected to be contaminated, and E_3 the event of no edge between v, w given $Y(v)$, $Y(w)$ as in the uncontaminated model $\mathbb{G}_{\text{un}}$. Then,

$$\begin{aligned} P(E_1|E_2) &= P(E_1|E_2|E_3)P(E_3) + P(E_1|E_2|E_3^c)P(E_3^c) \\ &= 1 \times P(E_3) + 0 \times P(E_3^c) \\ &= J_{K\times K} - B. \end{aligned} \tag{3.31}$$

If $\text{rank}(B_{\text{un}}) = d$, then it almost always holds that $d+1 \leq \text{rank}(B_{\text{rev}}) = \text{rank}(P_{\text{rev}}) \leq 2d$. The negative eigenvalues of P_{rev} are due to the linkage reversion contamination. The number of blocks in the contaminated model $\mathbb{G}_{\text{rev}}$ is $2K$, where K blocks correspond to $(1-p_l)\pi_{\text{un}}$ and K blocks correspond to $p_l\pi_{\text{un}}$. Although the number of blocks in the model changes to $2K$ due to contamination, the number of classes in the vertex classification problem remains K. Clearly, as $p_l \to 1$, we recover the complement of $\text{SBM}([n], B_{\text{un}}, \pi_{\text{un}})$– that is, $\text{SBM}([n], J_{K\times K} - B_{\text{un}}, \pi_{\text{un}})$.

3.4 The Contamination Effect

When a stochastic blockmodel is contaminated by the two procedures described in Section 3.3, the model parameters and the model dimension are changed. Suppose the original model dimension d and knowledge of the contamination are known,

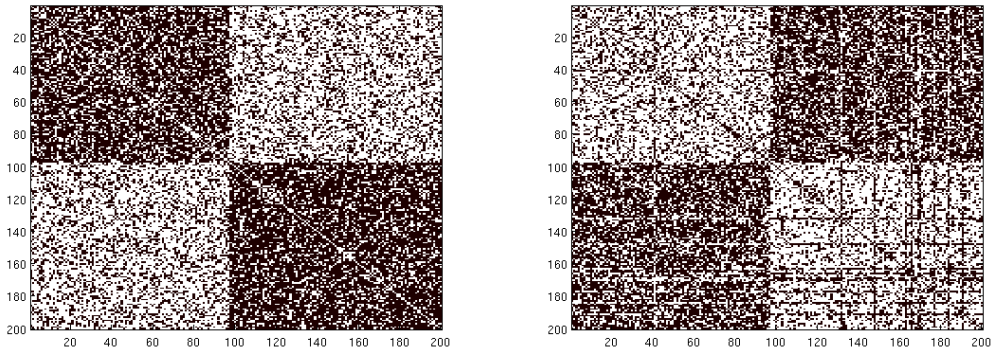


Figure 3.3: An example of the linkage reversion contamination. (Left) No contamination. (Right) Linkage reversion contamination.

we naturally know the contaminated model dimension d_{occ} or d_{rev}. Then $\text{ASE}_{d_{\text{occ}}}$ and $\text{ASE}_{d_{\text{rev}}}$ are consistent asymptotically. Subsequent 1NN is asymptotically Bayes optimal, and LDA will exhibit great performance. However, if we know only d but not the contamination procedure, then we will consider d as the default embedding dimension.

Let us look at an example of how contamination affects the spectral embedding and subsequently the estimation of latent positions. Figure 3.4 and Figure 3.5 show an example of the scree plots obtained from the contaminated adjacency matrices A_{occ} and A_{rev}. In this example, the model dimension is known to be $d = 2$. We inspect the scree plots based on the principle of statistical parsimony, and use an automatic elbow-selecting procedure based on profile likelihood proposed in [102]. The selected elbows suggest that choosing $\hat{d} = 2$ is reasonable in both cases of contamination. Despite the results in [84] and [41], we cannot be guaranteed to successfully choose

the embedding dimension in practice. Consequently, the performance of $\text{ASE}_{\hat{d}_{\text{occ}}}$ and $\text{ASE}_{\hat{d}_{\text{rev}}}$ will degrade.

Figure 3.6 and Figure 3.7 demonstrate that, as the contamination proportions p_o and p_l increase – that is, the data gets heavily contaminated, latent positions change as reflected in the estimated latent positions $\hat{\mathcal{X}}_{\hat{d}_{\text{occ}}}$ and $\hat{\mathcal{X}}_{\hat{d}_{\text{rev}}}$. In this plot, the embedding dimensions $\hat{d}_{\text{occ}}$ and $\hat{d}_{\text{rev}}$ are selected from the scree plots of A_{occ} and A_{rev} respectively using a profile likelihood method [102]. In particular, as the occlusion rate p_o increases, more vertices from different classes are embedded closer to each other. Subsequent vertex classification on the contaminated $\hat{\mathcal{X}}_{\hat{d}}$ using 1NN or LDA will degrade in performance. Indeed, knowing the model dimension is critical to the success of vertex classification using the ASE procedures, whereas in practice, the model dimension is often unknown. This motivates us to seek a robust vertex classifier which does not depend heavily on the model dimension, but still maintains good classification performance.

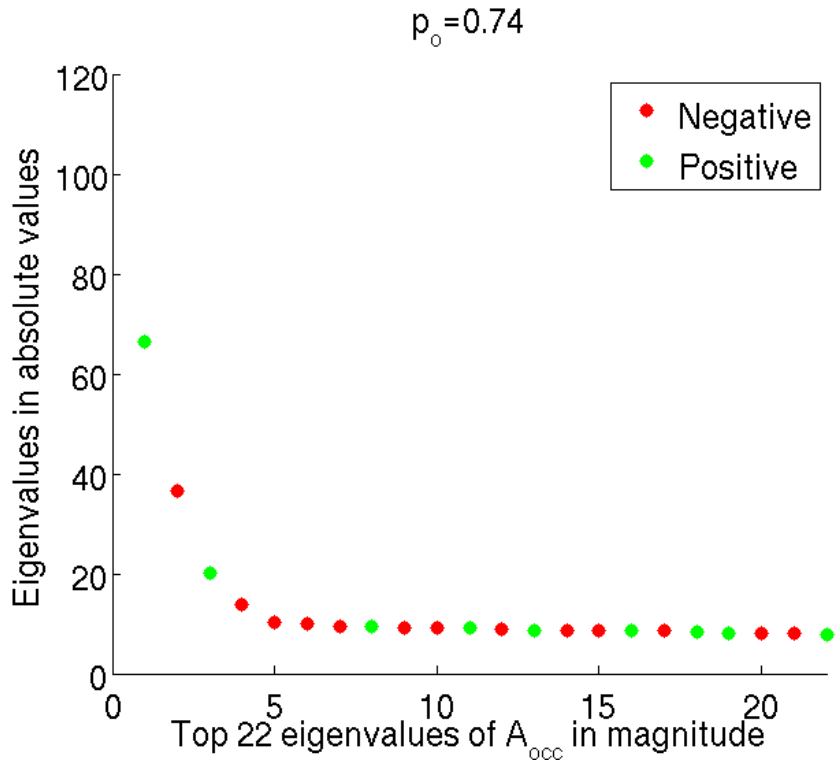


Figure 3.4: Scree plot of the occlusion contaminated adjacency matrix. Scree plot of the occlusion contaminated adjacency matrix A_{occ} at occlusion rate $p_o = 0.74$ with $n = 200$. The parameters B_{un} and π_{un} are given in Equation 3.37. The red dots are the negative eigenvalues of A_{occ} due to occlusion contamination. The green dots are the positive eigenvalues of A_{occ}. If no information about d or no knowledge of contamination is available, the principle of statistical parsimony suggests that choosing the estimated dimension $\hat{d} = 2$ is reasonable in this case [102].

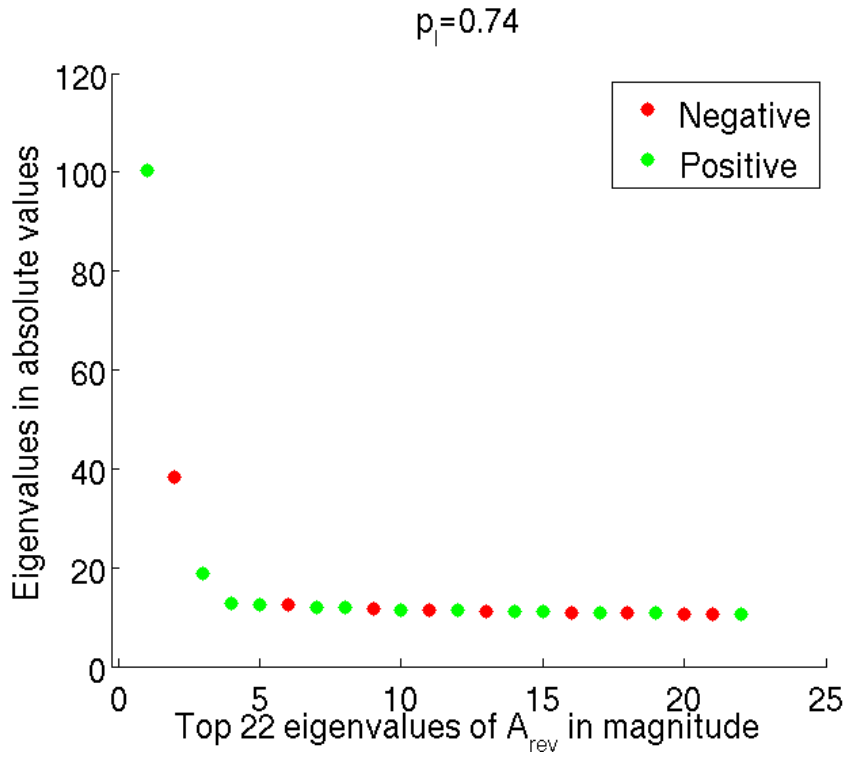


Figure 3.5: Scree plot of the linkage reversion contaminated adjacency matrix. Scree plot of the linkage reversion contaminated adjacency matrix A_{rev} at linkage reversion rate $p_l = 0.74$ with $n = 200$. The parameters B_{un} and π_{un} are given in Equation 3.37. The red dots are the negative eigenvalues of A_{rev} due to linkage reversion contamination. The green dots are the positive eigenvalues of A_{rev}. If no information about d or no knowledge of contamination is available, the principle of statistical parsimony suggests that choosing the estimated dimension $\hat{d} = 2$ is reasonable in this case [102].

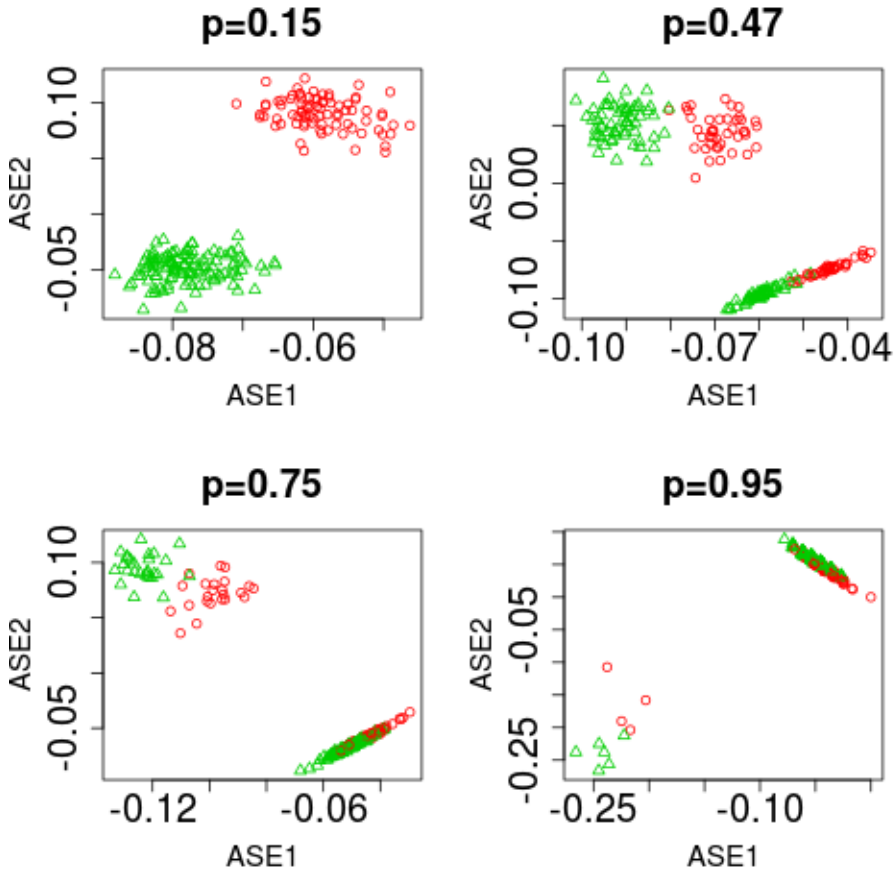


Figure 3.6: The occlusion contamination effect on estimated latent positions. A depiction of the occlusion effect on the latent positions as reflected in the estimated latent positions $\hat{\mathcal{X}}_{\hat{d}=2}$ with $n = 200$. The parameters B_{un} and π_{un} are given in Equation 3.37. The four-panel displays the contamination effect on latent position estimation for different increasing values of the occlusion rate p_o. As p_o increases, vertices from different blocks become close in the embedded space. For p_o close to 1, $\text{ASE}_{\hat{d}=2}$ will eventually yield only one cloud at 0. We see the obvious deleterious effects on classification.

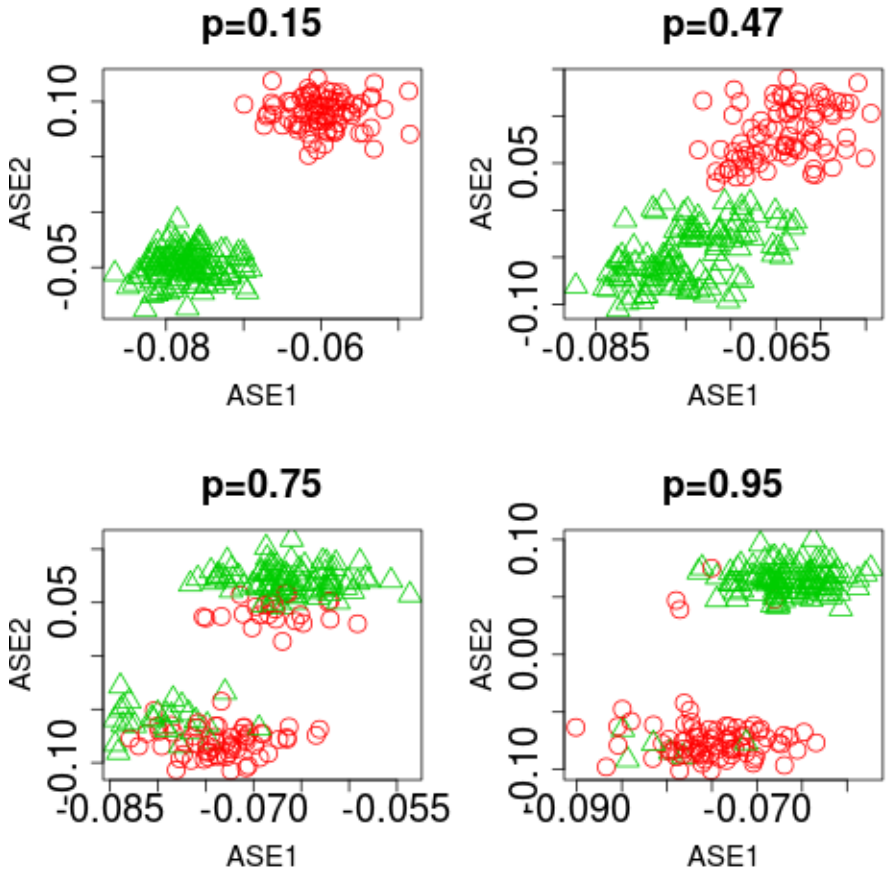


Figure 3.7: The linkage reversion contamination effect on estimated latent positions. A depiction of the linkage reversion effect on the latent positions as reflected in the estimated latent positions $\hat{\mathcal{X}}_{\hat{d}=2}$ with $n = 200$. The parameters B_{un} and π_{un} are given in Equation 3.37. The four-panel displays the contamination effect on latent position estimation for different increasing values of the linkage reversion rate p_l. As p_l increases, vertices from different blocks become close in the embedded space. For $p_l = 1$, $\text{ASE}_{\hat{d}=2}$ will yield two clouds corresponding to $\text{SBM}(200,\ J_{2\times 2} - B_{\text{un}},\ \pi_{\text{un}})$. We see the obvious deleterious effects on classification for some choices of p_l.

3.5 Sparse Representation Classifier

3.5.1 The Algorithm

In this section, we present our proposed sparse representation classifier (SRC) for vertex classification, and explain the main steps. Instead of employing adjacency spectral embedding, and then applying classifiers such as the linear discriminant analysis or the nearest neighbor classifier on the embedding representation $\hat{\mathcal{X}}_{\hat{d}}$, we recover a sparse linear combination of a test vertex expressed in terms of the vertices in the training set, and use the recovered sparse representation coefficients to classify the test vertex.

The setting for SRC is formulated as follows. We observe the adjacency matrix A on n vertices $\{v_1, \ldots, v_{n-1}, v\}$, where the first $n-1$ vertices are associated with known vertex labels $Y_i \in [K]$ out of K classes. Suppose there are n_k training vertices in each class $k \in [K]$, so that the size $n-1$ of the training set is given by $n-1 = \sum_{k\in[K]} n_k$. Let $a_{k,1}, ..., a_{k,n_k}$ denote the columns in A corresponding to the n_k training vertices in class k. Define $A_k := [a_{k,1}, ..., a_{k,n_k}] \in \{0,1\}^{(n-1)\times n_k}$ and normalize each column of A_k to have unit L_2 norm, i.e., $d_{k,j} = \frac{a_{k,j}}{\|a_{k,j}\|_2}$ for $1 \leq j \leq n_k$. The notations of $A_1, \ldots, A_K$ are now normalized. We concatenate $A_1, \ldots, A_K$ such that $D := [A_1, \ldots, A_K] \in \mathbb{R}_{>0}^{(n-1)\times(n-1)}$. The matrix D is called the dictionary. We present SRC in Algorithm 2.

Algorithm 2 Robust vertex classification.

Goal: Classify the vertex v.

Input: Adjacency matrix $A \in \{0,1\}^{n \times n}$ on vertices $\{v_1, \ldots, v_{n-1}, v\}$, where the first n vertices are associated with observed labels $Y_i \in [K]$. Let $\phi \in \{0,1\}^{n-1}$ be the n-th row (or equivalently column) of A, removing the zero in the n-th coordinate.

1. **Arrange the training vertices**: Let $a_{k,1}, ..., a_{k,n_k}$ denote the columns in A corresponding to the n_k training vertices in class k. Define $A_k := [a_{k,1}, ..., a_{k,n_k}] \in \mathbb{R}^{(n-1) \times n_k}$ and normalize each column of A_k to have unit L_2 norm, i.e., $d_{k,j} = \frac{a_{k,j}}{\|a_{k,j}\|_2}$ for $1 \leq j \leq n_k$. We concatenate $A_1, \ldots, A_K$ such that $D := [A_1, \ldots, A_K] \in \mathbb{R}^{(n-1) \times (n-1)}$.

2. **Solve the minimization problem:**

$$L_0 - \text{minimization: } \hat{x} = \underset{x}{\operatorname{argmin}} \|x\|_0 \text{ subject to } \|\phi - Dx\|_2 \leq \epsilon,$$

or

$$L_1 - \text{minimization: } \hat{x} = \underset{x}{\operatorname{argmin}} \|x\|_1 \text{ subject to } \|\phi - Dx\|_2 \leq \epsilon.$$

3. **Compute the distance of ϕ to each class k:** $r_k(\phi) = \|\phi - D\hat{x}_k\|_2$, where $\hat{x}_k = [0, ..., 0, \hat{x}_{k,1}, ..., \hat{x}_{k,n_k}, ..., 0]^T \in \mathbb{R}^{n-1}$ is the recovered coefficients corresponding to the k-th class.

4. **Classify test vertex**: $\hat{Y} = \operatorname{argmin}_k r_k(\phi)$.

3.5.2 The Test Vertex Expressed as a Sparse Representation of the Training Vertices

We remove the test vertex's connectivity with itself. Let $\phi \in \{0,1\}^{n-1}$ be the n-th row (or column) of A, removing the zero in the n-th coordinate. Hence, the vector ϕ is represented in terms of its connectivity to the training vertices.

3.5.3 L_0 or L_1 Minimization

Formally, the optimization problem is set up as

$$L_0 - \text{minimization: } \underset{x}{\operatorname{argmin}} \|x\|_0 \text{ subject to } \|\phi - Dx\|_2 \leq \epsilon, \tag{3.32}$$

where $\|\cdot\|_0$ equals the number of nonzero entries or sparsity level of x. Directly solving the L_0 minimization problem is NP-hard [67], that is, the complexity of the search in L_0 minimization grows exponentially with n. One may solve the L_0 minimization problem in a greedy manner (see, e.g., [91], [26], [36]). The coefficient vector x can also be solved via the L_1 minimization problem

$$L_1 - \text{minimization: } \underset{x}{\operatorname{argmin}} \|x\|_1 \text{ subject to } \|\phi - Dx\|_2 \leq \epsilon. \tag{3.33}$$

See [30], [31] [46], [13], [35], [79] and [90] for the equivalence of solving L_0 and L_1 minimization problems. The model consistency of the LASSO in sparse recovery is proved in [101]. The augmented Lagrangian multiplier technique for solving the L_1

minimization problem is studied in [97].

In this work, we use the orthogonal matching pursuit (OMP), a greedy approximation of L_1 minimization. OMP is an iterative greedy algorithm that selects the columns most correlated with the residual in order to find the sparsest solution $\hat{x}$ [71], [91], [12], [54]. The major advantage of OMP in solving sparse recovery problem is that the algorithm is simple to implement and fast to compute [90]. This algorithm is presented in Algorithm 3.

Algorithm 3 Orthogonal Matching Pursuit

Input: Training vertices $D \in \mathbb{R}^{(n-1)\times(n-1)}$, test vertex ϕ, a specified sparsity level s and/or a specified tolerance error ϵ.

Output: recovered sparse coefficient $\hat{x}$ with $\|\hat{x}\|_0 = s$.

Initialize the residual $r_0 = \phi$, iteration $t = 1$, index set $\Lambda_0 = \emptyset$.

while $t < s$ or $r_t < \epsilon$ **do**

Step 1: Find the index i_t such that $i_t = \text{argmax}_{j=1,2,\ldots,n-1} |\langle i_{t-1}, d_{\cdot,j}\rangle|$. If there are multiple i_t's, break the tie deterministically.

Step 2: $\Lambda_t \leftarrow \Lambda_{t-1} \cup \{i_t\}$. Denote the corresponding submatrix as $D_t \leftarrow [D_{t-1} d_{\cdot,i_t}]$.

Step 3: Solve for the least-square problem $\hat{x}_t = \text{argmin}_x \|D_t x - \phi\|_2$, and update the residual $r_t = \phi - D_t \hat{x}_t$.

end while

One can also use other L_0 greedy methods or L_1 solvers [26, 36] to solve the

minimization problem in Algorithm 2.

3.5.4 Classification

Let $\hat{x}$ be the recovered sparse vector of coefficients solved by Equation 3.32 or Equation 3.33. We also express $\hat{x}$ as

$$\hat{x} = \sum_{k=1}^{K} \hat{x}_k, \tag{3.34}$$

where each $\hat{x}_k = [0, ..., 0, \hat{x}_{k,1}, ..., \hat{x}_{k,n_k}, ..., 0]^T \in \mathbb{R}^{n-1}$ is the recovered coefficients corresponding to class k. We compute the distance between ϕ and the recovered vector $D\hat{x}_k$ in the class k by

$$r_k(\phi) = \|\phi - D\hat{x}_k\|_2. \tag{3.35}$$

The label Y of the test vertex v is estimated using the minimum L_2 distance

$$\hat{Y} = \text{argmin}_k r_k(\phi). \tag{3.36}$$

3.6 Robustness of Sparse Representation Classifier for Vertex Classification

When the true model dimension d is unknown, $\text{ASE}_{\hat{d}}$ may not be consistent. When contamination results in a changed model dimension, the performance of subsequent classifiers composed with $\text{ASE}_{\hat{d}}$ could degrade. We consider a classifier robust, if it

maintains relatively low error rate under data contamination. Our proposed sparse representation classifier (SRC) for vertex classification does not require knowing the embedding dimension, and SRC_s classification performance is much more stable with respect to sparsity levels s than $\text{1NN}\circ\text{ASE}_{\hat{d}}$ and $\text{LDA}\circ\text{ASE}_{\hat{d}}$ with respect to embedding dimensions $\hat{d}$.

3.7 Numerical Experiments

3.7.1 Simulation

3.7.1.1 No Contamination

We simulate the probability matrix for an uncontaminated stochastic blockmodel $\mathbb{G}_{\text{un}}$ with $K = 2$ blocks ($Y \in \{1, 2\}$) and parameters

$$B_{\text{un}} = \begin{pmatrix} 0.7 & 0.32 \\ 0.32 & 0.75 \end{pmatrix}$$
$$\pi_{\text{un}} = (0.4, 0.6)^T. \tag{3.37}$$

We first assess the performance of all classifiers on the uncontaminated graph under the assumption that the model dimension $d = 2$ is known. The experiment is done via leave–one–out cross validation. For large number of vertices n, all classifiers perform well, with an error rate of almost 0 as seen in Fig 3.8. $\text{LDAoASE}_{d=2}$ performs the best for all choices of vertices. In this ideal setting, SRC_5 does not outperform

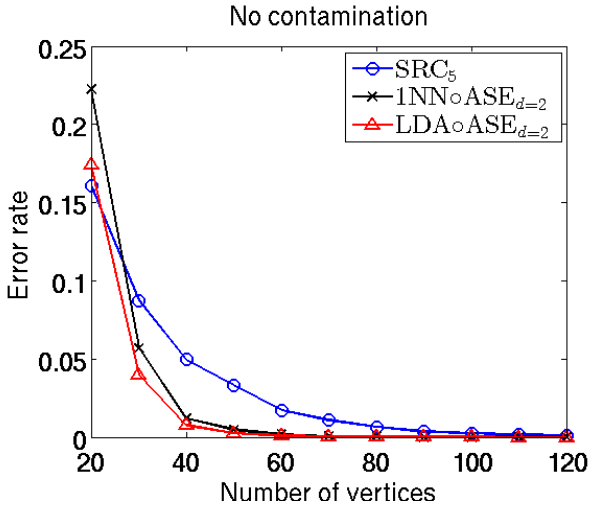


Figure 3.8: Vertex classification performance for uncontaminated model. For the uncontaminated model, when the model dimension $d = 2$ is known, for small number of vertices, SRC_5 does not outperform $\text{1NN}\circ\text{ASE}_{d=2}$ or $\text{LDA}\circ\text{ASE}_{d=2}$. When the number of vertices n is large, all three classifiers exhibit perfect performance. $\text{LDA}\circ\text{ASE}_{d=2}$ exhibits the best performance. The chance line is at 0.4. The number of Monte Carlo replicates is 500.

$\text{NN}\circ\text{ASE}_{d=2}$ or $\text{LDA}\circ\text{ASE}_{d=2}$. Then we fix the number of vertices $n = \{40, 100\}$ and vary the sparsity level s and embedding dimension $\hat{d}$. Figure 3.9 demonstrates the three classifiers' performance. The true model dimension is $d = 2$, so when embedded to $\hat{d} = 1$, $\text{NN}\circ\text{ASE}_1$ or $\text{LDA}\circ\text{ASE}_1$ perform very badly. For $n = 40$, as we increase the embedding dimension $d \in \{1, 2, \ldots, 25\}$, the performance of $\text{NN}\circ\text{ASE}_{\hat{d}}$ and $\text{LDA}\circ\text{ASE}_{\hat{d}}$ improve and then degrade. This phenomenon is due to the bias-variance tradeoff. $\text{1NN}\circ\text{ASE}_{\hat{d}}$ degrades more than $\text{LDA}\circ\text{ASE}_{\hat{d}}$ in performance when

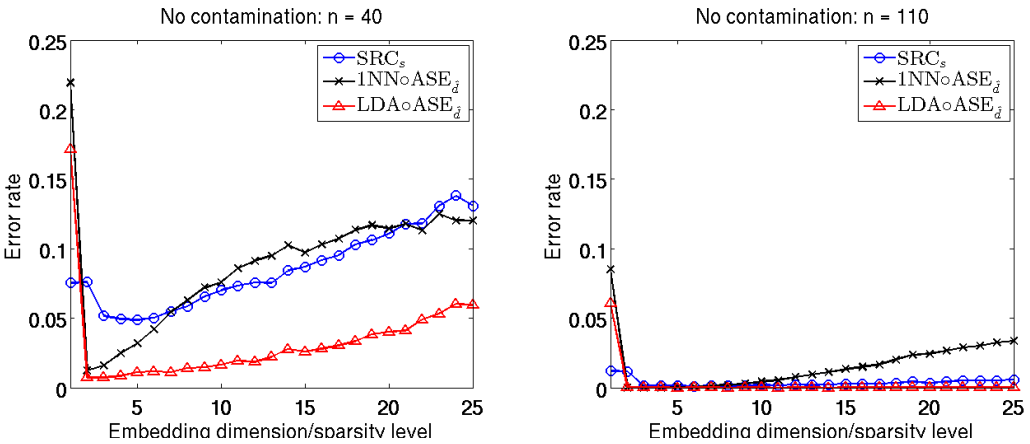


Figure 3.9: Vertex classification on the uncontaminated data with fixed n. (Left) $n = 40$. (Right) $n = 110$. For both cases, SRC$_s$ demonstrates relatively stabler performance with respect to sparsity level s compared to 1NN∘ASE$_{\hat{d}}$ and LDA∘ASE$_{\hat{d}}$ with respect to embedding dimension $\hat{d}$.

we increase $\hat{d}$. This is due to the nonparametric nature of NN but parametric nature of LDA. In this example, SRC$_s$ does not outperform NN∘ASE$_{\hat{d}}$ and LDA∘ASE$_{\hat{d}}$ for all s, but it demonstrates its relatively stabler performance with respect to s compared to 1NN∘ASE$_{\hat{d}}$ and LDA∘ASE$_{\hat{d}}$. Moreover, choosing $s > 10$ may not be suitable for SRC, since the sparsity ratio $\frac{s}{n} > \frac{10}{40} = 0.25$ makes the recovered coefficient not sparse. For $n = 110$, we see that SRC$_s$ and LDA∘ASE$_{\hat{d}}$ perform well, while 1NN∘ASE$_{\hat{d}}$ degrades for larger embedding dimensions. SRC$_s$ demonstrates very stable performance for all $s \in \{1, 2, \ldots, 25\}$.

Next we examine how block dissimilarity affects classification performance on uncontaminated data. For this experiment, we select $s = 5$ for SRC. We consider the

block probability matrix B having a symmetric form,

$$B_{\text{un}} = \begin{pmatrix} \alpha+\beta & \beta \\ \beta & \alpha+\beta \end{pmatrix},$$

where the block dissimilarity Δ is defined as

$$\Delta = |\alpha|. \tag{3.38}$$

We assume $\alpha > 0$. Hence, $\mathbb{G}_{\text{un}}$ is an affinity SBM. High values of Δ in the stochastic blockmodels imply strong block structures. In our experiment, we choose the parameters $\beta = 0.3$, $\alpha \in \{0.15, 0.17, 0.19, \ldots, 0.65\}$, $n \in \{100, 150, \ldots, 350\}$ and $\pi_{\text{un}} = (0.4, 0.6)^T$. We see that SRC performs better at larger Δ values as seen in Fig 3.10. The relative performances between SRC and NNoASE, SRC and LDAoASE are measured by the classification error differences $\text{SRC}_{\text{err}} - \text{NNoASE}_{\text{err}}$ and $\text{SRC}_{\text{err}} - \text{LDAoASE}_{\text{err}}$ respectively. For small values of Δ, the differences are positive for both comparisons as seen in Fig 3.11 and 3.12. SRC does not outperform NNoASE and LDAoASE for weak block signals.

3.7.1.2 Under Contamination

Now we assess the robustness of SRC_s, $\text{1NN}\circ\text{ASE}_{\hat{d}}$ and $\text{LDA}\circ\text{ASE}_{\hat{d}}$ under contamination. If the model dimension $d = 2$ is known and the exact contamination is known, then embedding to $d_{\text{occ}} = 4$ or $d_{\text{rev}} = 4$ is correct. If d is known, but no contamination information is available, or if d is unknown, one could select $\hat{d} = 2$

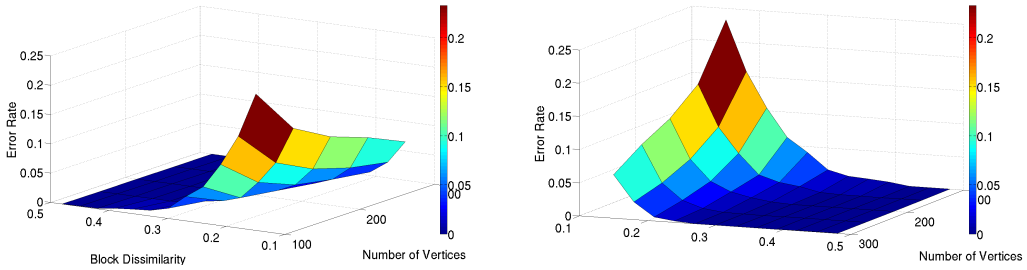


Figure 3.10: Surface plot of SRC error rate with respect to block dissimilarity Δ and number of vertices n. In this case, the graph is not contaminated, and the embedding dimension $d = 2$ is known. Larger Δ values imply stronger block structures. The error rate is high for small values of Δ when the block structures are weak. The number of Monte Carlo replicates is 30.

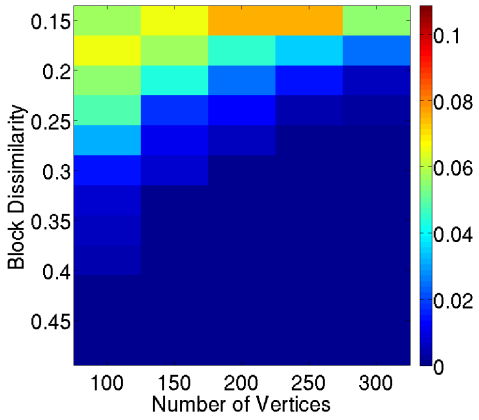


Figure 3.11: Heatmap of the performance difference $\text{SRC}_{\text{err}} - \text{NNoASE}_{\text{err}}$ with respect to block dissimilarity Δ and number of vertices n. For weak block structures at smaller values of Δs, SRC does not outperform NNoASE. As Δ gets larger, indicating stronger block structures, the performance difference becomes smaller and eventually zero. The number of Monte Carlo replicates is 30.

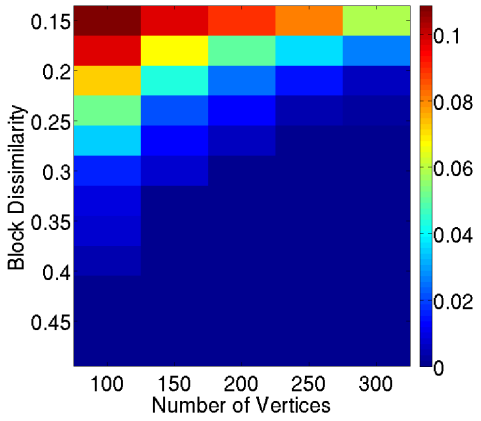


Figure 3.12: Heatmap of the performance difference $\text{SRC}_{\text{err}} - \text{LDAoASE}_{\text{err}}$ with respect to block dissimilarity Δ and the number of vertices n. We see that for weak block structures at smaller values of Δs, SRC does not outperform LDAoASE. As Δ increases, indicating stronger block structures, the performance difference becomes smaller and eventually zero. The number of Monte Carlo replicates is 30.

based on the scree plots of the contaminated adjacency matrices as seen in Figure 3.4 and Figure 3.5.

Figure 3.13 presents the misclassification error of $\text{SRC}_{s=5}$, $\text{1NN}\circ\text{ASE}_{\hat{d}}$ and $\text{LDA}\circ\text{ASE}_{\hat{d}}$ under occlusion contamination for $\hat{d} = 2, 4$. All classifiers degrade as the occlusion rate p_o increases above 0.4. $\text{LDA}\circ\text{ASE}_{d=4}$ performs amongst the best for all values of p_o. Both $\text{LDA}\circ\text{ASE}_{d=4}$ and $\text{1NN}\circ\text{ASE}_{d=4}$ degrade in performance for $p_o > 0.75$. The degradation is due to more severe occlusion contamination in the sampled graph. Both $\text{LDA}\circ\text{ASE}_{\hat{d}=2}$ and $\text{1NN}\circ\text{ASE}_{\hat{d}=2}$ degrade in performance as early as $p_o > 0.45$. The degradation is due to occlusion contamination and embedding dimension misspecification. In this case, $\text{SRC}_{s=5}$ outperforms $\text{LDA}\circ\text{ASE}_{\hat{d}=2}$ and $\text{1NN}\circ\text{ASE}_{\hat{d}=2}$, indicating strong robustness to contamination. For each occlusion rate $p_o \in [0.45, 0.93]$, the Wilcoxon signed rank test is applied with the null hypotheses that the error differences $\text{SRC}_{\text{error}} - \text{1NN}\circ\text{ASE}_{\hat{d}=2\ \text{error}}$ and $\text{SRC}_{\text{error}} - \text{LDA}\circ\text{ASE}_{\hat{d}=2\ \text{error}}$ come from a distribution with zero median respectively. All the p-values are less than 0.005.

Next, we vary the sparsity level s and embedding dimension $\hat{d}$, and compare the classification performance under occlusion contamination with fixed occlusion rate $p_o \in \{0.6, 0.7, 0.9\}$. As seen in Figure 3.14, SRC_s demonstrates its stable performance with respect to various sparsity level s, compared to $\text{1NN}\circ\text{ASE}_{\hat{d}}$ and $\text{LDA}\circ\text{ASE}_{\hat{d}}$ with respect to embedding dimension $\hat{d}$.

Figure 3.15 shows the misclassification error of $\text{SRC}_{s=5}$, $\text{1NN}\circ\text{ASE}_{\hat{d}}$ and $\text{LDA}\circ\text{ASE}_{\hat{d}}$ under linkage reversion for $\hat{d} = 2, 4$. $\text{LDA}\circ\text{ASE}_{d=4}$ performs the best for all val-

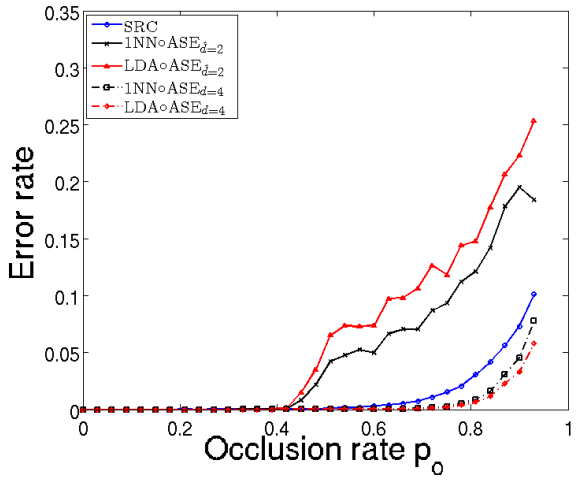


Figure 3.13: Classification performance on the occlusion contaminated data. Performance of the five classifiers when p_o-proportion of the vertices are chosen to have occluded connections for $p_o \in \{0, 0.03, \ldots, 0.93\}$. We simulate 100 SBMs with B_{un}, π_{un} given in Eq. 3.37 and $n = 200$, and average the misclassification error for the five classifiers over the 100 Monte Carlo replicates. All classifiers degrade as the occlusion rate p_o increases. Both LDA∘ASE$_{d=4}$ and 1NN∘ASE$_{d=4}$ degrade in performance for $p_o > 0.75$. The degradation is due to higher sparsity level in the sampled graph. Both LDA∘ASE$_{\hat{d}=2}$ and 1NN∘ASE$_{\hat{d}=2}$ degrade in performance as early as $p_o > 0.45$. The degradation is due to higher sparsity level and embedding dimension misspecification. SRC has much lower misclassification rate compared to 1NN∘ASE$_{\hat{d}=2}$ and LDA∘ASE$_{\hat{d}=2}$ for all values of $p_o > 0.4$, where $\hat{d} = 2$ is chosen based on the scree plot of the occluded adjancecy matrix A_{occ}. LDA∘ASE$_{d=4}$ performs the best for all values of p_o. The standard errors are small compared to the differences in performance. Not requiring information on the model dimension, SRC outperforms LDA∘ASE$_{\hat{d}=2}$ and 1NN∘ASE$_{\hat{d}=2}$, indicating robustness to occlusion contamination.

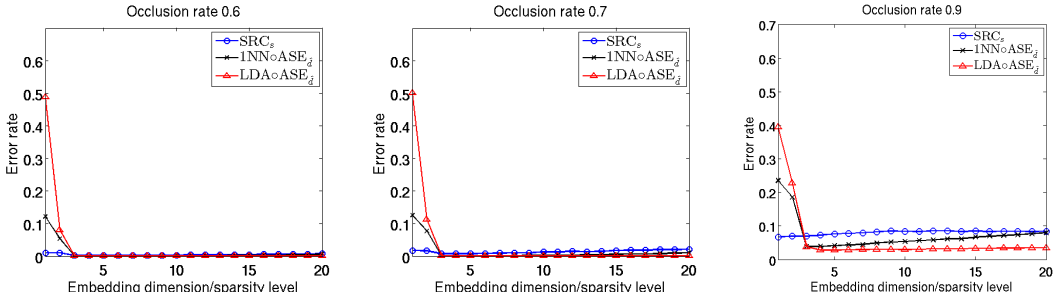


Figure 3.14: Vertex classification on the occluded data with fixed occlusion rate p_o. (Left) $p_o = 0.6$. (Middle) $p_o = 0.7$. (Right) $p_o = 0.9$. For all cases, SRC$_s$ demonstrates relatively stabler performance with respect to sparsity level s compared to 1NN∘ASE$_{\hat{d}}$ and LDA∘ASE$_{\hat{d}}$ with respect to embedding dimension $\hat{d}$.

ues of p_l. Both LDA∘ASE$_{d=4}$ and 1NN∘ASE$_{d=4}$ degrade in performance for $p_l \in [0.75, 0.95]$. The degradation is due to a weaker block signal in the sampled graph. Both LDA∘ASE$_{\hat{d}=2}$ and 1NN∘ASE$_{\hat{d}=2}$ degrade in performance for $p_l \in [0.4, 0.95]$. The degradation is due to a weaker block signal and embedding dimension misspecification. As $p_l \to 1$, all classifiers perform well since the reversed block signal becomes stronger. For $p_l = 1$, the sampled graph is distributed according to the complement of the original SBM. SRC$_{s=5}$ outperforms LDA∘ASE$_{\hat{d}=2}$ and 1NN∘ASE$_{\hat{d}=2}$, indicating strong robustness to contamination. For each linkage reversion rate $p_l \in [0.4, 0.95]$, the Wilcoxon signed rank test is applied with the null hypotheses that the error differences SRC$_{\text{error}}$ − 1NN∘ASE$_{\hat{d}=2\ \text{error}}$ and SRC$_{\text{error}}$ − LDA∘ASE$_{\hat{d}=2\ \text{error}}$ come from a distribution with zero median respectively. All the p-values are less than 0.005.

Next, we vary the sparsity level s and embedding dimension $\hat{d}$, and compare the

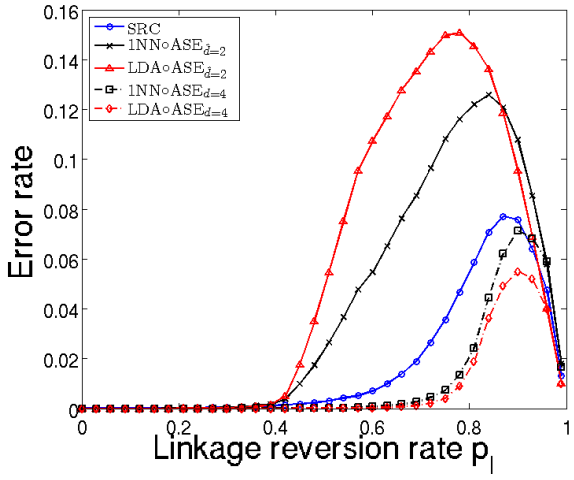


Figure 3.15: Classification performance on the linkage reversion contaminated data. Performance of the five classifiers when p_l-proportion of the vertices are chosen to have reversed connections for $p_l \in \{0, 0.03, \ldots, 1\}$. We simulate 100 SBMs with B_{un}, π_{un} given in Eq. 3.37 and $n = 200$, and average the misclassification error for the five classifiers over the 100 Monte Carlo replicates. Both LDA$\circ$ASE$_{d=4}$ and 1NN$\circ$ASE$_{d=4}$ degrade in performance for $p_l \in [0.75, 0.95]$. The degradation is due to a weaker block signal in the sampled graph. Both LDA$\circ$ASE$_{\hat{d}=2}$ and 1NN$\circ$ASE$_{\hat{d}=2}$ degrade in performance for $p_l \in [0.4, 0.95]$. The degradation is due to a weaker block signal and embedding dimension misspecification. As $p_l \to 1$, all classifiers perform well since the reversed block signal becomes stronger. SRC$_{s=5}$ has much lower misclassification rate compared to 1NN$\circ$ASE$_{\hat{d}=2}$ and LDA$\circ$ASE$_{\hat{d}=2}$ for all values of $p_l \in [0.4, 0.95]$, where $\hat{d} = 2$ is chosen based on the scree plot of the linkage reversed adjancecy matrix A_{rev}. LDA$\circ$ASE$_{d=4}$ performs the best for all values of p_l.

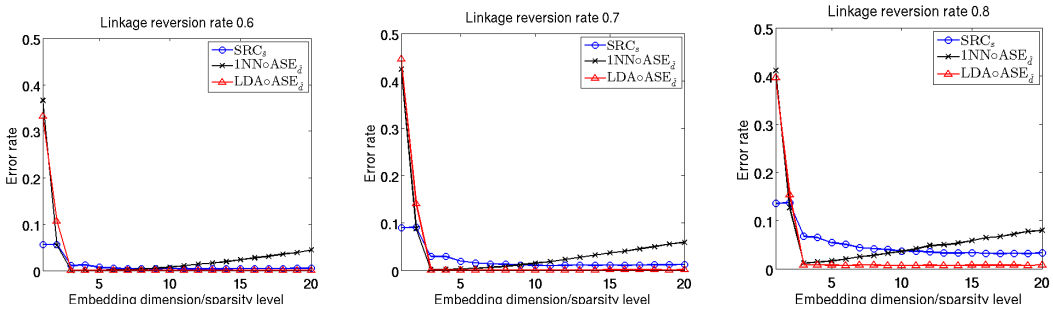


Figure 3.16: Vertex classification on the data contaminated by linkage reversion. (Left) $p_l = 0.6$. (Middle) $p_l = 0.7$. (Right) $p_l = 0.8$. For all cases, SRC$_s$ demonstrates relatively stabler performance with respect to sparsity level s compared to 1NN∘ASE$_{\hat{d}}$ and LDA∘ASE$_{\hat{d}}$ with respect to embedding dimension $\hat{d}$.

classification performance under linkage reversion contamination with fixed occlusion rate $p_l \in \{0.6, 0.7, 0.8\}$. As seen in Figure 3.16, SRC$_s$ demonstrates its stable performance with respect to various sparsity level s, compared to 1NN∘ASE$_{\hat{d}}$ and LDA∘ASE$_{\hat{d}}$ with respect to embedding dimension $\hat{d}$.

3.7.2 Enron E-mail Network

We apply the sparse representation classifier to the e-mail communication network consisting of 184 Enron executives as actors from November 1998 to June 2002 [53, 73]. The communication matrix A is symmetrized, binarized and made hollow, yielding 2097 edges. The matrix sparsity level is $\frac{2097}{\binom{184}{2}} = 12.46\%$. To acquire the class labels, we applied adjacency spectral embedding on A, where the embedding dimension $\hat{d} = 9$ is selected at the first elbow of the scree plot of A [102]. We then applied a clustering

algorithm (Mclust [42]) on $\text{ASE}_{\hat{d}=9}$ to obtain class labels of each actor.The proportion of Class 1 actors is 45.11% and the proportion of Class 2 actors is 54.89%. The chance error for this classification task is 45.11%. Figure 3.17 is the adjacency matrix with actors sorted according to their class memberships. The estimated block matrix $\hat{B} = \left(\begin{smallmatrix} 0.2451 & 0.0746 \\ 0.0746 & 0.1286 \end{smallmatrix}\right)$ is diagonally dominant. The Enron actors communicate relatively more frequently with members in their own block than with the other block.

The true model dimension for the Enron communication network is unknown. We first choose the embedding dimension $\hat{d} = 2$ because of the two class labels and the fairly strong two-block signals. Figure 3.18 shows the misclassification errors of SRC_s, $\text{1NN}\circ\text{ASE}_{\hat{d}}$ and $\text{LDA}\circ\text{ASE}_{\hat{d}}$ as we vary the embedding dimension $\hat{d} \in \{1, 2, \ldots, 80\}$ and sparsity level $s \in \{1, 2, \ldots, 80\}$ respectively. As $\hat{d}$ increases to 28, $\text{LDA}\circ\text{ASE}_{\hat{d}}$ improves in performance, since more signal is included in the embedded space. As $\hat{d}$ continues to increase, $\text{LDA}\circ\text{ASE}_{\hat{d}}$ degrades in performance, since more noise is included. $\text{1NN}\circ\text{ASE}_{\hat{d}}$ improves in performance as $\hat{d}$ increases to 8, since more signal is included in the embedded space. As $\hat{d}$ continues to increase, the performance of $\text{1NN}\circ\text{ASE}_{\hat{d}}$ gradually degrades. SRC_s performance is more robust with respect to various sparsity level compared to $\text{1NN}\circ\text{ASE}_{\hat{d}}$ and $\text{LDA}\circ\text{ASE}_{\hat{d}}$ with respect to $\hat{d}$. This demonstrates SRC robustness to contamination and its practical advantage in random graph inference.

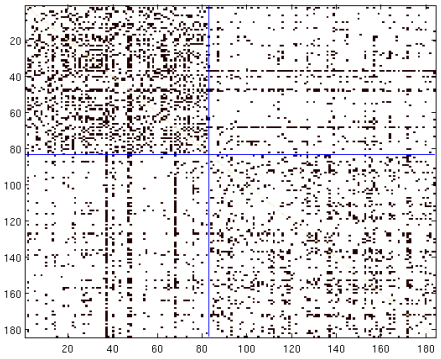


Figure 3.17: **Adjacency matrix of the Enron communication network.** A fairly strong two-block structure is reflected in the adjacency matrix.

3.7.3 Adjective and Noun Network

This dataset, collected in [68], is a network of common adjective and noun adjacency matrices from the novel "David Copperfield" by Charles Dickens. In the network, the vertices are the 60 most frequently used adjectives and 60 most frequently used nouns in the book. The edges are present if any pair of words occur in adjacent position in the book. We apply SRC_s, $\text{1NN}\circ\text{ASE}_{\hat{d}}$, and $\text{LDA}\circ\text{ASE}_{\hat{d}}$ on this dataset, and vary the embedding dimension and sparsity level $d, s \in \{1, 2, \ldots, 50\}$. The chance error is 50%. The performance of the three classifiers is seen in Figure 3.17. SRC_s outperforms $\text{1NN}\circ\text{ASE}_{\hat{d}}$, and $\text{LDA}\circ\text{ASE}_{\hat{d}}$, and exhibits stable performance with respect to various sparsity level s.

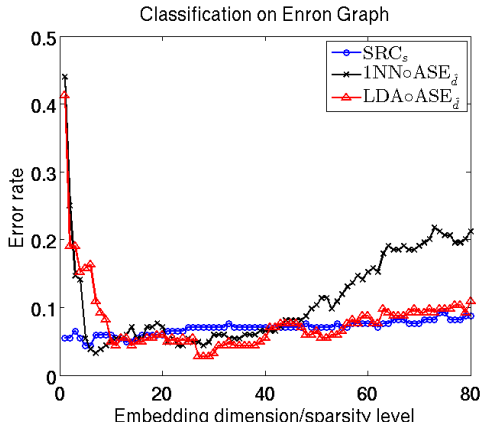


Figure 3.18: Classification performance on the Enron network. We show the performance of SRC_s, as we vary the sparsity level $s \in \{1, 2, \ldots, 80\}$, and the performance of $\text{1NN}\circ\text{ASE}_{\hat{d}}$ and $\text{LDA}\circ\text{ASE}_{\hat{d}}$, as we vary the embedding dimension $\hat{d} \in \{1, 2, \ldots, 80\}$. $\text{LDA}\circ\text{ASE}_{\hat{d}}$ improves in performance as $\hat{d}$ increases to 28, since more signal is included in the embedded space. As $\hat{d}$ continues to increase to 80, $\text{LDA}\circ\text{ASE}_{\hat{d}}$ degrades in performance, since more noise is included. $\text{1NN}\circ\text{ASE}_{\hat{d}}$ improves in performance as $\hat{d}$ increases to 8, since more signal is included in the embedded space. As $\hat{d}$ continues to increase to 80, the performance of $\text{1NN}\circ\text{ASE}_{\hat{d}}$ gradually degrades. SRC demonstrates stable performance with respect to various sparsity levels.

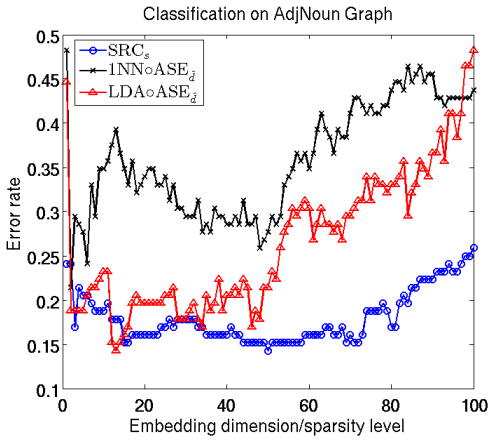


Figure 3.19: Vertex classification performance on the adjective and noun network. SRC$_s$ demonstrates superior and stable performance compared to 1NN∘ASE$_{\hat{d}}$, and LDA∘ASE$_{\hat{d}}$.

3.7.4 Political Blog Sphere

The political blog sphere was collected in February 2005 [3]. The vertices are blogs during the time of the 2004 presidential election, and edges exist if the blogs are linked. There are two classes of the blogs: liberal and conservative. We apply SRC$_s$, 1NN∘ASE$_{\hat{d}}$, and LDA∘ASE$_{\hat{d}}$ on this dataset. Figure 3.20 demonstrate the performance. Again, we see that SRC$_s$ has stable performance with respect to various sparsity level. For all selections of sparsity levels/embedding dimensions, SRC$_s$ outperforms 1NN∘ASE$_{\hat{d}}$, and LDA∘ASE$_{\hat{d}}$.

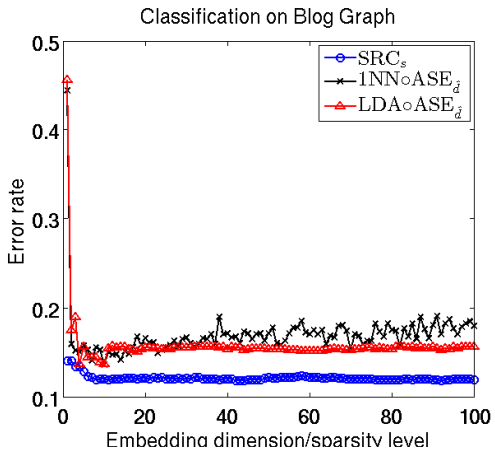


Figure 3.20: Vertex classification performance on the political blog network. SRC$_s$ demonstrates stable performance with respect to various sparsity levels s and outperforms 1NN∘ASE$_{\hat{d}}$ and LDA∘ASE$_{\hat{d}}$.

3.7.5 The *Caenorhabditis Elegans* Neural Connectome

We apply SRC to the *Caenorhabditis elegans* (*C.elegans*) neural connectome ([47], [45]). The *C.elegans* nervous system consists of 302 neurons, which include 20 neurons of the pharyngeal nervous system and 282 neurons of the somatic nervous system. Our analysis is particularly restricted to the 279 somatic neurons ([93]). Those neurons are classified into 3 classes: motor neurons (42.29%), interneurons (29.75%) and sensory neurons (27.96%). See Chapter 7 for the history and research background for this worm. Here we particularly consider classification on the electric connectome.

Figure 7.3 (in Chapter 7) presents the adjacency matrix of the electric connectome

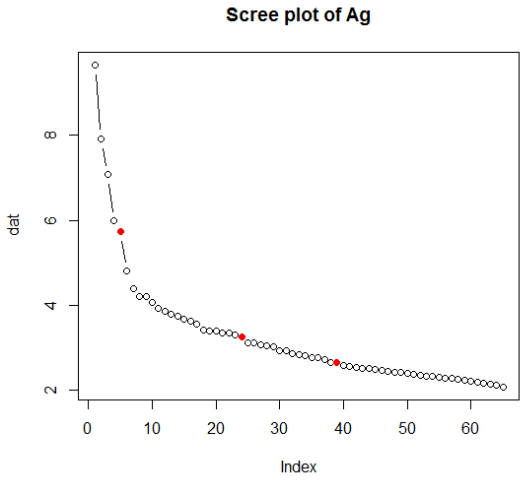


Figure 3.21: Scree plots of the *C.elegans* electric connectome. The red dots mark the first three elbows.

A_g, where a three-block structure is seen. Figure 3.21 displays the connectome's eigenstructures, which is demonstrated to be approximately low rank. The red dots are the identified "elbows" using an automatic dimension selection method [102].

We vary the embedding dimension $\hat{d} \in \{1, 2, \ldots, 100\}$ and examine the performance of SRC_s, $\text{1NN}\circ\text{ASE}_{\hat{d}}$ and $\text{LDA}\circ\text{ASE}_{\hat{d}}$, as seen in Figure 3.22. As $\hat{d}$ increases to 60, $\text{LDA}\circ\text{ASE}_{\hat{d}}$ improves in performance since more signal is included in the embedded space. As $\hat{d}$ continues to increase, $\text{LDA}\circ\text{ASE}_{\hat{d}}$ degrades in performance since more noise is included. $\text{9NN}\circ\text{ASE}_{\hat{d}}$ improves in performance as $\hat{d}$ increases, since more signal is included in the embedded space. As $\hat{d}$ continues to increase, the performance of $\text{1NN}\circ\text{ASE}_{\hat{d}}$ gradually degrades but at much slower rate than $\text{LDA}\circ\text{ASE}_{\hat{d}}$. The exhibited phenomenon is due to the nonparametric nature of 9NN but parametric nature

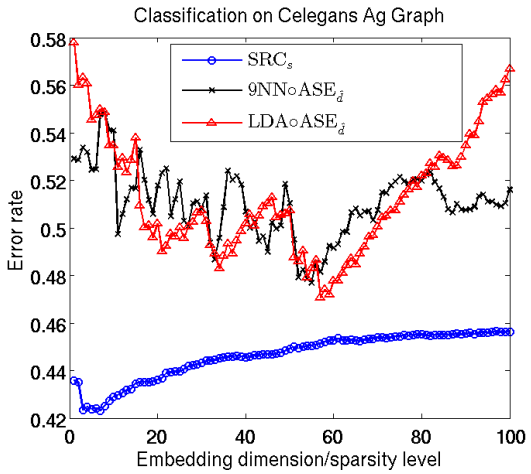


Figure 3.22: Classification performance on *C.elegans* electric connectome. SRC$_s$ maintains much lower error rate and is stable against various sparsity level s.

of LDA. For all sparsity level s, SRC$_s$ exhibits superior and stable performance. This demonstrates that SRC$_s$ is robust to data contamination and its practical advantage in random graph inference.

3.7.6 Political Book Graph

In this graph, the vertices are the 105 books about US politics and sold by Amazon.com [68]. The edges exist if any pairs of books were purchased by the same customer. There are 3 class labels on the books: liberal (46.67%), neural (40.95%) and conservative (12.28%). The performance of the three classifiers is seen in Figure 3.23. SRC$_s$ outperforms 1NN∘ASE$_{\hat{d}}$, and LDA∘ASE$_{\hat{d}}$.

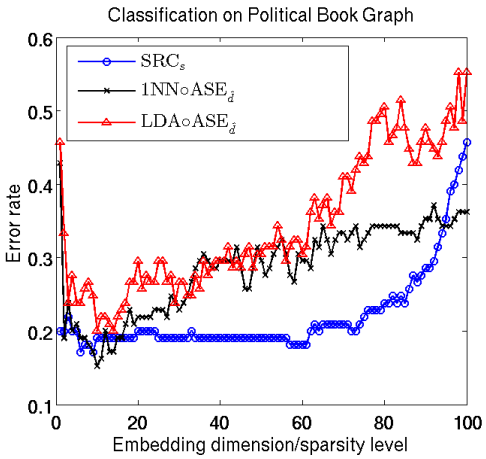


Figure 3.23: Classification performance on the political book graph. SRC$_s$ demonstrates superior to 1NN∘ASE$_{\hat{d}}$ and LDA∘ASE$_{\hat{d}}$.

3.7.7 Wikipedia Graphs

The Wikipedia dataset contains 1382 English articles on "algebraic geometry" and the corresponding 1382 French articles. For each language, a graph is made using the document hyperlinks. All articles are partitioned into five disjoint topics. We apply SRC$_s$, 9NN∘ASE$_{\hat{d}}$, and LDA∘ASE$_{\hat{d}}$ on both the English and the French graphs. The superior performance of SRC$_s$ is seen in Figure 3.24.

3.8 Discussion

Adjacency spectral embedding is a feature extraction approach for latent position graphs. When feature extraction is composed with simple classifiers such as NN or LDA, the choice of feature space or embedding dimension is crucial. Given the model

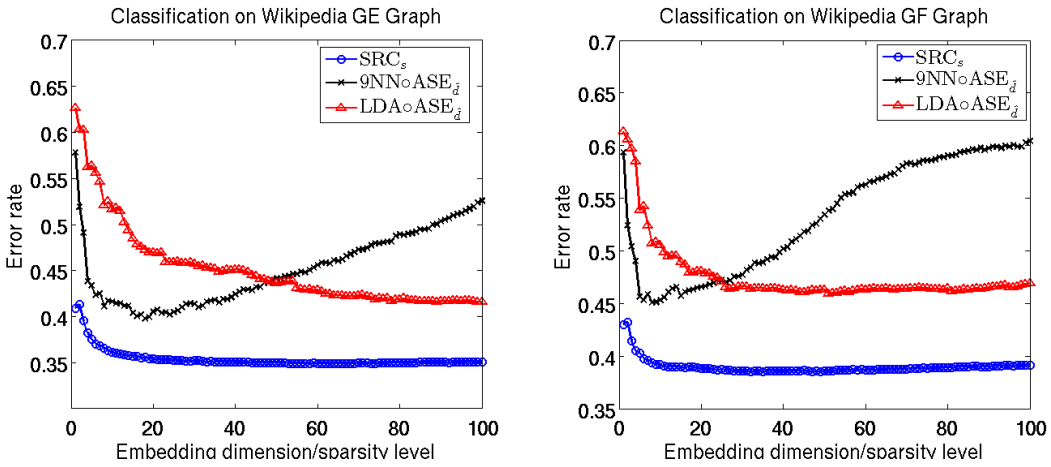


Figure 3.24: Classification performance on the Wikipedia (Left) English graph, (Right) French graph. Again, SRC$_s$ demonstrates superior than 9NN∘ASE$_{\hat{d}}$, and LDA∘ASE$_{\hat{d}}$. Moreover, its performance is more stable with respect to different sparsity levels s, compared to the performance of 9NN∘ASE$_{\hat{d}}$ and LDA∘ASE$_{\hat{d}}$ with respect to the embedding dimension $\hat{d}$.

dimension d for a stochastic blockmodel, ASE_d is consistent and the exploitation task of vertex classification via $\text{1NN}\circ\text{ASE}_d$ and $\text{LDA}\circ\text{ASE}_d$ is Bayes optimal or Bayes plug-in. The success of ASE procedures depends heavily on the knowledge of d. Otherwise, for finite samples, the performance of ASE degrades significantly when the estimated embedding dimension is not the model dimension of the stochastic blockmodel. Therefore ASE is not robust to model dimension misspecification.

Nonetheless, in practical inference tasks, the model dimension d is unknown. In this chapter, we present a robust vertex classifier via sparse signal representation for latent position graphs. The sparse representation classifier does not need information on the embedding dimension of the stochastic blockmodels, but it still maintains good classification performance. As seen in the simulation studies, when d is known, SRC does not outperform $\text{1NN}\circ\text{ASE}_d$ and $\text{LDA}\circ\text{ASE}_d$. When d is estimated using the scree plot of the adjacency matrix, it does outperform $\text{1NN}\circ\text{ASE}_{\hat{d}}$ and $\text{LDA}\circ\text{ASE}_{\hat{d}}$. In real data experiments, the true model dimension d is unknown. SRC_s outperforms $\text{1NN}\circ\text{ASE}_{\hat{d}}$ and $\text{LDA}\circ\text{ASE}_{\hat{d}}$, as the embedding dimension $\hat{d}$ and the sparsity level s vary. We see that SRC_s demonstrates stable performance for various sparsity levels. We followed [64] and [80], which suggest imputing the diagonal of the adjacency matrix to improve performance. The increase in prediction accuracy is shown in the ASE approaches. However, the improvement is not enough to surpass the good performance by SRC in the real data experiments. All the numerical studies strongly indicate the robustness of SRC to contamination.

Our findings indicate a variety of directions for future work. In the following subsections, we briefly discuss some major and interesting aspects worthy of further investigation.

3.8.1 L_0 or L_2 Sparsity Approach

Sparse representation allows us to use all the information observed from the graph directly, and proceed inference in the graph space with L_0 sparsity. This sparsity is encouraged through the whole procedure. Adjacency spectral embedding represents graphs in a transformed Euclidean space with L_2 sparsity, and we usually truncate the dimension, which may eliminate noise and/or useful information. This sparsity is not present in the original graph, but only introduced after the transformation. The choice of inference methodology depends not only on whether enough information is given regarding the model dimension, but also on whether one wants to work in the original graph space or the embedded Euclidean space.

3.8.2 L_2 Normalization

In Algorithm 2, we normalize the columns of the dictionary D so that each column has L_2 unit norm. Empirically, we see improvement in SRC performance under L_2 normalization for a finite number of vertices, as seen in Figure 3.25. Research on whether L_2 normalization on D in SRC has theoretical impact in the asymptotics

deserves investigation.

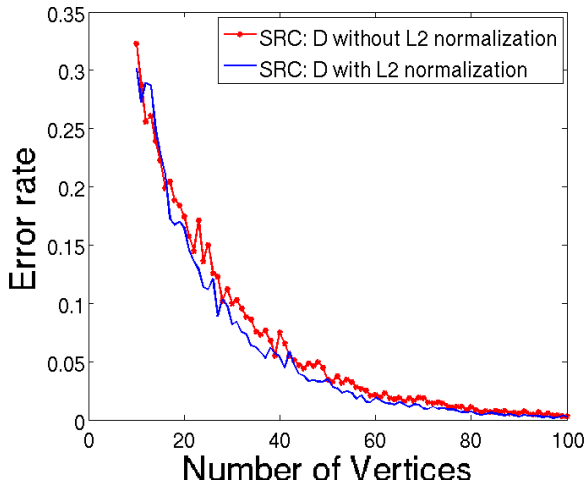


Figure 3.25: Examination of SRC performance with or without L_2 normalization on columns of D. We compare SRC performance when columns of D are L_2 normalized and when columns of D are not L_2 normalized. The parameters B and π are given in Equation 3.37 with $n \in \{10, 11, \ldots, 100\}$ and we run 100 Monte Carlo replicates for each n. We see an improvement in SRC performance when L_2 normalization is applied. The Wilcoxon signed rank test reports a p-value less than 0.05 under the the null hypothesis that the error difference $\text{SRC}_{\text{error},L_2} - \text{SRC}_{\text{error,no } L_2}$ comes from a distribution with zero median.

3.8.3 Consistency of SRC

In our recent work [81], we have shown that SRC is consistent under a principal angle condition. The principal angle condition extends beyond the L_1 minimization

and subspace assumption. We expect to extend the principal angle condition for stochastic blockmodels, and prove theoretical performance guarantees.

3.8.4 Other Implementations of SRC

In the previous experiments, our SRC procedure is implemented using the orthogonal matching pursuit (OMP). We have recently proposed two alternative implementations in [81]: L_1 homotopy and marginal regression. We apply these implementations to non-graph datasets, and see their superior performance over spectral-embedding based classifier. We have also seen their superior performance on graphs. See 3.26 for examples. The best implementation in terms of accuracy and runtime is worth investigating.

3.8.5 Extensions to Similarity Matrices

Our recent work [81] further extends SRC to similarity matrices, compares the classification performance of SRC to classical embedding techniques. We present one numerical example on the Wikipedia text similarity experiment. The Wikipedia dataset contains 1382 English documents obtained from Wikipedia, and their French language corresponding documents. Each document belongs to exactly one of the five classes. We apply Latent Semantic Indexing (LSI) for text processing [34], and then subsequently calculate the cosine similarity to obtain two 1382×1382 similarity

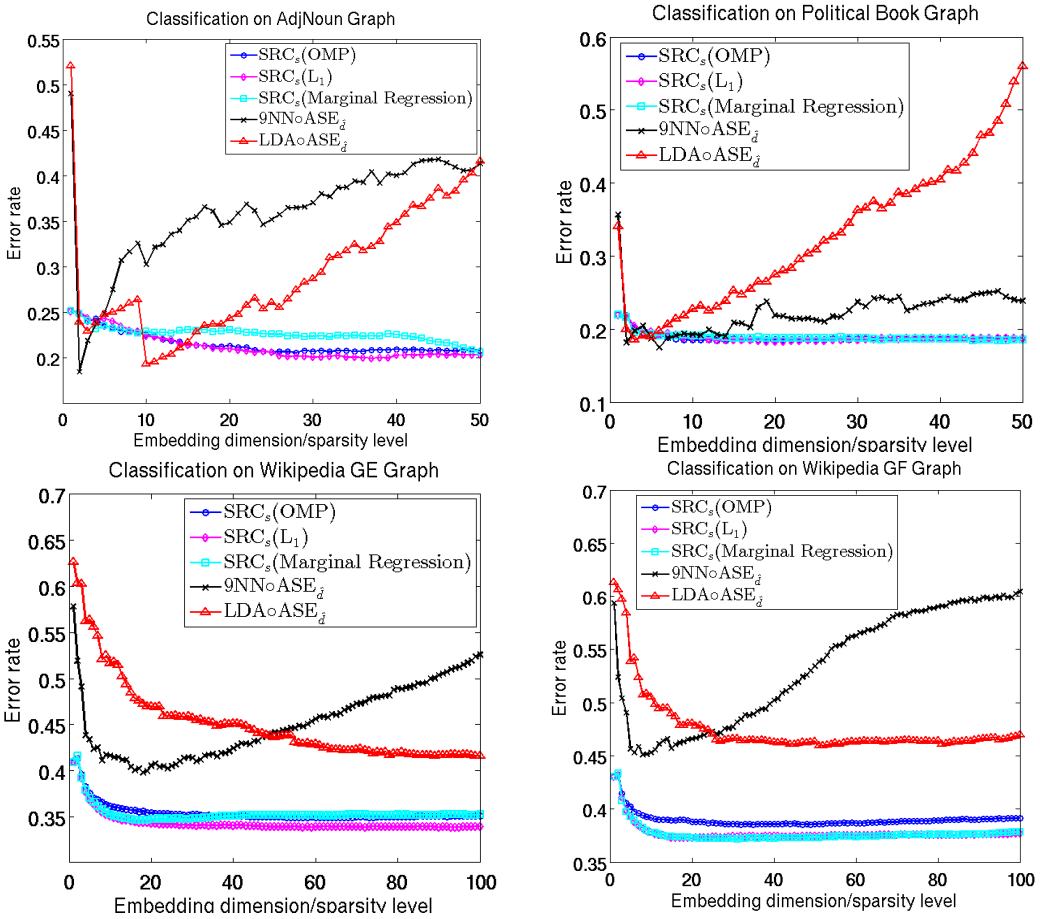


Figure 3.26: We compare three implementations of SRC: OMP, L_1 homotopy, and marginal regression, with 9NN$\circ$ASE$_{\hat{d}}$ and LDA$\circ$ASE$_{\hat{d}}$. We see that all three implementations of SRC demonstrates stable and superior performance than 9NN$\circ$ASE$_{\hat{d}}$ and LDA$\circ$ASE$_{\hat{d}}$. This simulation experiment is performed in collaboration with Cencheng Shen.

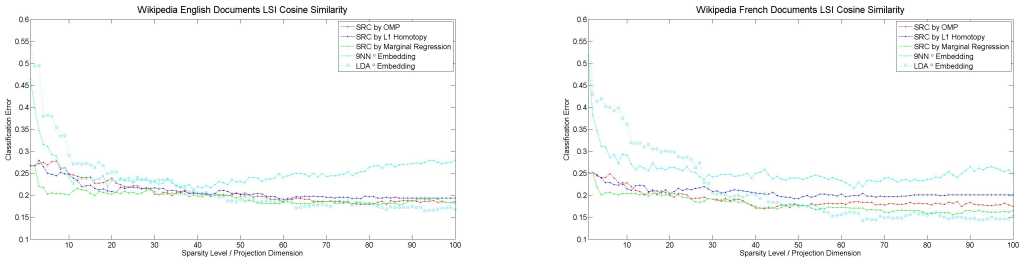


Figure 3.27: The classification error against sparsity level/neighborhood choice/embedding dimension. (Left) English text similarity. (Right) French text similarity. For $d \in \{1, 100\}$, all three implementations of SRC_s outperform 9NN∘Embedding and LDA∘Embedding. As d increases, LDA∘Embedding outperforms SRC_s. This may be caused by the choice of the cosine similarity measure, which is suitable for text data. This allows LDA to perform better within a proper projection dimension range. This simulation experiment is performed in collaboration with Cencheng Shen.

matrices for both English and French documents respectively. The superior performance of SRC_s is shown in Figure 3.27. We see that all three implementations of SRC perform well on similarity data. It is worth answering the question of what type of transformation can improve SRC performance.

Chapter 4

Vertex Clustering

In this chapter, we propose a vertex clustering approach, which employs adjacency spectral embedding followed by a model-based clustering technique. We assume the graphs are realized from a stochastic blockmodel SBM($[n], \pi, B$). For unsupervised learning on SBMs, see [15], [57], [20] and [8]. This chapter mainly focuses on demonstrating the usefulness of our proposed approach in the field of online advertising [16]. We describe our proposed model-based vertex clustering approach in Section 4.2, illustrate its effectiveness via simulation in Section 4.4, and apply it to a case study in online advertising Section 4.5. We will explain the basic concept of online advertising and the business motivation of using our approach in Section 4.5.

4.1 The Problem of Vertex Clustering

In the classical setting for unsupervised learning, we observe independently identically distributed feature vectors $X_1, X_2, \ldots, X_n$, where each $X_i : \Omega \to \mathbb{R}^d$ is a random vector for some probability space Ω. Here we consider the case when the feature vectors $X_1, X_2, \ldots, X_n$ are unobserved. Instead we observe a latent position random graph $\mathcal{G}(X_1, X_2, \ldots, X_n)$ on n vertices. We intend to cluster the vertices using the observed graph.

4.2 The Algorithm

For stochastic blockmodels with K blocks and a known model dimension d, Sussman et al. [84] and Rohe et al. [78] respectively have shown that adjacency spectral embedding and Laplacian spectral embedding are consistent estimates of the latent positions. The resulted embedding is a K-mixture and D-variate Gaussian distributions asymptotically. Such results motivate us to propose a model-based clustering approach on the embedded space of the stochastic blockmodel. The optimal number of clusters and covariance structure correspond to the model selection criterion by the Bayesian information criterion (BIC). Our approach is presented in Algorithm 4. We denote the algorithms by ASE and LAP respectively, if M is either the adjacency matrix or the Laplacian matrix.

We compare ASE and LAP with the integrated classification likelihood (ICL)

Algorithm 4 The Bayesian information criterion based vertex clustering approach

Input: Matrix $M \in \mathbb{R}^{n \times n}$, an integer $K \geq 1$, and an embedding dimension d.

Step 1 : Compute the first d orthonormal eigenpairs of M, denoted by $(U_M, S_M) \in \mathbb{R}^{n \times d} \times \mathbb{R}^d$, where S_M contains the d largest eigen-values in absolute value.

Step 2: Define the d-dimensional embedding of M to be $\hat{M} := U_M S_M^{1/2}$.

Step 3:

for k in $1 : K$ **do**

Fit Gaussian mixture models with different covariance types and k clusters to $\hat{M}$, and compute the BIC.

end for

Step 4: Cluster the vertices using the optimal model selected via the maximum Bayesian information criterion (BIC).

method [27], which is a likelihood maximization method for stochastic blockmodels, and the Louvain algorithm [9], which optimizes graph modularity and performs efficiently for large graphs.

4.3 Clustering Validation

Various clustering algorithms differ in their measurements of quality of partitions, and hence result in a different notion of what constitutes a cluster. For the simulation analysis, we measure the similarity between a clustering partition and the true block

memberships using the adjusted rand index (ARI) [52] defined as

$$ARI = \frac{RI - \mathbb{E}(RI)}{\max(RI) - \mathbb{E}(RI)} \in [0, 1], \tag{4.1}$$

where RI is the rand index [76] and $\mathbb{E}(RI)$ is the expected rand index. The higher ARI indicates a better clustering performance. For our case study, one challenge for clustering validation is the lack of ground truth. We examine the significance of the clusters using external datasets pertaining to business metrics in online advertising.

4.4 Simulation

In this section, we apply our proposed approach using both adjacency and Laplacian matrices, and compare the clustering performance with the integrated classification likelihood method, and the Louvain method. We simulate 100 independent stochastic blockmodels with $K = 2$ blocks and parameters $B = \begin{pmatrix} 0.12 & 0.05 \\ 0.05 & 0.08 \end{pmatrix}$ and $\pi = [0.4, 0.6]^T$. We assess the performance of the four clustering algorithms on the 100 graphs under the assumption that the model dimension $d = 2$ is known.

The simulation result is shown in Figure 4.1. ASE and LAP methods respectively are able to select the correct number of blocks with an average ARI above 90%. As the number of blocks increases from 2 to 6, the ARI of ASE and LAP decrease. This is due to the phenomenon of bias-variance tradeoff. ICL tends to select higher number of blocks. Louvain clustering algorithm optimizes the graph modularity. Thus, its

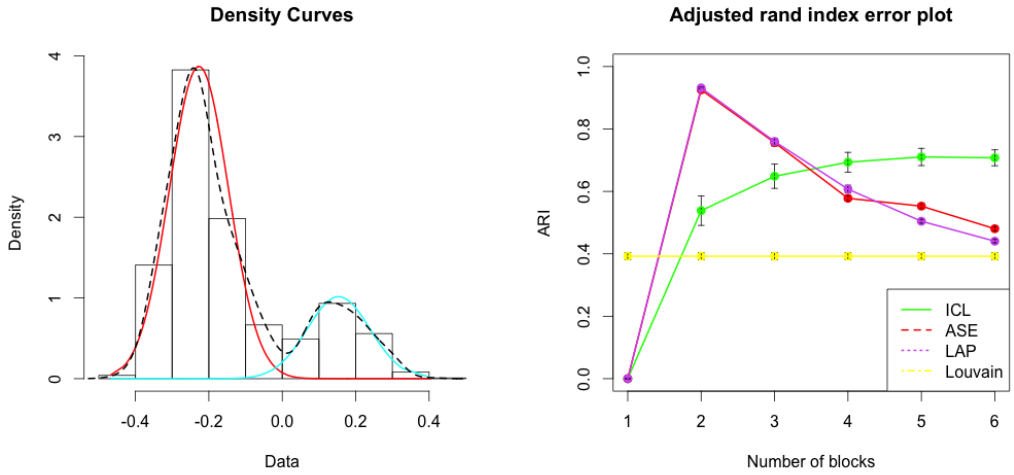


Figure 4.1: Vertex clustering performance on simulation. (Left) Density approximation of adjacency spectral embedding. We fit a bivariate Gaussian to the embedded data after adjacency spectral embedding. (Right) The error plot of ARI against the number of blocks. We present the performance of four vertex clustering methods. ASE and LAP methods respectively are able to select the correct number of blocks with an average ARI above 90%.

performance is constant with respect to the number of blocks. All three clustering algorithms outperform the Louvain clustering algorithm.

4.5 A Vertex Clustering Case Study: Online Advertising

Online advertising is market communication over the internet. This form of advertising has proven its importance in the golden digital age. It has become an important and huge industry. Having knowledge of the website attributes can contribute greatly to business strategies for ad-targeting, content display, inventory purchase or revenue prediction.

4.5.1 The Online Advertising Business

There are three major components in the online advertising business: the advertisers, the ad network and the publishers. The business flow of an ad network is shown in Figure 4.2. Advertisers intend to brand or advertise the products or promotional activities using campaigns. Each campaign constitutes various advertisements. An ad network acts like a middle man, and helps advertisers to post ads on various websites. The ad network receives payments from advertisers as revenue, and pays the publishers to purchase inventories. Inventories, such as websites or banners, publish ads. Different websites, based on their popularity, charge different amounts of money to hold ads. When many online users surf the websites and view the ads, the campaign is considered effective. A campaign is even more effective, when many online

users click or subsequently take an action such as registering or making a purchase[1]

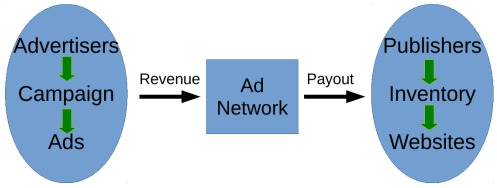


Figure 4.2: The business flow of an ad network. Advertisers pay the ad network, who finds publishers to publish ads on various websites. Each campaign constitutes many ads, which are placed in various online web sites. When many online users view the ads, the campaign is considered effective. When many online users click on the ads or take an action such as making a purchase after viewing the ads, the campaign is even more effective.

The field of online advertising is incredibly complex. Auction theory, bidding system, supervised learning, real time feed back systems are all useful in the analytics of this business. Here we tackle the problem of effectively targeting a wider variety of online users via discovering the intrinsic structures of websites. The business motivation comes in two aspects: save cost to purchase inventories, and reach a larger online audience. To tackle this problem, we intend to find "similar" websites to post the ads, so that we can selectively choose websites which cost less but has similar online

[1]Here we do not differentiate the metrics for evaluating the effectiveness of ads. In online advertising, different types of ads have different business metrics. For example, for a cost-per-impression ad, the number of views measures the ad effectiveness. For a cost-per-click ad, the number of clicks measures the ad effectiveness. For a cost-per-action ad, the number of actions measures the ad effectiveness.

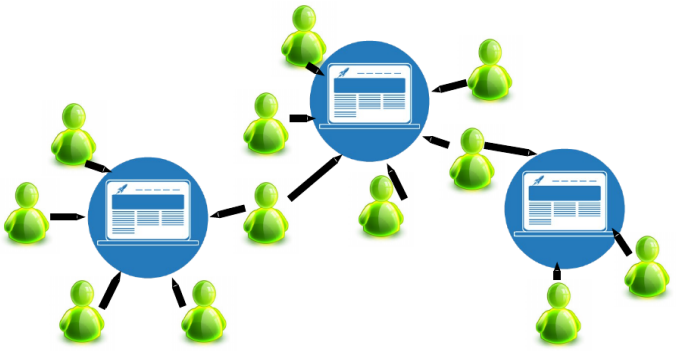


Figure 4.3: The data generating mechanism.

users. We also intend to find dissimilar groups of ads, so that we can post the ads on these dissimilar groups to reach more diverse audience. This can serve as a basic guidance for website inventory acquisition in an ad network. We believe our approach is the first step towards understanding and attempting such a business problem.

4.5.2 Data Description

The dataset for our analysis is obtained under a family-event campaign during the day of July 1, 2014. We use the relational events of online users who visit all the websites, which show an ad under this campaign. Our graph is built with the websites as vertices, and an edge exists between websites i and j, if they share at least one common online user. After removing the isolated vertices, the resulted adjacency matrix is symmetrized, binarized, hollow and of size 1569×1569. In addition to the website network, we obtain the data the website topics, revenue generated per

website, impressions (the volume of ad display) per website, and number of clicks on ads per website.

4.5.3 Results

We compare the results of the four vertex clustering algorithms on the website network. In practice, the true model dimension d of the stochastic blockmodel is unknown. Again, we select the"elbow" using a profile likelihood maximization method proposed in [102]. The low rank eigen-structures of the network is shown in Figure 4.4. The first elbow is 25. In this work, we select the first elbow 25 as the embedding dimension. We denote the algorithms as ASE_1 and LAP_1 respectively using the adjacency matrix and the Laplacian matrix in Algorithm 4, with embedding dimension selected at the first elbow.

As we pointed out before, in practice, the ground truth is often unavailable. Hence the number of blocks is detected using the optimization criterion of each clustering method. Figure 4.5 presents the model selection result by the four clustering algorithms. We examine the intrinsic block structures detected by the four clustering algorithms. The adjacency matrices with vertices sorted according to the partitions of the four clustering method as seen in Figure 4.6. The block structure detected by ASE_1 reflects the clearest block signal compared to the block structures detected by the other three algorithms. Next, we compare the ARIs of the 6 pairs of methods. The ARIs, as shown in Table 4.1, indicate that the partitions by ASE and LAP are

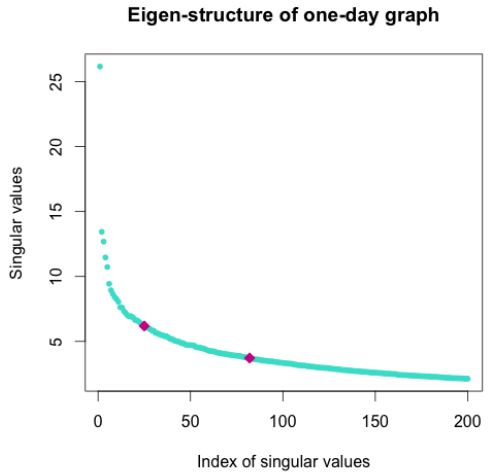


Figure 4.4: The eigen-structure of the one-day website network. The low eigen-structure is shown in the red dots. The first elbow is 25 as selected using a profile likelihood based dimension selection method [102]. In our experiment, we will use $\hat{d} = 25$ as the embedding dimension.

most similar. The partitions by ICL and Louvain are least similar.

Besides using intrinsic metrics, we evaluate the clusters using external datasets. In terms of website topics, Cluster 1 discovered by ASE mainly contains references and popular sites. Cluster 5 discovered by ASE mainly contains websites on politics, baby, teen, gallery. See Table 4.2. The clusters discovered by the other clustering algorithms do not correspond to any significant topic clusters.

In addition, we evaluate the clusters using business metrics: revenue, clicks and impressions. For each clustering algorithm, we apply pairwise two-sided Wilcoxon rank sum test with the null hypothesis that the revenue/clicks/impressions are the

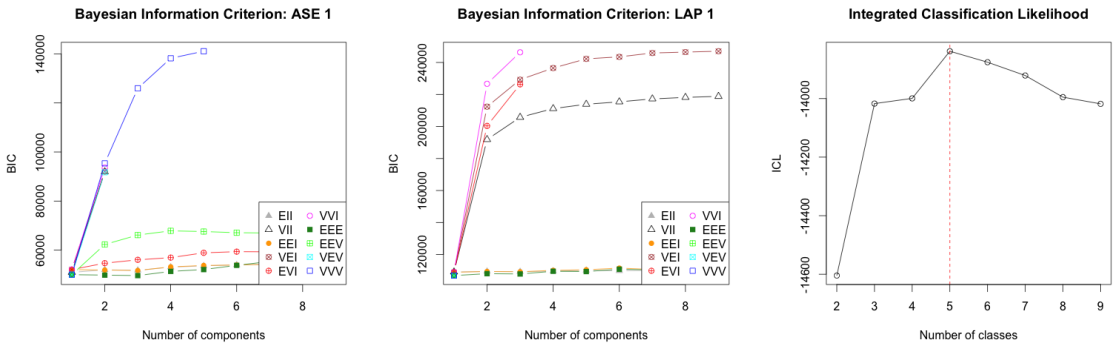


Figure 4.5: Selecting the number of blocks using different clustering criteria. (Left): The BIC plot of ASE_1. The optimal number of blocks is 5 corresponding to the largest BIC. (Middle): The BIC plot of LAP_1. The optimal number of blocks is 9 corresponding to the largest BIC. (Right) The ICL plot of the goodness-of-fit method. The optimal number of blocks is 5. Louvain, not shown here, selects 72 clusters when optimizing the graph modularity.

same for the two clusters. At a significance level of 5%, ASE discovers 3 statistically significant clusters based on the external data metrics.

4.6 Discussion

In this chapter, we propose a Bayesian information criterion based vertex clustering approach for stochastic blockmodels, and apply this approach to a case study in online advertising. We demonstrate in simulation that our proposed algorithms are able to detect the correct number of blocks. In the online advertising web graph

(ASE, LAP)	0.232	(ASE, ICL)	0.097	(ASE, Louvain)	0.038
(LAP, ICL)	0.101	(LAP, Louvain)	0.058	(ICL, Louvain)	-0.015

Table 4.1: The ARIs between the 6 pairs of clustering algorithms. The partition detected by ASE is most similar to LAP. The partition detected by ICL and Louvain is least similar.

Cluster 1	references, mainstream sites.
Cluster 2	engines, video games, computer review.
Cluster 3	engines, computer, TV, gossip, national news, design.
Cluster 4	engines, job search, local news, stocks, weather.
Cluster 5	baby, politics, shopping, gallery, teen.

Table 4.2: A presentation of the dominant website topics in each cluster discovered by the ASE_1.

experiment, our proposed algorithm ASE is able to detect significant website clusters validated using website topics, impressions, revenue and number of clicks. The applications of our approach not only extend to further cluster-based inference, but also can serve as a simple and basic guidance for website inventory acquisition. While our proposed approach is presented for undirected and unweighted graphs, it adapts to directed and weighted graphs. Our approach requires the knowledge of the model dimension of the stochastic blockmodel. Recall that as mentioned in Chapter 3, in

practice the true embedding dimension is unknown. The work presented in Chapter 3 [17] on avoiding selecting the embedding dimension and robustly perform vertex classification may motivate a robust vertex clustering approach. We are optimistic that random graph framework is valuable for online advertising research.

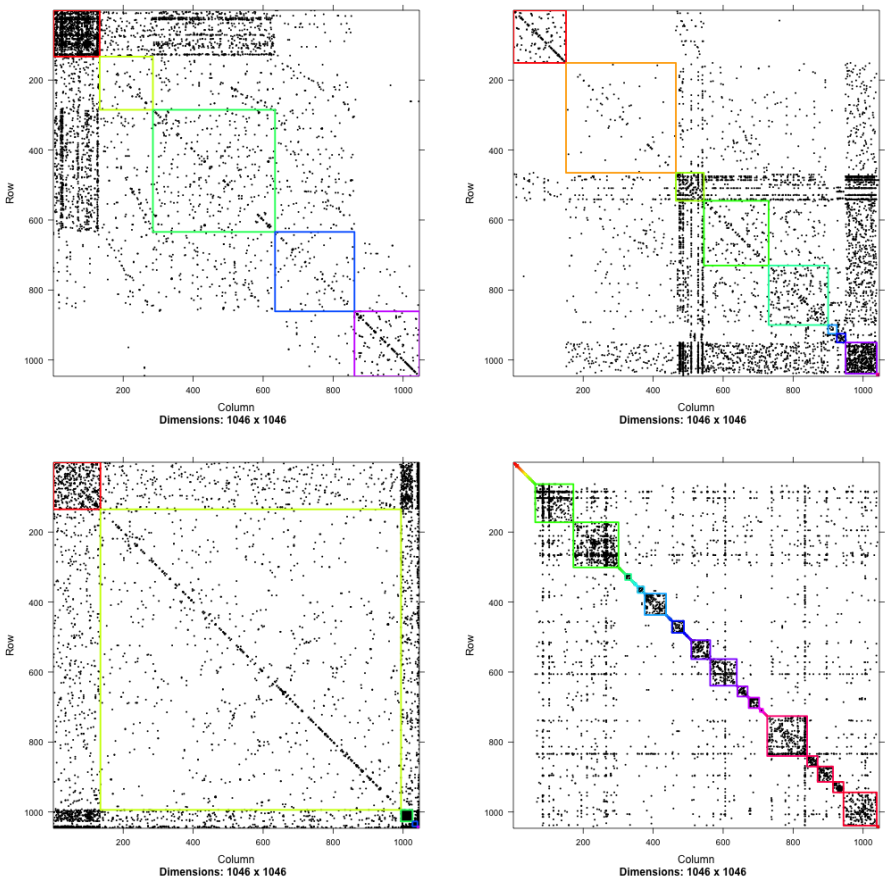


Figure 4.6: Adjacency matrices sorted according four clustering algorithms. The positions of the vertices are corresponding to ASE, LAP, ICL and Louvain. We can see that ASE reflects the clearest block structure.

Chapter 5

Vertex Nomination

Assume that a graph is realized from a stochastic blockmodel such that one of the blocks is of interest. Further suppose that a subset of vertices are already observed as interesting or uninteresting. Among the remaining unobserved vertices, we want a prioritization for the vertices to be in the interesting block. The task is to create a nomination list with an abundance of interesting vertices near the top of the list. For example, in a financial network, suppose we know a subset of traders who have committed fraud. Our task is to nominate the remaining people in a list such that the people at the top of the list are fraudsters.

In this chapter, we present several vertex nomination schemes: the canonical vertex nomination scheme, the canonical sampling vertex nomination scheme, the likelihood maximization vertex nomination scheme, and the spectral partitioning vertex nomination scheme. In particular, our main contribution is the canonical sampling

vertex nomination scheme, which not only approximates the best possible canonical nomination scheme, but also scales to big graph data. We prove a theoretic performance guarantee for the canonical sampling vertex nomination scheme, and demonstrate its scalability to large graphs, nomination performance and runtime in both simulated and real data.

The problem of vertex nomination was dealt with in [23], where the authors considered a likelihood model based on the context and content statistics. Marchette et al. [64] extended vertex nomination to the random dot product graph [98], a generalization of the stochastic blockmodel. Lee and Priebe [55] formulated a Bayesian model framework incorporating the context and content statistics. We proposed several vertex nomination schemes and proved their theoretical performance guarantees [40]. The canonical vertex nomination scheme is the "gold standard". In that, using a mean average precision metric, the canonical vertex nomination scheme outperforms all other vertex nomination schemes. Note that our definition of mean average precision is slightly different from the definition commonly used in information retrieval. We will discuss this difference in Section 5.1.2. Our most recent work presents a canonical sampling vertex nomination scheme, which approximates the "gold standard" of the canonical vertex nomination scheme, and scales to large graphs [19].

5.1 The Vertex Nomination Problem

5.1.1 The Stochastic Blockmodel in the Vertex Nomination Setting

Our vertex nomination is set up within the framework of stochastic blockmodels $\text{SBM}(\vec{n}, \mathcal{L}, B)$ as defined in Definition 2.6. Here we slightly modify this definition due to our task-specific purpose. Assume the vertex set $V = [n] := \{1, 2, \ldots, m, m+1, \ldots, n\}$ is partitioned into $V = S \cup U$, where $S := [m] = \{1, 2, \ldots, m\}$, and $U := \{m+1, \ldots, n\}$. The set S consists of m non-ambiguous vertices, whose block memberships have been observed. The set U consists of $n - m$ ambiguous vertices, whose block memberships have not been observed, and the vertex nomination task is to order the ambiguous vertices into a nomination list such that an abundance of vertices of interest are at the beginning of the list. Define $\vec{n} := (n_1, \ldots, n_K) \in \mathbb{N}^K$ such that n_k denotes the number of vertices in V that belongs to block k. Note that $\sum_{k \in [K]} n_k = n$. Define $\vec{m} := (m_1, \ldots, m_K) \in \mathbb{N}^K$ such that m_k denotes the number of vertices in S that belongs to block k. Note that $\sum_{k \in [K]} m_k = m$. Hence, $\vec{m} \leq \vec{n}$ componentwise. The entries of the vector $\vec{n} - \vec{m}$ are the cardinalities of the blocks among the $n - m$ ambiguous vertices in U.

Let $\mathcal{L}$ denote the set of all possible block assignment functions $l : V \to [K]$ such that $|\{v \in V : \phi(v) = k\}| = n_k$. Note that l is deterministic on S, since we know

the values of l on the non-ambiguous vertices in S. We reparametrize the stochastic blockmodel as SBM$(\vec{n}, \vec{m}, B)$ by including vector $\vec{m}$ into the parametrization. This definition SBM$(\vec{n}, \vec{m}, \mathcal{L}, B)$ is not stochastically different from the definition of SBM$(\vec{n}, \mathcal{L}, B)$, and is suitable for the scenario of vertex nomination.

Under the SBM$(\vec{n}, \vec{m}, \mathcal{L}, B)$ framework, we assume that only the vertices from the first block V_1 are interesting. We define the vertex nomination scheme as follows:

Definition 5.1. ***Vertex Nomination Scheme*** The vertex nomination scheme Φ on a stochastic blockmodel is a mapping such that, to each graph $G \sim \text{SBM}(\vec{n}, \vec{m}, \mathcal{L}, B)$, associates a linear ordering of the ambiguous vertices in V, denoted as a list $(\Phi_G(1), \Phi_G(2), \ldots, \Phi_G(n-m))$.

5.1.2 The Evaluation Criterion

The metric to evaluate a vertex nomination scheme Φ is mean average precision of Φ, which we define here. For simplicity and without loss of generality, suppose the block we are interested in is U_1, which has cardinality $n_1 - m_1$. For any graph G with vertex set V, and for any integer $j \in [n-m]$, the precision at depth j of Φ for G is defined as

$$\text{PD} := \frac{|\{\Phi_G(i) : i \in [j]\} \cap U_1|}{j}. \tag{5.1}$$

The precision at depth j denotes the fraction of the first j vertices on the nomination list that are in U_1.

We also define the pure average precision of Φ as

$$\mathrm{AP}(\Phi) = \frac{1}{n_1 - m_1} \sum_{j=1}^{n_1 - m_1} \left(\frac{|\{\Phi_G(i) : i \in [j]\} \cup U_1|}{j}\right). \tag{5.2}$$

The pure average precision is the average precision among the "foremost" positions in the list – "foremost" in the sense that these would be precisely the positions having vertices of U_1, if the list had all ambiguous vertices of U_1 at the top of the list.

The mean average precision $\mathrm{MAP}(\Phi) \in [0, 1]$ is defined as the expected value of $\mathrm{AP}(\Phi)$ over the probability space associated with the underlying stochastic block-model. The higher $\mathrm{MAP}(\Phi)$ indicates better performance of the vertex nomination scheme. A nomination scheme, which uniformly at random selects an ordering, has MAP $\frac{n_1 - m_1}{n - m}$. This is the chance MAP.

Note that our definition of average precision, which is a pure average precision, is slightly different from the definition used in the information retrieval community; they define the average precision as below,

$$\mathrm{AP'}(\Phi) = \frac{1}{n_1 - m_1} \sum_{j=1}^{n - m} \left(\mathbb{1}_{\Phi_G(j) \in U_1} \frac{|\{\Phi_G(i) : i \in [j]\} \cup U_1|}{j}\right), \tag{5.3}$$

where $\mathbb{1}$ is the indicator function. This definition of average precision can be considered as an integral of the precision over recall.

Here, the mean average precision (MAP) we use is the expected value of the pure average precision in Definition 5.2, because this metric adapts our proof for theoretical performance guarantees.

5.2 The Canonical Vertex Nomination Scheme

In this section, we define the canonical vertex nomination scheme. Let $\binom{U}{n_1-m_1,n_2-m_2,\ldots,n_K-m_K}$ denote all possible $\binom{n-m}{n_1-m_1,n_2-m_2,\ldots,n_K-m_K}$ partitions of vertices in U partitioned into sets $\{U_1, U_2, \ldots, U_K\}$, with respective cardinalities $\{n_1 - m_1, n_2 - m_2, \ldots, n_K - m_K\}$. Define the following constants: for all $i \neq j$, let $e_{i,j}^{G,l}$ denote the number of edges in G with one endpoint in $\{w \in V : l(w) = i\}$ and the other endpoint in $\{w \in V : l(w) = j\}$. Define the constant $c_{i,j}^{G,l} := n_i n_j - e_{i,j}^{G,l}$. Let $e_{i,i}^{G,l}$ denote the number of edges in G with both endpoints in $\{w \in V : \phi(w) = i\}$. Define the constant $c_{i,i}^{G,l} := \binom{n_i}{2} - e_{i,i}^{G,l}$.

Note that the underlying sample space Θ for our vertex nomination task is a bivariate sample space $\Theta = (\mathcal{G}, \mathcal{L})$. Let $\mathcal{L}^{(i)}$ denote the set of all possible partitions such that the i-th ambiguous vertex is interesting: $\mathcal{L}^{(i)} := \{l \in \mathcal{L} | l(i) \in V_1\}$. For canonical vertex nomincation scheme, we are interested in the conditional probability of the i-th vertex belonging to U_1

$$\mathbb{P}(\mathcal{L}^{(i)}|G), \tag{5.4}$$

given a graph $G \sim \text{SBM}(\vec{n}, \vec{m}, \mathcal{L}, B)$. We can further simplify Equation 5.4 as

$$\begin{aligned} \mathbb{P}(\mathcal{L}^{(i)}|G) &= \frac{\mathbb{P}(G, \mathcal{L}^{(i)})}{\mathbb{P}(G)} = \frac{\sum_{l \in \mathcal{L}^{(i)}} \mathbb{P}(G, l)}{\sum_{l \in \mathcal{L}} \mathbb{P}(G, l)} & (5.5) \\ &= \frac{\sum_{l \in \mathcal{L}^{(i)}} \prod_{i=1}^{K} \prod_{j=i}^{K} (B_{i,j})^{e_{i,j}^{G,l}} (1 - B_{i,j})^{c_{i,j}^{G,l}}}{\sum_{l \in \mathcal{L}} \prod_{i=1}^{K} \prod_{j=i}^{K} (B_{i,j})^{e_{i,j}^{G,l}} (1 - B_{i,j})^{c_{i,j}^{G,l}}}. & (5.6) \end{aligned}$$

Equation 5.6 can be directly computed, since we assume the parameters B, $\vec{n}$ and $\vec{m}$ are known.

The canonical vertex nomination scheme Φ^C nominates the ambiguous vertices in decreasing order of the conditional probability via

$$\mathbb{P}(\Phi_G^C(1)|G) \geq \mathbb{P}(\Phi_G^C(2)|G) \geq \cdots \geq \mathbb{P}(\Phi_G^C(n-m)|G). \tag{5.7}$$

We show that the canonical nomination scheme Φ^C is proven to be the best possible nomination scheme using the mean average precision.

Theorem 5.1. *(Fishkind et al. [40])*

For every vertex nomination scheme Φ, $\text{MAP}(\Phi^C)$ is greater than or equal to $\text{MAP}(\Phi)$.

Proof. For $i = 1, 2, \ldots, n_1 - m_1$, let $\gamma_i := \frac{1}{n_1 - m_1} \sum_{j=i}^{n_1-m_1} \frac{1}{j}$. For each $i = n_1 + 1, n_1 + 2, \ldots, n$, let $\gamma_i := 0$. The sequence $\{\gamma_i\}_{i=1}^n$ is nonnegative and non-increasing. Then if $\{\kappa_i\}_{i=1}^n$ is any non-increasing and non-negative sequence, and let $\{\kappa_i^{'}\}_{i=1}^n$ be any permutation of $\{\kappa_i\}_{i=1}^n$. The following holds:

$$\sum_{i=1}^{n} \gamma_i \kappa_i^{'} \leq \sum_{i=1}^{n} \kappa_i \gamma_i. \tag{5.8}$$

Recall the definition of mean average precision of Φ.

$$\begin{aligned}
\text{MAP}(\Phi) &= \mathbb{E}(\sum_{i=1}^{n} \gamma_i \mathbb{1}_{\mathcal{L}^{(i)}}) = \sum_{i=1}^{n} \gamma_i \mathbb{P}(\mathcal{L}^{(i)}) && (5.9)\\
&= \sum_{i=1}^{n} \gamma_i (\sum_{G \in \mathcal{G}} \mathbb{P}(G)\mathbb{P}(\mathcal{L}^{(i)}|G)) && (5.10)\\
&= \sum_{G \in \mathcal{G}} \mathbb{P}(G)(\sum_{i=1}^{n} \gamma_i \mathbb{P}(G)\mathbb{P}(\mathcal{L}^{(i)}|G))) && (5.11)\\
&\leq \sum_{G \in \mathcal{G}} \mathbb{P}(G)(\sum_{i=1}^{n} \gamma_i \mathbb{P}(\Pi_{\Phi^C}^{(i)}|G)) && (5.12)\\
&= \sum_{i=1}^{n} \gamma_i \mathbb{P}(\mathcal{L}^{\{(i),C\}}) = \mathbb{E}(\sum_{i=1}^{n} \gamma_i \mathbb{1}_{\mathcal{L}_{\Phi^C}^{(i)}}) && (5.13)
\end{aligned}$$

□

Indeed the canonical vertex nomination provides the best possible performance for nomination. Its role is analogous to the role of Bayes classifier, which provides the lowest possible error in supervised classification framework.

5.3 The Likelihood Maximization Vertex Nomination Scheme

One popular and principled nomination scheme is the likelihood maximization vertex nomination scheme Φ^L proposed in [40]. This approach consists of two main steps: estimate the maximum likelihood estimator of the unknown parameter l with a prior uniform distribution over $\mathcal{L}$, and compute a geometric mean of a collection of numbers in order to nominate. Using the same notation as above, let $\mathcal{L}$ denote the set of all possible block assignments $l : V \to \{1, 2, \ldots, K\}$, where l agrees with b on S, i.e., $l(S) = b(S)$, and $|v_i \in V : l(i) = k| = n_k$ for $k \in [K]$. For any $l \in \mathcal{L}$, define $e_{k,q}(l)$ to be the number of edges in graph G with one endpoint in $\{w_i \in V : l(i) = k\}$, and the other endpoint in $\{w_i \in V : l(i) = q\}$, for $k = 1, 2, \ldots, K$ and $q = k+1, k+2, \ldots, K$. Define the constant $c_{k,q}(l) := n_k n_q - e_{k,q}(l)$. Define $e_{k,k}(l)$ to be the number of edges in G with both endpoints in $\{w_i \in W : l(w_i) = k\}$, and $c_{k,k}(l) := \binom{n_k}{2} - e_{k,k}(l)$.

With the constants defined as above, we can construct the likelihood function of

the variate (l, G) as follows:

$$\mathbb{P}(l, G) := \frac{1}{|\mathcal{L}|}\mathbb{P}(G|l) = \frac{1}{|\mathcal{L}|}\prod_{k=1}^{K}\prod_{q=k}^{K}(B_{k,q})^{e_{k,q}(l)}(1 - B_{k,q})^{c_{k,q}(l)}. \tag{5.14}$$

Our goal is to find the maximum likelihood estimator $\hat{l} \in \mathcal{L}$, such that the likelihood function in Equation 5.14 is maximized. Since $\frac{1}{|\mathcal{L}|}$ is a constant, we only concern ourself with the part $\prod_{k=1}^{K}\prod_{q=k}^{K}(B_{k,q})^{e_{k,q}(l)}(1 - B_{k,q})^{c_{k,q}(l)}$. Taking the logarithms of the likelihood function, and omit terms which do not contain l, we have:

$$\hat{l} := \arg\max_{l\in\mathcal{L}} \mathbb{P}(l, G) = \arg\max_{l\in\mathcal{L}} \sum_{k=1}^{K}\sum_{q=k}^{K} e_{k,q}(l) \log\left(\frac{B_{k,q}}{1 - B_{k,q}}\right) \tag{5.15}$$

$$= \arg\max_{l\in\mathcal{L}} \sum_{\{w_i,w_j\}\in\binom{V}{2}} \mathbb{1}_{\{w_i \text{ and} w_j \text{ are adjacent}\}} \log\left(\frac{B_{l(i),l(j)}}{1 - B_{l(i),l(j)}}\right) \tag{5.16}$$

Equation 5.16 is solved using seeded graph matching algorithm [39].

The second main step of the likelihood maximization vertex nomination scheme Φ^L is to compute a geometric mean of a collection of numbers. For any pair of two vertices $v_i, v_j \in V$ such that $\hat{l}(i) = 1$ and $\hat{l}(j) \neq 1$. Define $\hat{l}_{v_i\to v_j} \in \mathcal{L}$ such that $\hat{l}_{v_i\to v_j}$ agrees with $\hat{l}$ for all vertices except that $\hat{l}_{v_i\to v_j}(j) = 1$ and $\hat{l}_{v_i\to v_j}(i) = \hat{j}$. We consider the geometric mean of ratio of likelihoods

$$\left(\prod_{v_j\in V:\hat{l}(j)\neq 1} \frac{\mathbb{P}(\hat{l}_{v_i\to v_j}, G)}{\mathbb{P}(\hat{l}, G)}\right)^{\frac{1}{n-n_1+m_1}} \tag{5.17}$$

as a measure for ordering that $b(i) = 1$. The geometric mean

$$\left(\prod_{v_j\in V:\hat{l}(j)\neq 1} \frac{\mathbb{P}(\hat{l}_{v_i\to v_j}, G)}{\mathbb{P}(\hat{l}, G)}\right)^{\frac{1}{n_1-m_1}} \tag{5.18}$$

is a measure for ordering that $l(i) \neq 1$.

The likelihood maximization vertex nomination scheme Φ^L satisfies $\Phi^L_G(1), \Phi^L_G(2), \ldots, \Phi^L_G(n_1 - m_1)$ with $\hat{l}(i) = 1$ in increasing order of the geometric mean in Equation 5.17, and $\Phi^L_G(n_1 - m_1 + 1), \Phi^L_G(n_1 - m_1 + 2), \ldots, \Phi^L_G(n - m)$ with $\hat{l}(j) \neq 1$ in decreasing order of the geometric mean in Equation 5.18.

Empirically the likelihood maximization vertex nomination scheme achieves good performance in terms of accuracy. While likelihood maximization vertex nomination scheme Φ^L is practical to implement for graph inference, it is limited on the order of a thousand of vertices, since the state-of-the-art SGM algorithm in solving Equation 5.16 has complexity $O(n^3)$.

5.4 The Spectral Partitioning Vertex Nomination Scheme

The spectral partitioning vertex nomination scheme Φ^S incorporates the technique of adjacency spectral embedding introduced in Chapter 2.5, and nominates the ambiguous vertices based on the increasing order of the Euclidean or Mahalonobis distance away from a cluster centroid. The spectral partitioning vertex nomination scheme Φ^S does not assume that the block communication probability matrix B and the block size $\{n_1, n_2, \ldots, n_K\}$ are known. It only assumes the knowledge of the number of blocks K and the rank d of the block communication probability matrix B. This scheme consists of two steps. The first step uses adjacency spectral embedding

to transform the graph into d-dimensional Euclidean embedding $\hat{\mathcal{X}} \in \mathbb{R}^{n \times d}$. The second step is to cluster the rows of $\hat{\mathcal{X}}$ into K clusters. Suppose c is the centroid of the cluster that is associated with the most vertices known to be in the block of interest V_1. The spectral partitioning vertex nomination scheme Φ^S satisfies the ordering of vertices $\Phi^S_G(1), \Phi^S_G(2), \ldots, \Phi^S_G(n-m)$ in increasing order of the distance between c and their corresponding rows in $\hat{\mathcal{X}}$. The metric (for example, Euclidean or Mahalonobis distance) used to rank the vertices depends on which clustering algorithm is employed. The K-means clustering algorithm nominates the vertices based on the Euclidean distance to the centroids, while the model-based clustering algorithm (Mclust) nominates the vertices using the Mahalonobis distance to the centroids. In Section 5.6, we use the Mclust algorithm for simulation and real data experiments due to its empirical improvement over the K-means algorithm.

Theoretically, the spectral partitioning vertex nomination scheme Φ^S scheme is shown to cluster vertices perfectly in the limit [63]. It follows that $\text{MAP}(\Phi^S)$ converges to 1, and it thus nominates perfectly asymptotically [40]. The spectral partitioning vertex nomination scheme Φ^S is also computationally tractable for large graphs. However, it usually does not perform well on small-sized graphs with tens of vertices. Recall in Chapter 3 that the success of adjacency spectral embedding requires the knowledge of the embedding dimension, thus the spectral partitioning vertex nomination scheme also requires knowing the embedding dimension. In practice, this piece of information is often unknown.

5.5 The Canonical Sampling Vertex Nomination Scheme

So far we have introduced several vertex nomination schemes: the canonical nomination scheme Φ^C, the likelihood maximization nomination scheme Φ^L, and the spectral partitioning nomination scheme Φ^S, for graphs realized from a stochastic blockmodel SBM$(\vec{n}, \vec{m}, \mathcal{L}, B)$. The above schemes possess theoretical performance guarantees in terms of the mean average precision as shown in [40]. In this section, we propose the canonical sampling nomination scheme.

5.5.1 Motivation

The canonical vertex nomination scheme, denoted by Φ^C, is the optimal vertex nomination scheme, because its MAP is better than or equal to that of any other vertex nomination scheme Φ [40], and we therefore term it the "gold standard". However Φ^C is computationally intractable, because current computation methods seem to have complexities to grow exponentially with the number of ambiguous vertices. Hence Φ^C is not practical for graph inference. Consider the conditional probability space of $\mathcal{L}|G$. For each $l \in \mathcal{L}$, we may try to sample directly from the conditional space $\mathcal{L}|G$ under the following probability

$$\mathbb{P}(l|G) = \frac{\prod_{i=1}^{K}\prod_{j=i}^{K}(B_{i,j})^{e_{i,j}^{G,l}}(1-B_{i,j})^{c_{i,j}^{G,l}}}{\sum_{l'\in\mathcal{L}}\prod_{i=1}^{K}\prod_{j=i}^{K}(B_{i,j})^{e_{i,j}^{G,l'}}(1-B_{i,j})^{c_{i,j}^{G,l'}}}. \tag{5.19}$$

Then we can use this proportion to estimate the conditional probability in Equation 5.4. Note that the denominator in Equation 5.19 is intractable to calculate.

The likelihood maximization vertex nomination scheme Φ^L, although not the best possible scheme, achieves great performance. It uses the state-of-the-art algorithm for solving the seeded graph matching (SGM) problem. Empirically the likelihood maximization vertex nomination scheme Φ^L performs reasonably well, and it is practical to implement for graph inference. However, it is limited on the order of a thousand of vertices, since the state-of-the-art SGM algorithm has complexity $O(n^3)$.

The spectral partitioning vertex nomination scheme Φ^S, incorporating the technique of adjacency spectral embedding and clustering, achieves perfect nomination in the limit. Moreover, this scheme is computationally tractable for big graph data. However, Φ^S is usually not applicable for small-sized graphs on few tens of vertices.

In the age of big data, scalable methodologies with performance guarantee are in high demand. Here, we propose a canonical sampling vertex nomination scheme Φ^{CS}, which not only preserves the "gold standard" property of the canonical vertex nomination scheme Φ^C in the limit, but also scales to big graph data.

5.5.2 The Canonical Sampling Vertex Nomination Scheme

Our goal is to extend the canonical vertex nomination scalable for big graph data, and preserve the optimal performance guarantee of this scalable version to the nomination scheme. While direct calculating Equation 5.4 is not feasible, a natural approach is to approximate the conditional probability in Equation 5.4 via Markov Chain Monte Carlo (MCMC) approaches. The Metropolis-Hastings algorithm is an MCMC method for approximating a distribution that is difficult to directly sample from ([66], [21], [48], [82]).

The classical setting of MCMC allows us to sample from the distribution of $\mathbb{P}(\mathcal{L}|G)$. However, in the vertex nomination setting, we intend to generate samples from $\mathbb{P}(\mathcal{L}^{(i)}|G)$. We next describe how to approximate the canonical vertex nomination scheme. Let $\{l^t\}_{t=0,1,2,\ldots}$ denote the stochastic process of the partition on the ambiguous vertices at time t. Let l' denote the candidate partition. Let $Burnin$ denote the number of burn-in in the Metropolis-Hastings algorithm. Let T denote the total number of samples. We present our proposed canonical sample vertex nomination in Algorithm 5.

Our proposal candidate of block assignment l' is created by selecting and swapping two vertices. The first vertex v_i is selected uniformly at random from U_1 and the second vertex v_j is selected uniformly at random from $U \backslash U_1$. Then we exchange the

Algorithm 5 Canonical Sampling Vertex Nomination Φ^{CS}

Uniformly at random select partition $l^{-Burnin} \in \Pi$ ▷ Initialization

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

Select a vertex $v_i \in U_1$ and $v_j \in U \backslash U_1$ ▷ Generate candidate by swap

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

Compute $a = \frac{\mathbb{P}[G, l']}{\mathbb{P}[G, l^{t-1}]}$

Perform a Bernoulli trial with parameter $\min\{1, a\}$

if Success **then**

$l^t \leftarrow l'$ ▷ Accept new block assignment

else Failure

$l^t \leftarrow l^{t-1}$ ▷ Reject new block assignment

end if

end for

memberships of v_i and v_j. The act of exchanging two vertices in a partition is called a **swap**, which we denote by $l(v_i, v_j)$. Hence the candidate block assignment at time t is generated via a swap of the block assignment at time $t-1$, i.e., $l' := l^{t-1}(v_i, v_j) \in \mathcal{L}$. Abusing notations a little bit, we use l to denote the state of the block assignment at time $t-1$, and l' denote the candidate at time t. We repeatedly swap vertices to generate candidates of block assignment. Then we perform a Bernoulli trial with parameter $\min\left\{ 1, \ a := \frac{\prod_{i=1}^{K}\prod_{j=i}^{K}(B_{i,j})^{e_{i,j}^{G,l'}}(1-B_{i,j})^{c_{i,j}^{G,l'}}}{\prod_{i=1}^{K}\prod_{j=i}^{K}(B_{i,j})^{e_{i,j}^{G,l}}(1-B_{i,j})^{c_{i,j}^{G,l}}} \right\}$, which is our acceptance-rejection regime. If it is a success then the Markov chain transitions from state l to state l', and if it is a failure then the Markov chain transition is to just remain at state l. We repeat this procedure until some convergence criterion is met or a predetermined number of iterations T is reached. At last, we estimate the conditional probability $\mathbb{P}(\mathcal{L}^{(i)}|G)$ as

$$\widehat{\mathbb{P}}((\mathcal{L}^{(i)}|G)) := \frac{N_T(\mathcal{L}^{(i)})}{T},$$

where $N_T(\mathcal{L}^{(i)})$ is the number of times the i-vertex is accepted to be in V_1.

5.5.3 Performance Guarantee

For any $l, l' \in \mathcal{L}$, denote the Markov transition probability from state l to state l' by $\mathbb{P}(l \to l')$. We define this transition probability as follows: If l and l' differ by more than two vertices, then $\mathbb{P}(l \to l') := 0$. If l and l' differ on exactly two vertices,

then

$$\mathbb{P}(l \to l') := \frac{1}{\binom{n-m}{2} - \sum_{i=1}^{K} \binom{n_i - m_i}{2}} \cdot \min \left\{ 1, \ \frac{\prod_{i=1}^{K} \prod_{j=i}^{K} (B_{i,j})^{e_{i,j}^{G,l'}} (1 - B_{i,j})^{c_{i,j}^{G,l'}}}{\prod_{i=1}^{K} \prod_{j=i}^{K} (B_{i,j})^{e_{i,j}^{G,l}} (1 - B_{i,j})^{c_{i,j}^{G,l}}} \right\}; \tag{5.20}$$

Also set the transition probability of remaining at the state l as $\mathbb{P}(l \to l) := 1 - \sum_{l'' \in \mathcal{L}} \mathbb{P}(l \to l'')$.

Note that for any $l \in \mathcal{L}$, there are exactly $\binom{n-m}{2} - \sum_{i=1}^{K} \binom{n_i - m_i}{2}$ members of $\mathcal{L}$ that differ from l on V by exactly two elements, thus these transition probabilities are nonnegative and sum to 1. Note that for all $l, l' \in \mathcal{L}$ it holds

$$\mathbb{P}(l \to l') \cdot \mathbb{P}(l|G) = \mathbb{P}(l' \to l) \cdot \mathbb{P}(l'|G),$$

thus the limiting distribution of the Markov chain is as desired, $\mathbb{P}(l|G)$ for each $l \in \mathcal{L}$ [5].

Hence, we have the following two theorems.

Theorem 5.2. Regardless of the initialization of l^t, the limiting distribution of the Markov chain l^t converges to the desired distribution $\mathbb{P}(\cdot|G)$.

Hence, in the limit, the canonical sampling vertex nomination scheme Φ^{CS} generates samples from the distribution $\mathbb{P}(\mathcal{L}^{(i)}|G)$. Consequently it naturally follows that the canonical sampling vertex nomination scheme Φ^{CS} converges to the nomination order of the canonical vertex nomination scheme Φ^{C}. Hence the canonical sampling vertex nomination scheme Φ^{CS} approximates the best possible vertex nomination scheme. Theorem 5.2 naturally implies that the canonical sampling vertex

nomination scheme Φ^{CS} achieves optimal MAP= 1 with sufficient number of samples.

Indeed, our proposed vertex nomination scheme Φ^{CS} is the "gold standard" for big graph data.

Theorem 5.3. The MAP of the canonical sampling vertex nomination scheme converges to the MAP of the canonical vertex nomination scheme, $\text{MAP}(\Phi^{CS}) \to \text{MAP}(\Phi^{C})$ given enough number of samples. Hence the canonical sampling vertex nomination scheme approximates the best possible vertex nomination scheme.

5.6 Numerical Experiments

5.6.1 Simulation

5.6.1.1 Number of Samples in Φ^{CS}

In this experiment, we first explore the nomination performance and run time against the number of samples needed in the canonical sampling vertex nomination scheme Φ^{CS}. The parameters for the stochastic blockmodel of $K = 3$ blocks are

$$n - m = \begin{bmatrix} 200 \\ 150 \\ 150 \end{bmatrix}, \quad m = \begin{bmatrix} 20 \\ 0 \\ 0 \end{bmatrix}, \quad B = \begin{bmatrix} .50 & .44 & .47 \\ .44 & .59 & .53 \\ .47 & .53 & .44 \end{bmatrix},$$

Number of samples	1000	10000	100000	500000
MAP	0.80	0.875	0.923	0.945
Runtime	9.5	12.8	45.6	191

Table 5.1: MAP and runtime of Φ^{CS} for different numbers of samples $S \in \{1000, 10000, 100000, 50000\}$. The MAP and runtime are averaged over 200 independent Monte Carlo replicates. The runtime in this table is in seconds.

where $n - m = (n_1 - m_1, n_2 - m_2, n_3 - m_3) \in \mathbb{N}^3$ denotes the number of the ambiguous vertices in U in each block, $M = (m_1, m_2, m_3) \in \mathbb{N}^3$ denotes the number of non-ambiguous vertices in S in each block, and B is the block connectivity probability matrix. The task is to nominate vertices in the first block V_1. We vary the number of samples $Samp \in \{1000, 10000, 100000, 500000\}$, and examine the nomination performance and run time. For each choice of $Samp$, we select 25000 samples for burn-in, and run 200 independent Monte Carlo replicates. We evaluate the performance via MAP. The chance MAP for this experiment is 0.4.

Figure 5.1 demonstrates that the nomination performance improves, as we increase the number of samples $Samp \in \{1000, 10000, 100000, 500000\}$ to approximate $\mathbb{P}(\mathcal{L}^{(i)}|G)$. This is expected due to Theorem 5.2 and Theorem 5.3. The MAP and runtime for each $Samp \in \{1000, 10000, 100000, 500000\}$ are seen in Table 5.1.

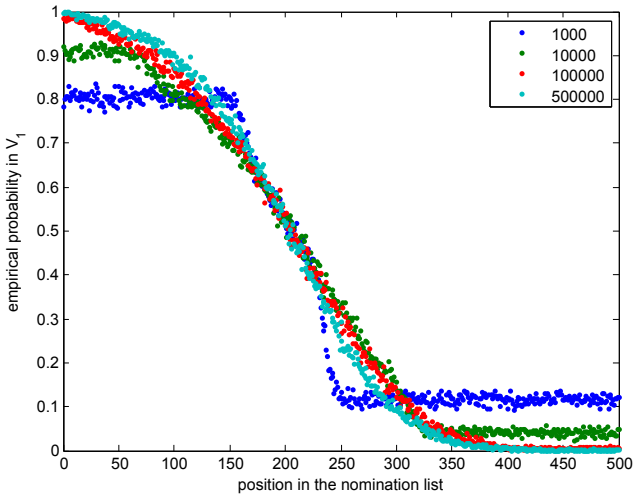


Figure 5.1: Performance of Φ^{CS} using $Samp \in \{1000, 10000, 100000, 50000\}$ number of samples. For each $Samp \in \{1000, 10000, 100000, 50000\}$, the first 25000 samples are burn-in, and we simulate 200 independent Monte Carlo replicates. This plot demonstrates the increase in performance with more samples. The simulation was performed in collaboration with Henry Pao. This figure also appears in Henry Pao's thesis [70].

5.6.1.2 Comparison of Performance on Graphs at Three Scales

For small sized graphs, Φ^C is computationally feasible, while Φ^S performs almost as chance. For mid-sized graphs, Φ^C is no longer feasible, Φ^L performs well, and so does Φ^S. For large-sized graphs, neither Φ^C or Φ^L are computationally feasible. Our proposed canonical sampling vertex nomination scheme Φ^{CS} is computationally feasible for graphs of all small, medium and large scales. In this experiment, we compare Φ^{CS} with the other three theoretically guaranteed nomination schemes on graphs of different scales. For Φ^{CS}, we use 25000 samples for burn-in, and 500000 samples after burn-in.

These simulation experiments are designed to compare the schemes at various scales. Again we have $K = 3$ blocks in a stochastic blockmodel. Define the block connectivity matrix as

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

which can be thought of as a mixture model between a three-block stochastic blockmodel and an Erdos-Renyi model with $p = 0.5$, and larger α reflects more block signal. We define the block size containing the ambiguous vertices to be $(\vec{n} - \vec{m})(\beta) =$

$\beta[4;3;3]$. For the three-scaled experiments, we have the following parameters:

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

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

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

where $\vec{n}-\vec{m}=(n_1-m_1,n_2-m_2,n_3-m_3)\in\mathbb{N}^3$ denotes the number of the ambiguous vertices in V in each block, and $M=(m_1,m_2,m_3)\in\mathbb{N}^3$ denotes the number of the vertices in U in each block. For each small, medium and large sized graph, we ran 50000, 200, and 100 independent Monte Carlo replicates respectively. The chance MAP for all three experiments are 0.4.

Figure 5.2 compares vertex nomination schemes Φ^C, Φ^{CS}, Φ^L, and Φ^S on small sized graphs. Note that Φ^{CS} performs slightly better than Φ^C, but the performance is within 1 standard error. The MAP and run time averaged over 50000 independent Monte Carlo replicates are shown in Tables 5.2 and 5.3. Our canonical sampling vertex nomination scheme Φ^{CS} has the longest run time, because we use 500000 samples to approximate the conditional probability. 500000 is over-sampling, since the number

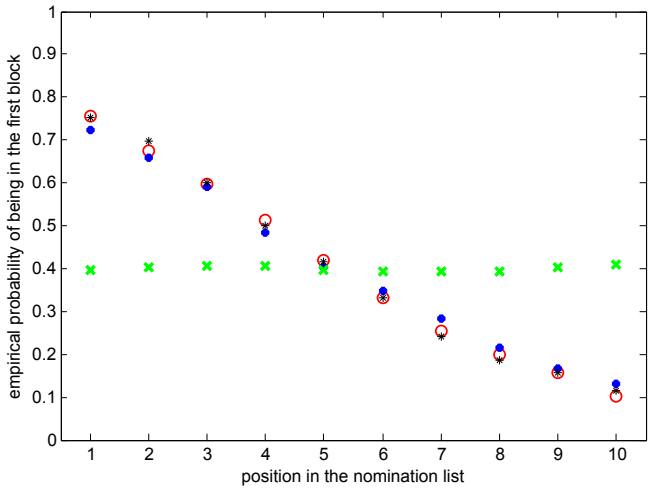


Figure 5.2: Vertex nomination scheme comparison on small graph simulation. We compare four nomination schemes: Φ^C (red), Φ^{CS} (black), Φ^L (blue), and Φ^S (green) on the small scale graph. The simulated graphs have 4 observed vertices and 10 ambiguous vertices. Φ^C, Φ^{CS}, and Φ^L perform similarly and much better than chance, while Φ^S performs as chance. The simulation experiment was performed in collaboration with Donniell Fishkind and Henry Pao. This figure also appears in Henry Pao's thesis [70].

of all possible partition is $\binom{10}{4,3,3} = 4200$. If we reduce the number of samples, the run time of Φ^{CS} will improve.

Figure 5.3 compares the performance of Φ^{CS}, Φ^L, and Φ^S on mid-sized graphs, while Φ^C is not applicable in this experiment. The MAP and run time averaged over 200 independent Monte Carlo replicates are shown in Tables 5.2 and 5.3. All three schemes perform significantly better than chance. Φ^{CS} outperforms Φ^L, and is

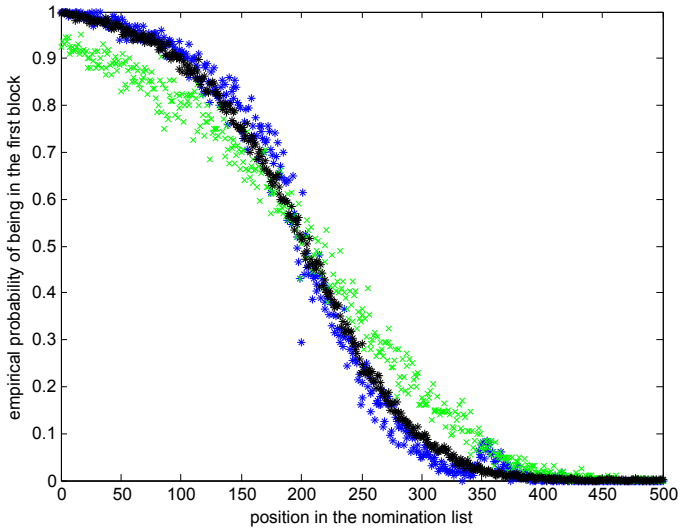


Figure 5.3: Vertex nomination scheme comparison on medium graph simulation. We compare three nomination schemes: Φ^{CS} (black), Φ^{L} (blue), and Φ^{S} (green) on mid-sized graphs. Φ^{C} is not applicable in this experiment, because of its exponential-growth complexity. All three schemes perform significantly better than chance. Φ^{CS} outperforms Φ^{L}, and is significantly superior to Φ^{S}. The simulation experiment was performed in collaboration with Donniell Fishkind and Henry Pao. This figure also appears in Henry Pao's thesis [70].

significantly superior to Φ^{S}. Moreover, Φ^{CS} is more than 30% faster than Φ^{L}.

Figure 5.4 compares the performance of Φ^{CS} and Φ^{S} on large-sized graphs. Φ^{C} and Φ^{L} are not applicable in this experiment, because of the exponential-growth complexity and $O(n^3)$ complexity respectively. The MAP and runtime averaged over 100 independent Monte Carlo replicates are shown in Tables 5.2 and 5.3. Both Φ^{CS} and Φ^{S} perform significantly better than chance. Φ^{S} outperforms Φ^{CS} by about 5%

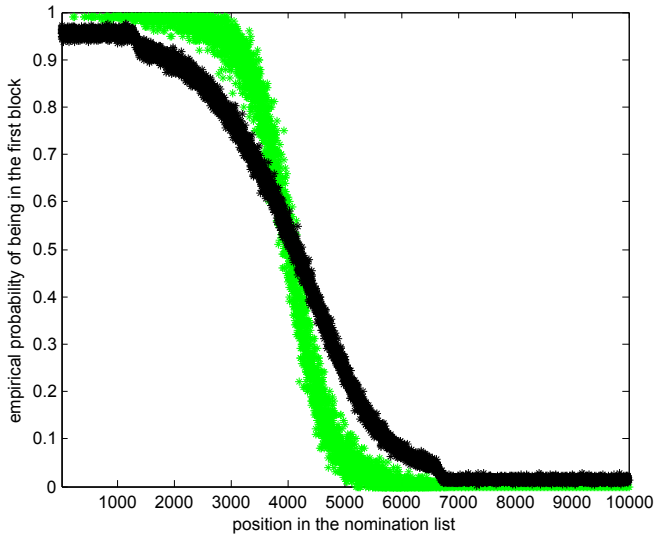


Figure 5.4: Vertex nomination scheme comparison on large graph simulation. We compare two nomination schemes: Φ^{CS} (black) and Φ^{S} (green) on large-sized graphs. Both Φ^{CS} and Φ^{S} perform significantly better than chance, and Φ^{S} outperforms Φ^{CS}. The simulation experiment was performed in collaboration with Donniell Fishkind and Henry Pao. This figure also appears in Henry Pao's thesis [70].

in terms of MAP. Both schemes have similar run time.

5.6.2 Real Data

We apply our proposed canonical sampling vertex nomination scheme Φ^{CS} on real datasets, and compare its performance with Φ^{L} and Φ^{S}. The canonical vertex nomination Φ^{C} is only feasible for small-sized graphs, so it is not suitable for the real data experiments. The vertex nomination setting assumes that the connectivity matrix Λ and the size of the ambiguous vertices in each block $\{n_1, n_2, \ldots, n_K\}$ are

MAP	Φ^C	Φ^L	Φ^{CS}	Φ^S
Small	0.6948	0.6515	0.6993	0.3991
Medium	N\A	0.9303	0.9452	0.7330
Large	N\A	N\A	0.9281	0.9859

Table 5.2: MAP of simulation experiments at three graph scales.

Runtime	Φ^C	Φ^L	Φ^{CS}	Φ^S
Small	1.4	.04	138	.02
Medium	N\A	286	191	.8
Large	N\A	N\A	546	534

Table 5.3: Runtime (in seconds) of simulation experiments at three graph scales.

known. Recall that Φ^{CS} and Φ^{S} use the knowledge of such parameters. In the real data experiments, we estimate the connectivity matrix Λ using the unambiguous vertices in S which are already observed to be interesting or uninteresting. In addition, we also compare the vertex nomination performance with a community detection methodology for mixed membership stochastic blockmodel [4], where we modify it for the task of vertex nomination [40] and denote it by Φ^{M}. Its performance is plotted in pink in the following figures.

5.6.2.1 The Political Blog Sphere

The political blog sphere was created from web blogs during the 2004 US presidential election [3]. In this dataset, the blogs are the vertices, and are connected if there is a web-link between any pair of blogs. The original dataset contains 1490 blogs with There are non-isolated 1224 blogs in this graph. Each blog belongs to a category of either conservative (636) or liberal (588). We binarize, symmetrize and make the blog graph hollow. The graph is visualized in Figure 5.5. The block signal is strong, as shown in the adjacency matrix in Figure 5.6.

Our objective is to nominate liberal blogs. For each Monte Carlo replicate, we randomly select $\vec{m} = [80, 80]^T$ vertices from the conservative and liberal categories respectively, and estimate the connectivity matrix B using those vertices. We repeat this experiment over 1000 independent Monte Carlo replicates. The chance MAP for this data set is 0.4774.

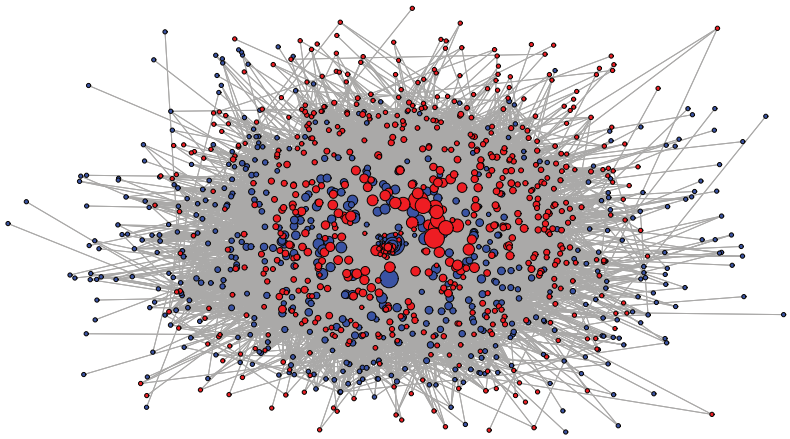


Figure 5.5: The political blog sphere data. The vertices are the web-blogs, and the edges exist if there is a web-link connecting two blogs. Each blog is either conservative (red) or liberal (blue). Our vertex nomination task here is to nominate the liberal blogs. The experiment was performed in collaboration with Henry Pao. This figure also appears in Henry Pao's thesis [70].

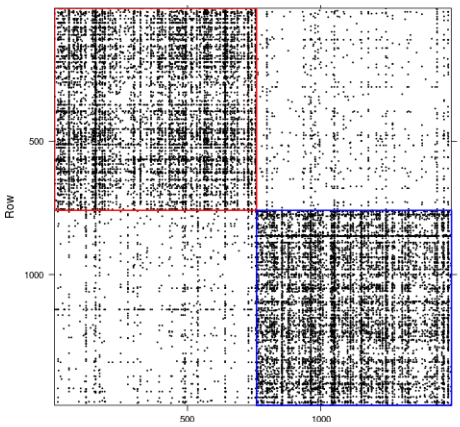


Figure 5.6: The adjacency matrix of the political blog graph with vertices sorted according to their political orientations. A strong two-block signal is reflected in the adjacency matrix. The red block corresponds to the conservative blogs, and the blue block corresponds to the liberal blogs.

Blog Sphere	Φ^{CS}	Φ^{L}	Φ^{S}	Φ^{M}
MAP	0.9317	0.8922	0.7856	0.5429

Table 5.4: MAP of the blog sphere experiment. The chance MAP is 0.4774.

Figure 5.7 presents the performance of Φ^{CS} (black), Φ^{L} (blue), Φ^{S} (green), and Φ^{M} (pink) on the blog sphere. All nomination schemes outperform chance significantly. Φ^{CS} performs the best. The MAP and runtime averaged over 1000 independent Monte Carlo replicates for three nomination schemes are presented in Table 5.4.

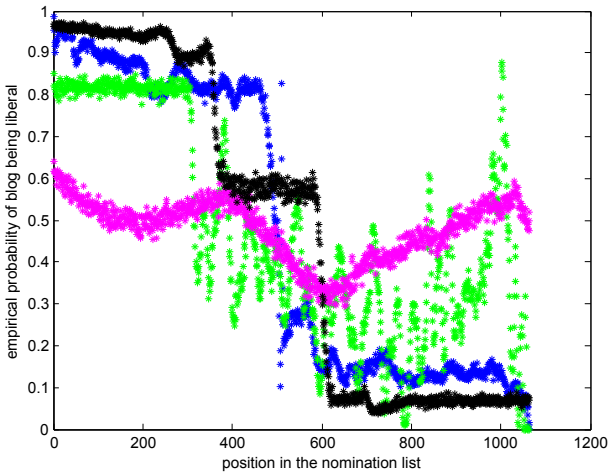


Figure 5.7: We compare the performance of Φ^{CS} (black), Φ^{L} (blue), Φ^{S} (green), and Φ^{M} (pink) on the blog data. Φ^{CS} outperforms all other nomination schemes. All schemes perform better than chance. The experiment was performed in collaboration with Henry Pao. This figure also appears in Henry Pao's thesis [70].

5.6.2.2 The Movie Network

The movie network is created via scrapping movie info-boxes from Wikipedia (see [40] for detail). This netwrok consists of 619 non-isolated movies as vertices, and the edges exist if any two movies share a common director producer or actor. Each movie belongs to exactly one category: comedies (227), action thrillers (157) and dramas (235). We binarize, symmetrize and make the movie network hollow. The movie network is visualized in Figure 5.8.

This network is not exactly a stochastic blockmodel, since it has a weak block signal, as reflected in Figure 5.9. Our objective is to nominate the comedy movies. For each Monte Carlo replicate, we randomly select $\vec{m} = [30, 30, 30]^T$ vertices from each movie category respectively, and estimate the connectivity matrix B using those vertices. We repeat this experiment over 1000 independent Monte Carlo replicates. The chance MAP for this data set is 0.3724.

Figure 5.10 demonstrates the performance of Φ^{CS}, Φ^{L}, Φ^{S} and Φ^{M} on the movie network. Φ^{L} slightly outperforms Φ^{CS}, and both perform significantly better than chance, while Φ^{S} and Φ^{M} perform just as chance. The MAP and runtime averaged over the 1000 independent Monte Carlo replicates are displayed in Table 5.5. The movie network is not exactly from a stochastic blockmodel generating mechanism. However the superior performance of Φ^{CS} indicates its robustness to model misspecification, and thus its practical value for real data inference.

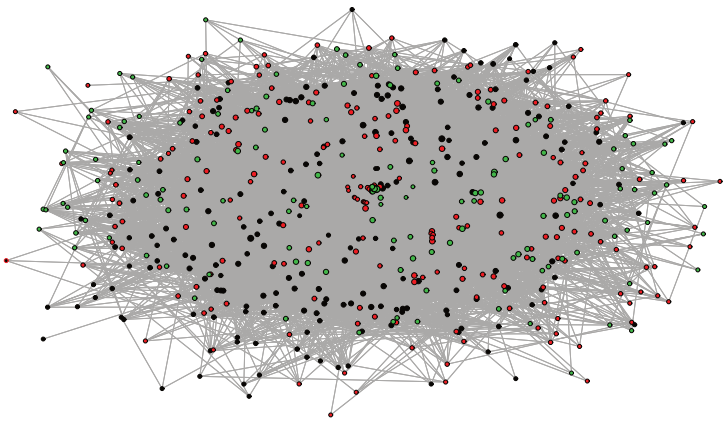


Figure 5.8: The movie network. The vertices are movies, and the edges exist if any two movies have a common director producer or actor. Each movie is categorized as comedies (black), action thrillers (green) or dramas (red). Our vertex nomination task here is to nominate comedies. The experiment was performed in collaboration with Henry Pao. This figure also appears in Henry Pao's thesis [70].

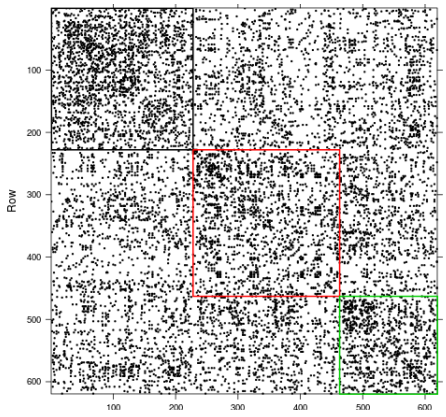


Figure 5.9: The adjacency matrix of the movie network with vertices sorted according to their movie categories. This network does not have a strong block signal. The black block corresponds to the comedies, the red block corresponds to the dramas, and the green block corresponds to the action thrillers.

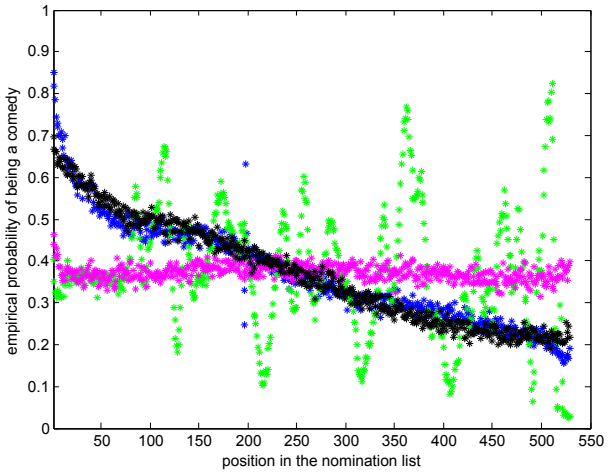


Figure 5.10: We compare the performance of Φ^{CS} (black), Φ^{L} (blue), Φ^{S} (green) and Φ^{M} (pink) on the movie data. Φ^{CS} is almost as good as Φ^{L}. The movie data is not a stochastic blockmodel as seen in its adjacency structure. All schemes perform better than chance. However the superior performance of Φ^{CS} indicates its robustness to model misspecification and practical value for real data inference. The experiment was performed in collaboration with Henry Pao. This figure also appears in Henry Pao's thesis [70].

Movie Network	Φ^{CS}	Φ^{L}	Φ^{S}	Φ^{M}
MAP	0.5707	0.5814	0.3764	0.3766

Table 5.5: MAP of the movie network experiment. The chance MAP is 0.3724.

5.7 Discussion

In this chapter, we propose a scalable canonical sampling vertex nomination scheme, explain its sampling procedure, and prove its theoretical guarantees [19]. We also compare the canonical sampling vertex nomination scheme with several other vertex nomination schemes: the canonical vertex nomination scheme, the likelihood maximization likelihood vertex nomination scheme, and the spectral partitioning vertex nomination scheme proposed from our recent work in [40]. All the nomination schemes are constructed within the framework of the stochastic blockmodel SBM$(\vec{n}, \vec{m}, \mathcal{L}, B)$. This model assumption allows the principled development of the schemes and guarantees their theoretical performance.

The canonical vertex nomination scheme is the best possible scheme applicable to graphs of a few tens of vertices. The likelihood maximization vertex nomination scheme employs the state-of-the-art seeded graph matching algorithm of complexity $O(n^3)$, and empirically demonstrates to have superior nomination performance. The spectral partitioning vertex nomination scheme is simple and effective, applicable on very large graphs. The canonical sampling vertex nomination scheme, inspired by the canonical vertex nomination scheme, not only preserves the "gold standard" property of the canonical vertex nomination scheme, but also scales to big graph data. It uses the Metropolis-Hastings algorithm to approximate a conditional probability that is the essence of the canonical vertex nomination scheme.

The effectiveness in terms of the mean average precision and run time of the

likelihood maximization scheme and the canonical sampling scheme are evidently demonstrated in both simulation studies and real data experiments. The canonical sampling vertex nomination scheme is our first step towards making effective vertex nomination schemes scalable for big graphs. Some open questions such as: how many samples are needed to guarantee convergence, or when does convergence happen, are worth pondering and investigating. Statistical inference on big graph data is of incredible importance nowadays. It is also worth noting that the likelihood maximization vertex nomination scheme, which uses the state-of-the-art seeded graph matching algorithm, demonstrates to perform well on simulation and real data. However due to its $O(n^3)$ complexity, the likelihood maximization vertex nomination scheme is limited to be practical on mid-sized graphs. Recent work on scaling the seeded graph matching algorithm to big graphs [59] may motivate a scalable version of the likelihood maximization vertex nomination scheme. We are confident that a new generation of efficient and scalable vertex nomination schemes will greatly contribute to the research of pattern recognition on random graph.

Chapter 6

Seeded Graph Matching and Large Seeded Graph Matching

Joint graph inference considers using information from multiple graphs, and proceeds with inference in the joint space of the graphs. In this dissertation, we are particularly concerned about joint graph inference simultaneously on two graphs. Under the joint graph inference framework, we are concerned with the problem of seeded graph matching. The seeded graph matching problem seeks a bijection, which minimizes the number of edge disagreements between two graphs with additional constraint that the correspondence between a subset of vertices is known. Recent advances in developing theories and applications of seeded graph matching are gaining increasing attention from fields such as statistics, neuroscience, computer vision, and text analysis.

Currently, the state-of-the-art seeded graph matching has a cubic complexity in the number of vertices. In the age of big data, the computational limitation hinders the applicability of seeded graph matching for large graphs. This motivates us to propose a scalable seeded graph matching algorithm for large graphs, namely the large seeded graph matching algorithm. This algorithm utilizes a divide-and-conquer approach to parallelize graph matching, and is feasible on graphs of over 10000 vertices.

This chapter starts with introducing and describing the problem of graph matching and seeded graph matching in Sections 6.1 and 6.2. Section 6.3 discusses the motivation, proposes large seeded graph matching algorithm, and proves the theoretical performance guarantee. Section 6.4 presents simulation and real data experiments of seeded graph matching and large seeded graph matching. Section 6.5 investigates future directions for research in seeded graph matching.

6.1 Introduction

The problem of graph matching aims to find an alignment between the vertices across two graphs such that the number of edge disagreements is minimized. The complexity of determining if two graphs are isomorphic is unknown. The generalized (loopy, weighted) graph matching is known to be NP-hard. For an excellent survey on the problem of graph matching, see "30 Years of Graph Matching in Pattern Recognition" [22]. Ever since the graph matching problem was first posed in the late

1970s, a wave of graph matching algorithms have emerged and proved their practical values in various fields including statistics, neuroscience, computer vision, pattern recognition, text processing, and image analysis.

The problem of seeded graph matching was posed in [39]. It is the graph matching problem with the additional constraint that a subset of vertices have their correspondence known a priori. The seeded graph matching problem has received increasing attention and has demonstrated its effectiveness in the random graph inference framework. However its feasibility is limited for matching graphs with less than 1500 vertices. In the age of big data, scalable algorithms are in high demand. We demonstrate our proposed algorithm of seeded graph matching for large graphs works well on large real and synthetic graph data.

6.2 Seeded Graph Matching

In this section, we first formulate the problem of graph matching (GM), then discuss seeded graph matching (SGM), and present the SGM algorithm. Given two graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, GM seeks an alignment between the vertex sets V_1 and V_2 such that the topological structure is best preserved across the graphs. There are two categories of graph matching approaches: bijective and non-bijective. In a bijective graph matching setting, we assume that $|V_1| = |V_2| = n$, and GM seeks a one-to-one correspondence between V_1 and V_2. In a non-bijective setting, we do not

require the assumption of $|V_1| = |V_2| = n$, and GM seeks a (possibly) many-to-many or many-to-one correspondence between V_1 and V_2.

In this dissertation, we focus on the bijective graph matching setting, where the vertex sets $V_1 = V_2 = V$. In such a setting, one seeks a bijection $\psi : V \to V$ such that ψ minimizes the number of induced edge disagreements. The edge disagreement objective can be written as follows:

$$d(\psi) = |\{(i,j) \in V \times V : [i \sim_{G_1} j, \psi(i) \nsim_{G_2} \psi(j)] \text{ or } [i \nsim_{G_1} j, \psi(i) \sim_{G_2} \psi(j)]\}|. \quad (6.1)$$

Assume that the adjacency matrices for G_1 and G_2 are A_1 and A_2 respectively. Let $P(n)$ be the set of all permutation matrices of order n. Let P be the corresponding permutation matrix of the permutation ψ. Then we can rewrite the objective in Equation 6.1 as the following objective:

$$\min_{P \in P(n)} \|A_1 - PA_2P^T\|_F, \quad (6.2)$$

where $\|\cdot\|_F$ is the Frobenius norm. Denote the trace of a matrix by tr. Equation 6.2 can be rewritten as:

$$\begin{aligned}
\|A_1 - PA_2P^T\|_F^2 &= tr((A_1 - PA_2P^T)^T(A_1 - PA_2P^T)) \\
&= tr(A_1^TA_1 - A_1^TPA_2P^T - PA_2P^TA_1 + PA_2P^TA_2P^T) \\
&= tr(A_1^TA_1) - tr(A_1^TPA_2P^T) - tr(PA_2^TP^TA_1) + tr(PA_2P^TPA_2P^T) \\
&= \|A_1\|_F^2 - tr(A_1^TPA_2P^T) - tr(PA_2^TP^TA_1) + tr(P^TA_2^TA_2P^T) \\
&= \|A_1\|_F^2 - 2tr(A_1^TPA_2P^T) + \|A_2\|_F^2. \quad (6.3)
\end{aligned}$$

The first and last two terms in Equation 6.3 do not depend on P. Thus the problem in Equation 6.2 is equivalent to

$$\text{argmin}_{P\in P(n)}\|A_1 - PA_2P^T\|_F = \text{argmin}_{P\in P(n)} -2tr(A_1^TPA_2P^T) = \text{argmax}_{P\in P(n)} 2tr(A_1^TPA_2P^T). \tag{6.4}$$

If the graph is directed, weighted and loppy, then the graph matching problem is NP-hard. Hence, currently there are no efficient algorithms known for exact graph matching.

The seeded graph matching (SGM) problem is related to the graph matching problem. We assume that a partial correspondence between the vertices is known. Those vertices are called "seeds", and the word "seeded" refers to the information known about the partial alignment of the seeds. The addition of seeds improves the performance of many graph matching algorithms [39]. In practice, it is reasonable to have seed vertices. For instance, in neural connectomics, locational information of the partial brain alignment is an example of seeding.

SGM has the same objective as GM (Equation 6.2), except that we partially observe the permutation ψ. Let S_1 be the subset of vertices for which we know the correspondence across two graphs, and $S_2 := \{\psi(v) : v \in S_1\}$ contain the vertices whose corresponding vertices are in S_1. The vertices in S_1 and S_2 are called seeds, whose alignment is known. Without loss of generality, let $S_1 = S_2 = [m]$, and the alignment between S_1 and S_2 be the identity function. Thus, ψ is observed on $[m]$.

Assume that the adjacency matrices for G_1 and G_2 are A_1 and A_2 respectively.

We can split the adjacency matrices into four submatrices as follows:

$$A_1 = \begin{pmatrix} A_1^{11} & A_1^{12} \\ A_1^{21} & A_1^{22} \end{pmatrix} \qquad A_2 = \begin{pmatrix} A_2^{11} & A_2^{12} \\ A_2^{21} & A_2^{22} \end{pmatrix}, \tag{6.5}$$

where A_1^{11} and $A_2^{11} \in \mathbb{R}^{m \times m}$, A_1^{12} and $A_2^{12} \in \mathbb{R}^{m \times (n-m)}$, A_1^{21} and $A_2^{21} \in \mathbb{R}^{(n-m) \times m}$, A_1^{22} and $A_2^{22} \in \mathbb{R}^{(n-m) \times (n-m)}$. We substitute the matrix decompositions in Equation 6.5 into the objective of Equation 6.3. Denote the $m \times m$ identity matrix by $I_{m \times m}$, and let $\oplus$ denote the direct sum of matrices. Then the objective of SGM is equivalent to

$$f(P) = \underset{P \in P(n-m)}{\operatorname{argmin}} \; tr(A_1^T (I_{m \times m} \oplus P) A_2 (I_{m \times m} \oplus P^T)) \tag{6.6}$$

$$= \underset{P \in P(n-m)}{\operatorname{argmin}} \; tr\left(\begin{bmatrix} A_1^{11} & A_1^{12} \\ A_1^{21} & A_1^{22} \end{bmatrix}^T \begin{bmatrix} I_{m \times m} & 0_{m \times (n-m)} \\ 0_{(n-m) \times m} & P \end{bmatrix} \begin{bmatrix} A_2^{11} & A_2^{12} \\ A_2^{21} & A_2^{22} \end{bmatrix} \begin{bmatrix} I_{m \times m} & 0_{m \times (n-m)} \\ 0_{(n-m) \times m} & P^T \end{bmatrix} \right)$$

$$= \underset{P \in P(n-m)}{\operatorname{argmin}} \; tr\left(\begin{bmatrix} A_1^{11} & A_1^{12} \\ A_1^{21} & A_1^{22} \end{bmatrix}^T \begin{bmatrix} A_2^{11} & A_2^{12} P^T \\ P A_2^{21} & P A_2^{22} P^T \end{bmatrix} \right)$$

$$= \underset{P \in P(n-m)}{\operatorname{argmin}} \; tr((A_1^{11})^T A_2^{11}) + tr((A_1^{21})^T P A_2^{21}) + tr((A_1^{12})^T A_2^{12} P^T) + tr((A_1^{22})^T P A_2^{22} P^T)$$

$$= \underset{P \in P(n-m)}{\operatorname{argmin}} \; tr((A_1^{21})^T P A_2^{21}) + tr((A_1^{12})^T A_2^{12} P^T) + tr((A_1^{22})^T P A_2^{22} P^T).$$

Currently no efficient algorithm is known to exist for solving SGM. Hence, we seek an approximated solution to SGM.

6.2.1 Relaxation and the Frank-Wolfe Algorithm

We work with a relaxation of the objective function in Equation 6.2, and look for an approximate solution to seeded graph matching problem. We first consider a relaxation of the seeded graph matching problem. We relax $P(n-m)$ in Equation 6.7 to $D(n-m)$ the set of doubly stochastic matrices such that $P \in \mathbb{R}^{(n-m)\times(n-m)}$ with $P1_{n-m} = 1_{n-m}$, $P^T 1_{n-m} = 1_{n-m}$ and $P \geq 0_{(n-m)\times(n-m)}$ coordinatewise, where $0_{(n-m)\times(n-m)}$ is the $(n-m)\times(n-m)$ zero matrix, and $1_{(n-m)}$ is a length-$(n-m)$ vector of all ones.

Given m seeds, the relaxed objective of seeded graph matching is:

$$f(P) = \underset{P\in D(n-m)}{\operatorname{argmin}} \; tr(P^T A_1^{21} P (A_2^{21})^T) + tr(P^T (A_1^{21})^T A_2^{12}) + tr((A_2^{22})^T P A_2^{22}). \quad (6.7)$$

After this relaxation, the feasible region is a convex hull of the permutation matrices. Solutions to this relaxation are achieved using the Frank-Wolfe Algorithm [43], an iterative nonlinear optimization algorithm. The general optimization setup where the Frank-Wolfe algorithm is applicable is

$$\min_{x\in\mathcal{S}} f(x), \quad (6.8)$$

where $\mathcal{S}$ is a bounded and convex domain, and $f : \mathcal{S} \to \mathbb{R}$ is continuously differentiable. The main idea of this algorithm is to solve local linearizations of the objective function iteratively by using the solution from the previous step as the location of the linearization in the current step. The Frank-Wolfe Algorithm is presented in Algorithm 6.

Algorithm 6 Frank-Wolfe Algorithm

$t \leftarrow 1$, $\alpha = 1$ ▷ Initialization

Randomly select $x_0 \in \mathcal{S}$ or an initial estimate of x^*.

while x_t has not converged **do**

$s_t \leftarrow \arg\min_{s \in \mathcal{S}} s^T \bigtriangledown f(x_t)$ ▷ Linearization: 1st Order Approximation

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

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

$t \leftarrow t + 1$

end while

Output: $x^* = x_{t+1}$.

Remark 6.1. When the function $f(x)$ is quadratic, α can be found analytically. The steps of Direction-Finding and Line Search continue to repeat until termination conditions are met.

6.2.2 Solving the Approximated Seeded Graph Matching Problem

While there is no hope in solving the exact SGM problem efficiently, one seeks an approximate efficient solution for SGM. Fishkind et al. [39] modified the Fast Approximate Quadratic Assignment (FAQ) algorithm [94] to approximately solve the relaxed seeded graph matching algorithm. The seeded FAQ algorithm is the state-of-

the-art seeded graph matching algorithm, and it employs the Frank-Wolfe algorithm to solve the relaxed SGM problem. Let us first present the derivation of the steps of the seeded FAQ algorithm.

Recall the objective for SGM in Equation 6.7, which has gradient $\nabla_P(f)$,

$$\nabla_P(f) := A_1^{21}(A_2^{21})^T + (A_1^{12})^T A_2^{12} + A_1^{22} P (A_2^{22})^T + (A_1^{22})^T P A_2^{22}. \tag{6.9}$$

We initialize the Frank-Wolfe algorithm at the barycenter $P_0 = \frac{1}{n-m} 1_{n-m} 1_{n-m}^T$. We choose this initialization for simplicity, but other doubly stochastic matrices are suitable. For instance, one can also use the reversed Cuthill-McKee ordering as an initialization [58] or a convex initialization [61]. Recall in the Frank-Wolfe linearization step, we maximize $tr(Q^T \nabla P_0)$ over all doubly stochastic matrices $Q \in \mathbb{R}^{(n-m)\times(n-m)}$. The function $tr(Q^T \nabla P_0)$ is linear in Q, and we can use the Hungarian Algorithm to find the optimal Q, which will be denoted by $\tilde{Q}$. The complexity for this step is $O(n^3)$. The next step of applying the Frank-Wolfe algorithm is to maximize the one-dimensional objective function $f(\alpha \tilde{P} + (1-\alpha)\tilde{Q})$, where $\alpha \in [0,1]$, over all line segments from P_t to $\tilde{Q}$. Define the following constants:

$$c := tr((A_1^{22})^T \tilde{P} A_2^{22} \tilde{P}^T), \tag{6.10}$$

$$d := tr((A_1^{22})^T \tilde{P} A_2^{22} \tilde{Q}^T + (A_1^{22})^T \tilde{Q} A_2^{22} \tilde{P}^T), \tag{6.11}$$

$$e := tr((A_1^{22})^T \tilde{Q} A_2^{22} \tilde{Q}^T), \tag{6.12}$$

$$u := tr(\tilde{P}^T A_1^{21} (A_2^{21})^T + \tilde{P}^T (A_1^{12})^T A_2^{12}), \tag{6.13}$$

$$v := tr(\tilde{Q}^T A_1^{21} (A_2^{21})^T + \tilde{Q}^T (A_1^{12})^T A_2^{12}). \tag{6.14}$$

Based the definitions of c, d, e, u, v, the step of line search is simplified as

$$f(\alpha\tilde{P} + (1-\alpha)\tilde{Q}) = (c - d + e)\alpha^2 + (d - 2e + u - v)\alpha + (e + v). \tag{6.15}$$

We differentiate Equation 6.15 with respect to α to get the critical point $\tilde{\alpha} = \frac{-d+2e-u+v}{2(c-d+e)}$. Because of the $[0, 1]$-constraint of α, we set $\tilde{\alpha} := \min(1, \frac{-d+2e-u+v}{2(c-d+e)})$. If $\alpha > 1-\epsilon$, for some tolerance ϵ, then the seeded FAQ algorithm terminates. Otherwise, we repeat the procedure.

6.2.3 Projection

The Frank-Wolfe algorithm converges to a local optimum $\tilde{P}$, which may not be a permutation matrix, and thus a projection step is needed. Denote $P(n-m)$ as the space of all permutation matrices of order $n-m$. In this case, we seek a permutation matrix $\tilde{Q}$ such that

$$\tilde{Q} = \underset{Q \in P(n-m)}{\operatorname{argmin}} \|Q - \tilde{P}\|_1. \tag{6.16}$$

When Q is a permutation matrix, Equation 6.16 can be simplified to $2n - 2tr(Q^T\tilde{P})$. Hence, we can again minimize $tr(Q^T\tilde{P})$ using the Hungarian Algorithm with complexity $O(n^3)$. Hence, for a bounded number of iterations in the Frank-Wolfe algorithm, we have efficiently approximated the solution to the SGM problem.

6.2.4 Performance Evaluation Criterion

Assume the total number of vertices is n. The number of seeds is m. The performance of seeded graph matching solution is measured by the matching accuracy $\delta(m)$, defined as the number of correctly matched non-seeded vertices divided by the total number of non-seeds $n - m$. The matching accuracy for chance is $\frac{1}{n-m}$. More information regarding the partial correspondence between vertices is available, and we expect the matching accuracy to increase [39]. Examples of an evaluation of the seeded graph matching performance is shown in Figure 7.6.

6.3 Large Seeded Graph Matching

With advanced technologies, data sets are collected in enormous volumes and complex structures. The emerge of big data poses high demands for scalable algorithms. In this section, we propose a bijective scalable version of the state-of-the-art seeded graph matching algorithm for big graphs. We explore the possibility of making the seeded graph matching algorithm scalable for big graphs, present our proposed Large Seeded Graph Algorithm (LSGM), verify its theoretical guarantees, and demonstrate its effectiveness in simulation and real data experiments. Our real data example indicates the applicability of LSGM for matching scale-free large graphs.

6.3.1 Motivation

Growing volume and complexity of big graph data are posing significant challenges in graph inference. Among the developed approximated GM algorithms, the current state-of-the-art nonseeded matching algorithms are PATH [100], GLAG [37], and FAQ [94]. However they are not applicable for large graphs due to their $O(n^3)$ complexity. The current existing scalable bijective graph matching algorithm, the Umeyama's spectral approach (U) [92], has faster run time, but its performance degrades greatly compared to seeded FAQ. Furthermore, PATH, GLAG, FAQ and U do not leverage the information contained in a partial correspondence between vertices by incorporating seeds information. Our goal is to propose a bijective graph matching algorithm, which not only has lower computational cost than the state-of-the-art GM algorithms, but also leverages the information from seeds to enhance the matching performance.

6.3.2 The Large Seeded Graph Matching Algorithm

Our approach to make the SGM algorithm scalable is divide-and-conquer. The main three procedures in the LSGM algorithm are: joint spectral embedding, vertex clustering, and seeded graph matching within the clusters. In the last step, we achieve parallelization when SGM is done within the clusters. We present our proposed Large Seeded Graph Matching (LSGM) algorithm in Algorithm 7, and explain the steps in

detail. A depiction of the LSGM algorithm is seen in Figure 6.1.

Algorithm 7 Large Seeded Graph Matching Algorithm (LSGM)

Input: Adjacency matrices $A_1, A_2 \in \{0,1\}^{n \times n}$, number of seeds $m \in \mathbb{N}$, seed bijection $\phi : [m] \to [m]$.

Output: Matching ψ of the vertices of G_1 and G_2.

Step 1: Joint spectral embedding.

Step 2: (Divide) Joint clustering on the embedding.

Step 3: (Conquer) For each cluster, in parallel:

for $i = 1$ to K **do**

Match within cluster i across the graphs using state-of-the-art SGM algorithm in Section 6.2, yielding matching $\psi^{(i)}$.

end for

Output: Matching ψ on the entire two graphs is the direct sum of the matching within each cluster: $\psi = \oplus_{i=1}^{K} \psi^{(i)}$.

6.3.2.1 Joint Embedding and Clustering the Graphs

Step 2 and Step 3 in Algorithm 7 are described in detail in Algorithm 8.

Note that the Step 1 and Step 2 in Algorithm 8 are exactly adjacency spectral embedding (ASE) introduced in Algorithm 1 applied on each adjacency matrix respectively. ASE can be computed in $O(n^2 d)$ steps for $d \leq \sqrt{n}$ [11].

Since the embedded vertices of G_1 and G_2 are most likely not in two spaces which

Algorithm 8 Jointly embedding and clustering on G_1 and G_2

Input: Adjacency matrices $A_1, A_2 \in \{0,1\}^{n\times n}$, number of seeds $m \in \mathbb{N}$, seed bijection $\phi_S : [m] \to [m]$, the embedding dimension $d \in \mathbb{N}$, number of clusters $K \in \mathbb{N}$.

Output: Obtain K clusters of the $2n$ jointly embedded vertices.

Step 1: Compute the first d orthonormal eigenpairs of A_1 and A_2, namely (U_{A_1}, S_{A_1}) and (U_{A_2}, S_{A_2}) respectively.

Step 2: Estimate the latent positions via adjacency spectral embedding $\hat{\mathcal{X}}_{A_1} := U_{A_1} S_{A_1}^{1/2}$, $\hat{\mathcal{X}}_{A_2} := U_{A_2} S_{A_2}^{1/2}$.

Step 3: Align the embedded seeds via the orthogonal Procrustes fit problem $\hat{\mathcal{X}}_{A_1,m} := \hat{\mathcal{X}}_{A_1}([m],:)$, $\hat{\mathcal{X}}_{A_2,m} := \hat{\mathcal{X}}_{A_2}([m],:)$, $Q := \text{argmin}_{W\in W(d)} \|\hat{\mathcal{X}}_{A_1,m} W - \hat{\mathcal{X}}_{A_2,m}\|_F$, where $W(d) := \{W \in \mathbb{R}^{d\times d} : W^T W = I\}$.

Step 4: Align the two embedded adjacency matrices via applying the Procrustes transformation Q to $\hat{\mathcal{X}}_{A_1}$ obtaining the aligned embedding $\hat{\mathcal{X}}_{A_1} Q$ of A_1;

Step 5: Cluster the $2n$ embedded points $\begin{pmatrix} \hat{\mathcal{X}}_{A_1} Q \\ \hat{\mathcal{X}}_{A_2} \end{pmatrix} \in \mathbb{R}^{2n\times d}$ into K clusters via the K-means clustering procedure.

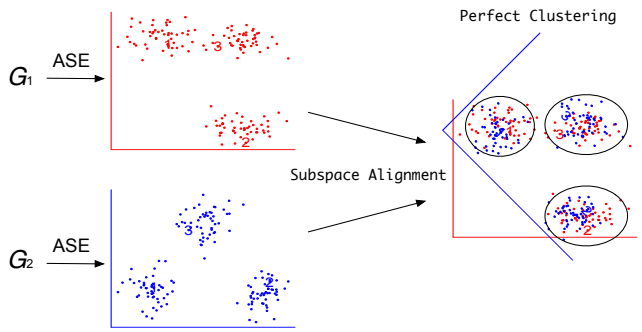


Figure 6.1: A depiction of the large seeded graph matching algorithm. The algorithm consists of four steps: adjacency spectral embedding, subspace alignment, clustering and seeded graph matching.

do not have the same orientation, we intend to align the $2n$ embedded vertices via the orthogonal Procrustes fit problem. Here we utilize the seeds whose alignment across the two graphs is known to find the Procrustes transformation, as seen in Step 3. In Step 4, we align the two sets of embedded vertices via the Procrustes transformation Q obtained via aligning the seeds. Hence the transformed and aligned embedding is $\hat{\mathcal{X}}_{A_1}Q$. In Step 5, we apply K-means clustering algorithm on the $2n$ vertices in the d-dimensional concatenated data frame $\begin{pmatrix} \hat{\mathcal{X}}_{A_1}Q \\ \hat{\mathcal{X}}_{A_2} \end{pmatrix} \in \mathbb{R}^{2n \times d}$.

6.3.2.2 The Low Computational Cost of Joint Embedding and Joint Clustering

Our proposed LSGM algorithm mainly focuses on the parallelization of the matching step, because the other two main steps – joint embedding via ASE, and joint

clustering via K-means are not computationally expensive. For large n, we are only interested in the first $d << n$ eigenpairs of A_1 and A_2. The step of ASE can be achieved using efficient singular value decomposition (SVD) algorithms. The SVD step can be calculated in $O(n^2 d)$ steps for $d << \sqrt{n}$ [7, 11]. The K-means algorithm has complexity $O(Kdn)$ for each iteration. Note that we do not implement parallelized versions of the SVD algorithms or the K-means clustering algorithms. The approximate SVD algorithm and the approximate clustering algorithm used in the LSGM algorithm are efficiently implementable. As we will see in Section 6.4, the matching step is the most time intensive step, while the non-parallized SVD and K-means take sufficiently small portion of the runtime.

6.3.2.3 Model Selection on Embedding Dimension

The embedding dimension d is often unknown in practice. In the simulation experiment in Section 6.4, we assume that d is given. In the real data experiment in Section 6.4, we estimate d using the partial scree plot of the two graphs, suggested in [14]. Recall in Chapter 3 and 4, we use an automatic profile likelihood procedure [102] to estimate d. This procedure, however, is not suitable for big graphs, as it requires knowing the full spectrum.

6.3.2.4 Model Selection on Number of Clusters

The number of clusters K is often unknown in practice, and usually dictated by the data. In the K-means clustering procedure, a specified K is required. In other clustering algorithms such as Mclust [42], the number of clusters K is selected using the Bayesian information criterion (BIC). In the simulation experiments in Section 6.4, we assume K is given. In the human brain connectome experiment, we set an upper bound on the maximum number of vertices in a cluster. We want to point out that our LSGM algorithm is insensitive to misspecification of K as in the simulation experiment illustrating its robustness to misspecified K.

6.3.2.5 Ensure Cluster Sizes Suitable for Bijective Seeded Graph Matching

After the joint embedding, we utilize seeded graph matching within the clusters. Our seeded graph matching is under the bijective framework so that the cluster sizes need to be consistent across two graphs. That is, for each pair of clusters across the two graphs, the number of vertices from G_1 must be equal to the number of vertices from G_2. However, the K-means algorithm cannot ensure this.

Suppose that for each $i = 1, 2, \ldots, K$, cluster i has c_i total vertices (from both graphs combined) with $c_1 \geq c_2 \geq \cdots \geq c_K$. Within cluster i, suppose there are $c_i^{(1)}$ vertices from G_1 and $c_i^{(2)}$ vertices from G_2. The ideal cluster size for cluster i would be

$2\left\lceil \frac{c_i^{(1)}+c_i^{(2)}}{2} \right\rceil$ with $\left\lceil \frac{c_i^{(1)}+c_i^{(2)}}{2} \right\rceil$ vertices from G_1 and G_2 respectively. However, it could happen that $\sum_i \left\lceil \frac{c_i^{(1)}+c_i^{(2)}}{2} \right\rceil \geq n$, though it always holds that $\sum_i \left\lceil \frac{c_i^{(1)}+c_i^{(2)}}{2} \right\rceil \leq n+2K$. Let $\{\tilde{c}_i\}_{i=1}^K$ denote the resized cluster size. In order to solve this issue, we first set the cluster i to be of size $2\tilde{c}_i = 2\left\lceil \frac{c_i^{(1)}+c_i^{(2)}}{2} \right\rceil$. Then starting from the smallest cluster, we remove 2 vertices: 1 vertex from each graph until $\sum_{i=1}^K \tilde{c}_i = n$. This procedure is mathematically formulated in Equation 6.17.

$$\widetilde{c}_i = 2\left\lceil \frac{c_i^{(1)} + c_i^{(2)}}{2} \right\rceil - 2 \cdot \mathbb{1}\left\{ \sum_{j=1}^{K} \left\lceil \frac{c_j^{(1)} + c_j^{(2)}}{2} \right\rceil \geq i + n \right\}. \tag{6.17}$$

6.3.2.6 Assign Vertices to Clusters

Once the issue of cluster sizes is resolved, we re-assign the vertices in the clusters. For each cluster i with centroid C_i, $\frac{\tilde{c}_i}{2}$ vertices which are closest to C_i are selected from each graph. Subsequently, we can match the two graphs within a cluster using bijective matching algorithms.

We view our resizing procedure described above as a refinement of the original K-means procedure, and not as providing a new clustering of the vertices. Empirically we see that our reassigned clusters are very similar to the original K-means clusters, often differing in only a few vertices. Other clustering algorithms such as Mclust is also applicable for the clustering task.

6.3.2.7 LSGM Modifications

Empirically, imputing the diagonal of the adjacency matrices can improve inference performance. We can follow the suggestions of [80] and [64] to perform diagonal augmentation via imputing the diagonal entries $a_{ii} = \frac{\deg(i)}{n-1}$, where $\deg(i)$ denotes the degree of vertex i. In addition, projecting the embedded vertices $\hat{X}$ onto the sphere has also shown to empirically improve clustering performance. Furthermore, within each cluster, seeded graph matching algorithm with different initializations such as barycenter or convex initialization may also influence the matching performance. Of course, whether these adjustments on the LSGM algorithm will improve matching performance heavily depends on the empirical graph structures.

6.3.3 Complexity of LSGM

We first consider the computational cost of each step in the LSGM algorithm. As noted before, for very large graphs, $d << n$; the computational step for eigen-decomposition on the adjacency matrix is $O(n^2 d)$ when $d \leq \sqrt{n}$. The orthogonal Procrustes fit problem has complexity at most $O(nd^2)$. As mentioned in Section 6.2, the state-of-the-art SGM algorithm has complexity $O(n^3)$ on the entire graph. In LSGM, note that SGM is not executed on the entire two graphs with size n. Rather, SGM proceeds within each cluster i of size $\tilde{c}_i$. If SGM is executed in parallel, then the complexity of the matching step is $O((\tilde{c}_{\max} + m)^3)$, where $\tilde{c}_{\max}$ denotes the maximum

cluster size after resizing (as seen in Section 6.3.2.5). If SGM is executed in sequence, then the complexity of the matching step is $O(K(\tilde{c}_{\max} + m)^3)$.

Assume there exists an $\alpha > 0$ such that $m = o(n^{1-\alpha})$, $K = \Omega(n^{\alpha})$ and each cluster size is bounded by $\tilde{c}_{\max} = O(n^{1-\alpha})$. If $\alpha \leq \frac{1}{3}$, the computational cost of LSGM is $O(n^2 d)$, when the matching step is fully parallelized. If $\alpha > \frac{1}{3}$, the computational cost of LSGM is $O(n^{3(1-\alpha)})$, when the matching step is fully parallelized. When $\alpha \approx \frac{1}{3}$, which denotes a modest number of modestly sized clusters, the computational cost of LSGM is $O(n^2 d)$.

6.3.4 Theoretical Guarantee Under the Stochastic Blockmodel Framework

In this section, we prove a theoretical guarantee for LSGM under the stochastic blockmodel framework, where we assume that G_1 and G_2 are realized from stochastic blockmodels with parametrization $\text{SBM}(\vec{n}, b, B)$. Recall the definition of correlated stochastic blockmodels in Definition 2.8. If G_1 and G_2 are ρ-correlated, then there is a natural latent alignment between the vertices of the two graphs – the identity function id_n. Again, as in Chapter 5, let $\vec{m} \in \mathbb{N}^K$ denote the vector whose entry is the number of seeds (non-ambiguous vertices) in each block. Then the m seeds have known correspondence across the two ρ-correlated graphs G_1 and G_2.

Let A_1 and A_2 be the adjacency matrices of G_1 and G_2 respectively. Let b_{A_1} and

b_{A_2} be the block assignment function of G_1 and G_2 respectively. Without loss of generality, let the true alignment function be id_n and let $b := b_{A_1} = b_{A_2}$. We define the clustering criterion for clustering the rows of $[\hat{\mathcal{X}}_{A_2}^T | (\hat{\mathcal{X}}_{A_1} Q)^T]^T$ into K clusters via

$$(\widehat{C}, \hat{b}) := \mathrm{argmin}_{C \in \mathbb{R}^{K \times d},\ b:[2n] \to [K]} \sum_{i=1}^{2n} \left\| \left[\begin{pmatrix} \hat{\mathcal{X}}_{A_2} \\ \hat{\mathcal{X}}_{A_1} Q \end{pmatrix} \right] (i,:) - C(b(i),:) \right\|_2^2, \tag{6.18}$$

In the next theorem, we prove that under mild assumptions, for all but finitely many n, the estimated memberships $\hat{b} = b$, and all of the vertices are perfectly clustered across the two graphs, and thus it implies that the joint clustering procedure yields a canonical matching of the vertices given the clustering. Note that the asymptotic result on the graph means that we consider a sequence of growing random graph models $G_1, G_2, \ldots, G_n$ with $n = 1, 2, \ldots$ vertices. Although this result is asymptotic in nature, it guarantees that LSGM will be effective in approximating the the true but unknown alignment for SBMs.

Theorem 6.1. Let $\lambda_i(M)$ denote the i-th largest eigenvalue of matrix M. If the following assumptions hold:

i. There exist constants ϵ_1, $\epsilon_2 > 0$ such that $K = O(n^{1/3-\epsilon_1})$ and $\min_i \vec{n}(i) = \Omega(n^{2/3+\epsilon_2})$;

ii. Define the eigen-gap

$$\delta_d := \min_{i,j \le d+1, i \ne j} |\lambda_i(\mathcal{X}\mathcal{X}^T) - \lambda_j(\mathcal{X}\mathcal{X}^T)|/n, \tag{6.19}$$

and

$$\beta := \beta(n, d, \delta_d) = \frac{260d\log(n)}{\delta_d n^{1/2}}, \tag{6.20}$$

if i, $j \in [n]$ are such that $\mathcal{X}(i,:) \neq \mathcal{X}(j,:)$ then $\|\mathcal{X}(i,:) - \mathcal{X}(j,:)\|_2 > 6n^{1/6}\beta$;

iii. Let $\{\mathcal{X}(i,:)\}_{i=1}^m$ be the latent positions corresponding to the seeded vertices, then we assume there exists an α satisfying $\alpha > 4\beta$ and $\sqrt{n}\beta/\alpha = o(n^{\epsilon_2/2}d/\delta_d)$ such that

$$\min_{v\,:\,\|v\|_2=1} \|\mathcal{X}([m],:)v^T\|_2 \geq \alpha\sqrt{m}; \tag{6.21}$$

then for all but finitely many n, the $\hat{b}$ satisfies $\hat{b} = b$, and LSGM finds the true alignment almost always.

Recall for a matrix M, the norm $\|M\|_{2\to\infty} := \max_i \|M(i,:)\|_2$. We first cite a lemma proved in [84] and [63].

Lemma 6.1. Let $\widehat{\mathcal{X}}_{A_1}$ and $\widehat{\mathcal{X}}_{A_2}$ denote the resulted adjacency spectral embedding on A_1 and A_2. Let $W_{A_1} = \text{argmin}_{W\in W(d)} \|\widehat{\mathcal{X}}_{A_1} - \mathcal{X}_{A_1}W\|_F$ and $W_{A_2} = \text{argmin}_{W\in W(d)} \|\widehat{\mathcal{X}}_{A_2} - \mathcal{X}_{A_1}W\|_F$. If $d = o(\sqrt{n})$, then it holds with probability one that for all but finitely many n that

$$\|\widehat{\mathcal{X}}_{A_1} - \mathcal{X}_{A_1}W_{A_1}\|_{2\to\infty} \leq \beta \text{ and } \|\widehat{\mathcal{X}}_{A_1} - \mathcal{X}_{A_1}W_{A_2}\|_{2\to\infty} \leq \beta. \tag{6.22}$$

Now we prove the following.

Lemma 6.2. Let $Q := \text{argmin}_{W\in W(d)} \|\widehat{\mathcal{X}}_{A_1}([m],:)W - \widehat{\mathcal{X}}_{A_2}([m],:)\|_F$. For all but finitely many n it holds that $\|\widehat{\mathcal{X}}_{A_1}Q - \widehat{\mathcal{X}}_{A_2}\|_{2\to\infty} \leq 8\beta/\alpha + 2\beta$.

Proof. Define $\tilde{Q} := W_{A_1}^\top W_{A_2}$. It follows from Equation 6.22 that

$$\|\widehat{\mathcal{X}}_{A_1}\tilde{Q} - \widehat{\mathcal{X}}_{A_2}\|_{2\to\infty} \leq \|\widehat{\mathcal{X}}_{A_1} - \mathcal{X}_{A_1}W_{A_1}\|_{2\to\infty} + \|\widehat{\mathcal{X}}_{A_1} - \mathcal{X}_{A_1}W_{A_2}\|_{2\to\infty} \leq 2\beta. \quad (6.23)$$

It is also easy to see that

$$\|\widehat{\mathcal{X}}_{A_1}([m],:)Q - \hat{\mathcal{X}}_{A_2}([m],:)\|_F \leq \|\widehat{\mathcal{X}}_{A_1}([m],:)\tilde{Q} - \widehat{\mathcal{X}}_{A_2}([m],:)\|_F \leq 2\beta\sqrt{m}. \quad (6.24)$$

Going from the other direction, we have,

$$\begin{aligned}\|\widehat{\mathcal{X}}_{A_1}([m],:)Q - \widehat{\mathcal{X}}_{A_2}([m],:)\|_F &\geq \|\widehat{\mathcal{X}}_{A_1}([m],:)Q - \widehat{\mathcal{X}}_{A_1}([m],:)\tilde{Q} + \widehat{\mathcal{X}}_{A_1}([m],:)\tilde{Q} - \widehat{\mathcal{X}}_{A_2}([m],:)\|_F \\ &\geq \|\widehat{\mathcal{X}}_{A_1}([m],:)(Q-\tilde{Q})\|_F - \|\widehat{\mathcal{X}}_{A_1}([m],:)\tilde{Q} - \widehat{\mathcal{X}}_{A_2}([m],:)\|_F \\ &\geq \|\widehat{\mathcal{X}}_{A_1}([m],:)(Q-\tilde{Q})\|_F - 2\beta\sqrt{m}. \quad (6.25)\end{aligned}$$

Hence,

$$\|\widehat{\mathcal{X}}_{A_1}([m],:)Q - \widehat{\mathcal{X}}_{A_2}([m],:)\|_F \leq 4\beta\sqrt{m}. \quad (6.26)$$

Define the singular value decomposition on $Q - \tilde{Q} := V_1 S V_2^\top$.

$$\begin{aligned}\|\widehat{\mathcal{X}}_{A_1}([m],:)(Q-\tilde{Q})\|_F &= \|\mathcal{X}_{A_1}([m],:)W_{A_1}(Q-\tilde{Q}) - \mathcal{X}_{A_1}([m],:)W_{A_1}(Q-\tilde{Q}) \\ &\quad + \widehat{\mathcal{X}}_{A_1}([m],:)(Q-\tilde{Q})\|_F \\ &\geq \|\mathcal{X}_{A_1}([m],:)W_{A_1}(Q-\tilde{Q})\|_F - \|\left(\widehat{\mathcal{X}}_{A_1}([m],:) - X([m],:)W_{A_1}\right)(Q-\tilde{Q})\|_F \\ &\geq \left(\sum_{i=1}^{m}\sum_{j=1}^{d}\langle\mathcal{X}_{A_1}(i,:), W_{A_1}V_1(:,j)\rangle S(j,j)^2\right)^{1/2} - 2\beta\sqrt{m}\|Q-\tilde{Q}\|_F \\ &\geq (\alpha - 2\beta)\sqrt{m}\|Q-\tilde{Q}\|_F \quad (6.27)\end{aligned}$$

Recall that Equation 6.21) states $\min_{\|v\|_2=1} \|X([m],:)v\|_2^2 \geq \alpha^2 m$ and by Equation 6.26, we have,

$$(\alpha - 2\beta)\sqrt{m}\|Q - \tilde{Q}\|_F \leq \|\widehat{\mathcal{X}}_{A_1}([m],:)(Q - \tilde{Q})\|_F \leq 4\beta\sqrt{m}. \tag{6.28}$$

Cancelling $\sqrt{m}$ on both sides, and divide by $(\alpha - 2\beta)$,

$$\|Q - \tilde{Q}\|_F \leq \frac{4\beta}{\alpha - 2\beta}. \tag{6.29}$$

Since $\|\widehat{\mathcal{X}}_{A_1}\|_{2\to\infty} \leq 1$ and $\alpha > 4\beta$ as stated in Assumption *iii* in Theorem 6.1, we have

$$\begin{aligned} \|\widehat{\mathcal{X}}_{A_1}Q - \widehat{\mathcal{X}}_{A_2}\|_{2\to\infty} &= \|\widehat{\mathcal{X}}_{A_1}(Q - \tilde{Q}) + \mathcal{X}_{A_1}\tilde{Q} - \widehat{\mathcal{X}}_{A_2}\|_{2\to\infty} \\ &\leq \|\widehat{\mathcal{X}}_{A_1}(Q - \tilde{Q})\|_{2\to\infty} + \|\widehat{\mathcal{X}}_{A_1}\tilde{Q} - \widehat{\mathcal{X}}_{A_2}\|_{2\to\infty} \\ &\leq \|\widehat{\mathcal{X}}_{A_1}\|_{2\to\infty}4\beta/(\alpha - 2\beta) + 2\beta \leq 8\beta/\alpha + 2\beta \quad \leq \frac{8\beta}{\alpha} + 2\beta. \end{aligned} \tag{6.30}$$

□

Lemma 6.3. For all but finitely many n, it holds that

$$\left\| \begin{pmatrix} \widehat{\mathcal{X}}_{A_2} \\ \widehat{\mathcal{X}}_{A_1}Q \end{pmatrix} - \begin{pmatrix} \mathcal{X}_{A_1}W_{A_2} \\ \mathcal{X}_{A_1}W_{A_2} \end{pmatrix} \right\|_{2\to\infty} \leq \frac{8\beta}{\alpha} + 3\beta.$$

Proof. Note that

$$\left\| \begin{pmatrix} \widehat{\mathcal{X}}_{A_2} \\ \widehat{\mathcal{X}}_{A_1}Q \end{pmatrix} - \begin{pmatrix} \mathcal{X}_{A_1}W_{A_2} \\ \mathcal{X}_{A_1}W_{A_2} \end{pmatrix} \right\|_{2\to\infty} = \max\{\|\hat{\mathcal{X}}_{A_2} - \mathcal{X}_{A_1}W_{A_2}\|_{2\to\infty}, \|\hat{\mathcal{X}}_{A_1}Q - \mathcal{X}_{A_1}W_{A_2}\|_{2\to\infty}\}. \tag{6.31}$$

Note that $\|\hat{\mathcal{X}}_{A_2} - \mathcal{X}_{A_1}W_{A_2}\|_{2\to\infty} \leq \beta$ as shown in Lemma 6.1. By Lemma 6.2, we have

$$\|\hat{\mathcal{X}}_{A_1}Q - \mathcal{X}_{A_1}W_{A_2}\|_{2\to\infty} = \|\hat{\mathcal{X}}_{A_1}Q - \hat{\mathcal{X}}_{A_2} + \hat{\mathcal{X}}_{A_2} - \mathcal{X}_{A_1}W_{A_2}\|_{2\to\infty} \tag{6.32}$$

$$\leq \|\hat{\mathcal{X}}_{A_1}Q - \hat{\mathcal{X}}_{A_2}\|_{2\to\infty} + \|\hat{\mathcal{X}}_{A_2} - \mathcal{X}_{A_1}W_{A_2}\|_{2\to\infty} \tag{6.33}$$

$$\leq \frac{8\beta}{\alpha} + 3\beta. \tag{6.34}$$

Hence,

$$\left\| \begin{pmatrix} \widehat{\mathcal{X}}_{A_2} \\ \widehat{\mathcal{X}}_{A_1}Q \end{pmatrix} - \begin{pmatrix} \mathcal{X}_{A_1}W_{A_2} \\ \mathcal{X}_{A_1}W_{A_2} \end{pmatrix} \right\|_{2\to\infty} \leq \frac{8\beta}{\alpha} + 3\beta. \tag{6.35}$$

□

With all the necessary ingredients ready, we proceed to prove Theorem 6.1 – the theoretical performance guarantee of LSGM.

Proof. Let $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_K$ be the L^2-balls of radius $r := n^{1/6}\beta$ around the K distinct rows of $\mathcal{X}_{A_1}W_{A_2}$. If $\mathcal{X}_{A_1}(i,:) \neq \mathcal{X}_{A_1}(j,:)$, then by Assumption *ii*

$$6r = 6n^{1/6}\beta \leq \|\mathcal{X}_{A_1}(i,:) - \mathcal{X}_{A_1}(j,:)\|_2 = \|(\mathcal{X}_{A_1}(i,:) - \mathcal{X}_{A_1}(j,:))W_{A_2}\|_2, \tag{6.36}$$

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

Define $\widehat{Z} := \begin{pmatrix} \widehat{\mathcal{X}}_{A_2} \\ \widehat{\mathcal{X}}_{A_1}Q \end{pmatrix} \in \mathbb{R}^{2n\times d}$, and define $Z := \begin{pmatrix} \mathcal{X}_{A_1}W_{A_2} \\ \mathcal{X}_{A_1}W_{A_2} \end{pmatrix} \in \mathbb{R}^{2n\times d}$. Let $(\widehat{\mathcal{C}}, \hat{b})$ be the optimal clustering of the rows of $\widehat{Z}$ from the clustering criterion in Equation 6.18. We first prove by contradiction. Suppose there is an index $i \in [2n]$ such that $\|Z(i,:) - \widehat{C}(i)\| > 2r$. Then by Lemma 6.1, we have

$$\|\widehat{\mathcal{X}}_{A_1} - \mathcal{X}_{A_1}W_{A_1}\|_{2\to\infty} \leq \beta \text{ and } \|\widehat{\mathcal{X}}_{A_1} - \mathcal{X}_{A_1}W_{A_2}\|_{2\to\infty} \leq \beta. \tag{6.37}$$

. Then using the triangle inequality, we have

$$\|\widehat{Z}(i,:) - \widehat{C}(i)\| \geq \|Z(i,:) - \widehat{C}(i)\| - \|\widehat{Z}(i,:) - Z(i,:)\| > 2r - \beta. \tag{6.38}$$

Thus, we must have

$$|\widehat{Z} - \widehat{C}\|_F > \sqrt{\min_k n_k}(2r - \beta). \tag{6.39}$$

According to Assumption i, $\sqrt{\min_k n_k} = \Omega(n^{2/3+\epsilon_2})$, and plug in back $r = n^{1/6}\beta$, we have

$$\|\widehat{Z} - \widehat{C}\|_F > \sqrt{\min_k n_k}(2r - \beta) \tag{6.40}$$

$$= \Omega(\sqrt{n^{2/3+\epsilon_2}})(2n^{1/6}\beta - \beta) \tag{6.41}$$

$$= \Omega\left(\frac{n^{\epsilon_2/2}d}{\delta_d}\right). \tag{6.42}$$

By Lemma 6.3, we have

$$\|\widehat{Z} - Z\|_F \leq \sqrt{2n}\left(\frac{8\beta}{\alpha} + 3\beta\right) = o\left(\frac{n^{\epsilon_2/2}d}{\delta_d}\right). \tag{6.43}$$

This contradicts the clustering criterion in Equation 6.18. Hence, it must holds $\|Z(i,:) - \widehat{C}(i)\| \leq 2r$ for all $i \in [2n]$.

Again by triangle inequality and Lemma 6.1, we have

$$\|Z - \hat{C}\|_{2\to\infty} = \|Z - \widehat{Z} + \widehat{Z} - \hat{X}\|_{2\to\infty} \tag{6.44}$$

$$\leq \|Z - \widehat{Z}\|_{2\to\infty} + \|\widehat{Z} - \hat{X}\|_{2\to\infty} \tag{6.45}$$

$$\leq 2r + \beta = (2 + o(1))r. \tag{6.46}$$

Assume $i, j \in [n]$ such that $\widehat{C}(i,:) \neq \widehat{C}(j,:)$. Then according to Assumption *ii*, we have $\|Z(i,:) - Z(j,:)\|_2 > 6r$. Then, by triangle inequality and Equation 6.46, it follows that

$$\|\widehat{Z}(i,:) - \widehat{C}(j,:)\|_2 = \|\widehat{Z}(i,:) - Z(i,:) + Z(i,:) - \widehat{C}(j,:)\|_2 \tag{6.47}$$

$$\geq \|\widehat{Z}(i,:) - Z(i,:)\|_2 - \|Z(i,:) - \widehat{C}(j,:)\|_2 \tag{6.48}$$

$$\geq 6r - \beta - 2r \tag{6.49}$$

$$= 4r - \beta = (4 + o(1))r. \tag{6.50}$$

It follows that for a sequence of SBM$(\vec{n}, b, B)$, for all but finitely many n, the clustering assignment $\hat{b} = \begin{pmatrix} b \\ & \\ b \end{pmatrix}^T \in \mathbb{N}^{2n}$. That is, the number of misclustered vertices is 0. Then by Theorem 1 in [62], we have that for all but finitely many n, $\psi^{(i)} = \{I_{u_i}\}$ for all $i \in [K]$. Hence, LSGM produces the perfect matching under the stochastic blockmodel assumption. □

6.4 Experiments

In this section, we apply Seeded FAQ and Large Seeded Graph Matching on simulation and real datasets. When comparing across graph matching algorithms, we measure effectiveness via the matching accuracy and runtime of the algorithms. Across both runtime and accuracy, our algorithm achieves excellent performance: achieving significantly better accuracy than existing state-of-the-art scalable bijective

matching algorithms (Umeyama's spectral approach [92]), and achieving significantly better accuracy and runtime than the existing state-of-the-art (in terms of accuracy) matching procedures (PATH [100], GLAG [37], FAQ [94]).

6.4.1 Comparison of Existing Graph Matching Algorithms

This experiment compares the performance of available bijective matching algorithms in terms of accuracy and run time. We consider two ρ-correlated SBM$(\vec{n}, b, B)$ random graphs with the following parameters: each of $\rho = 0.6$ and $\rho = 0.9$, $B = \begin{pmatrix} 0.6 & 0.3 \\ 0.3 & 0.6 \end{pmatrix} \in [0,1]^{2\times 2}$, $\vec{n} = [n/2, n/2]$, for each of $n = 100, 200, 300, 400$. We cluster the graphs into 2 clusters and run several graph matching algorithms on these clusters in paired experiments.The algorithms for comparison are Seeded FAQ (denoted by SGM in the figure legend) [39], FAQ [94], the spectral matching algorithm of Umeyama [92], the PATH algorithm [100], and the associated convex relaxation PATH CR, which is solved exactly using Frank-Wolfe methodology [44], and the GLAG algorithm [37].

We select a pair of high-low correlations and seeds: $\vec{m} = [3, 3]$ seeds for $\rho = 0.9$ and $\vec{m} = [5, 5]$ seeds for $\rho = 0.6$, all seeds chosen uniformly at random from the two blocks. The seeds are always used in the embedding and clustering procedure.

Figure 6.2 presents the accuracy of the graph matching algorithms, and Tables 6.2

and 6.1 present the runtime of these algorithms. Among all the matching algorithms, SGM is the only algorithm to use seeded vertices when matching the clusters. Hence it is not surprising that SGM achieves best performance.

In the $\rho = 0.9$ experiment, as we increase the size per block $n \in \{100, 200, 300, 400\}$, the graph matching problem is more difficult, thus resulting in a performance degradation for the nonseeded matching algorithms. Among those nonseeded matching algorithms, PATH and its associated convex relaxation achieve the best performance. PATH CR algorithm scales very well in running time but performs worse as n increases. PATH's running time scales poorly (as does that of the GLAG algorithm), as it has a significantly longer running time than SGM or PATH CR across all values of n. While PATH and PATH CR achieve similar results to SGM for $n = 100, 200, 300$, PATH requires significantly more runtime, and at $n = 400$, PATH CR degrades in performance. Incorporating seeds for GLAG, the PATH algorithm and PATH CR may yield significantly faster run time and less performance degradation as n increases.

The performance of SGM is stable, scaling well in run time and achieving excellent matching performance across all n. This illustrates the importance of leverage information from the seeds. Here the correlation $\rho = 0.9$ is very high, and the matching problem becomes easier. For smaller n, PATH and PATH CR perform similarly as SGM, suggesting that seeds are less important when matching very similar graphs. All matching algorithms, except Umeyama, outperform chance.

We next examine the performance for lower correlated ($\rho = 0.6$) SBMs. SGM

Table 6.1: Mean Runtime of Bijective Graph Matching Algorithms at $\rho = 0.9$

$n =$	100	200	300	400
SGM	0.14	0.78	2.13	4.49
FAQ	0.51	3.12	9.13	16.67
Umeyama	0.09	0.14	0.21	0.34
PATH CR	0.30	1.24	2.96	3.46
PATH	2.21	9.90	15.82	69.31
GLAG	8.53	33.83	109.48	261.72

yields average accuracy over 99% for all selected n, and significantly outperforms all the other nonseeded graph matching algorithms. Note that we need more seeds to achieve the same performance with the lower correlation. In summary, the nonseeded graph matching algorithms PATH, PATH CR, FAQ and GLAG perform significantly better than chance. However PATH and GLAG scale poorly in running time. On the other hand, PATH CR and FAQ scale well in running time, but their matching performance decrease significantly as n increases. In addition, all the algorithms, except SGM, perform significantly worse as ρ decreases. As real data is, at best, weakly correlated, this points to the importance of using seeds in matching graphs in practice.

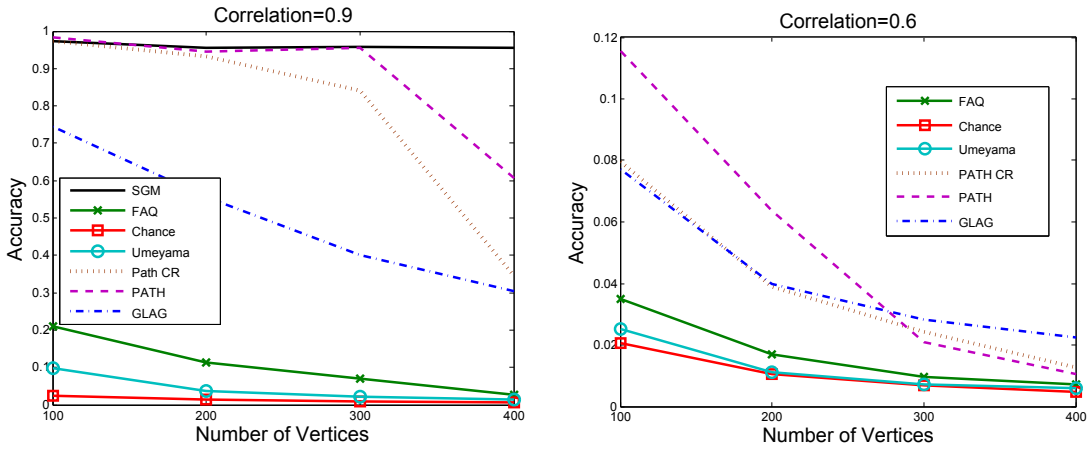


Figure 6.2: Mean accuracy (top) and mean runtime (bottom) for graph matching algorithms for $\rho = 0.9$ (left) and $\rho = 0.6$ (right). For each value of n, we ran 100 Monte Carlo replicates. We do not include the perfect accuracy results for SGM for $\rho = 0.6$ here. When we increase the size per block $n \in \{100, 200, 300, 400\}$, the graph matching problem is more difficult. Hence a degradation in performance is shown for the six nonseeded matching algorithms. PATH and PATH CR have superior performance for small values of n, and their performance drops as n increases. Note the difference in scales for the left and right accuracy plots. The accuracy of the $\rho = 0.6$ experiment is lower than in the $\rho = 0.9$ experiment. This simulation experiment was performed in collaboration with Henry Pao.

Table 6.2: Mean Runtime of Bijective Graph Matching Algorithms at $\rho = 0.6$

$n =$	100	200	300	400
SGM	0.16	0.64	1.76	2.91
FAQ	0.58	3.05	9.09	15.56
Umeyama	0.13	0.19	0.27	0.41
PATH CR	0.42	1.25	2.73	2.36
PATH	3.70	27.00	83.60	142.63
GLAG	8.11	32.41	108.22	235.34

6.4.2 Comparison between SGM and LSGM

In this section, we compare the performance of SGM and LSGM. Theoretically LSGM achieves perfect performance. However, for matching small graphs with $n <$ 2000, SGM is preferred over LSGM. In its matching step, LSGM only considers the connectivity structures within clusters, but not across clusters, while SGM considers the overall connectivity structure. Of course, for matching big graphs, LSGM is the only choice, as SGM has $O(n^3)$ complexity. The LSGM approach contains two main steps: first embed and cluster the two graphs, and then match the subgraphs induced by the clustering accordingly. In this experiment, we explore how much accuracy of LSGM is practically lost for match mid-sized graphs.

We match across two 0.7-correlated SBMs with $K = 3$ blocks, $\vec{n} = (200, 200, 200)$,

with block probability matrix $B = \begin{pmatrix} 0.6 & 0.3 & 0.2 \\ 0.3 & 0.7 & 0.3 \\ 0.2 & 0.3 & 0.7 \end{pmatrix}$, and the number of seeds $m = \{3, 4, 5, 6, 7\}$ drawn uniformly from the 600 vertices.

The performance is seen in Figure 6.3. The matching accuracy of SGM is depicted in the pink curve, and the matching accuracy of LSGM is seen in the blue curve. Note that with only 4 seeds, SGM perfectly matches across the graphs, though LSGM requires 7 seeds for comparable performance.

6.4.3 Robustness to Misspecified K

As mentioned in Section 6.3.2.4, our LSGM algorithm is robustness to misspecified K. To demonstrate this, we simulate two pairs of ρ-correlated SBMs with $\rho \in \{0.6, 0.9\}$, $K = 10$, $\vec{n} = [100, 100, \ldots, 100] \in \mathbb{N}^{10}$, $B \in \{0.3, 0.6\}^{10\times 10}$ with 0.3 on the diagonal and 0.6off-diagonal, and $m = 20$ randomly selected seeds from the $100 \times 10 = 1000$ vertices. We vary the maximum allowed cluster size to be $\{100, 200, 300, 400, 500\}$. The number of Monte Carlo replicates for this experiment is 20. We also included the "oracle" accuracy, which is the maximum possible matching accuracy given the clustering.

As seen in Figure 6.4, LSGM is robust to misspecified number of clusters K, as its performance is stable for all numbers of maximum cluster sizes. Moreover, the accuracy of SGM performed within each cluster is significantly better than all other

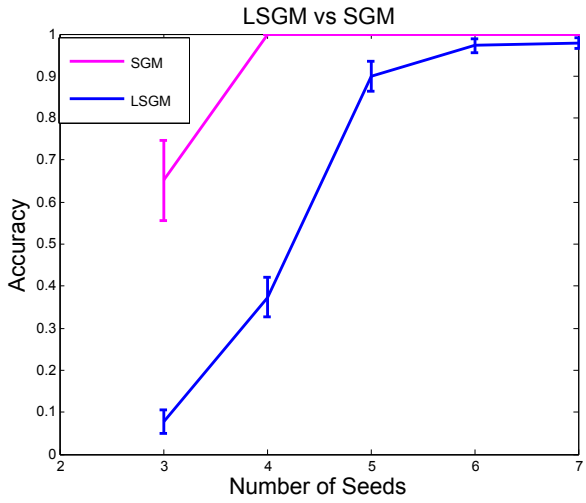


Figure 6.3: The matching accuracy of SGM and LSGM. We simulate two 0.7-correlated SBMs with $K = 3$, $B = \begin{pmatrix} 0.6 & 0.3 & 0.2 \\ 0.3 & 0.7 & 0.3 \\ 0.2 & 0.3 & 0.7 \end{pmatrix}$, $\vec{n} = (200, 200, 200)$, and $m = 3, 4, 5, 6, 7$ seeds randomly assigned to one of the three blocks. The matching performance of SGM seen in the pink curve is higher than the matching performance of LSGM for $m = [3, 4, 5]$. LSGM eventually has comparable performance as SGM for $m = [6, 7]$. This simulation experiment was performed in collaboration with Henry Pao. This figure also appears in Henry Pao's thesis [70].

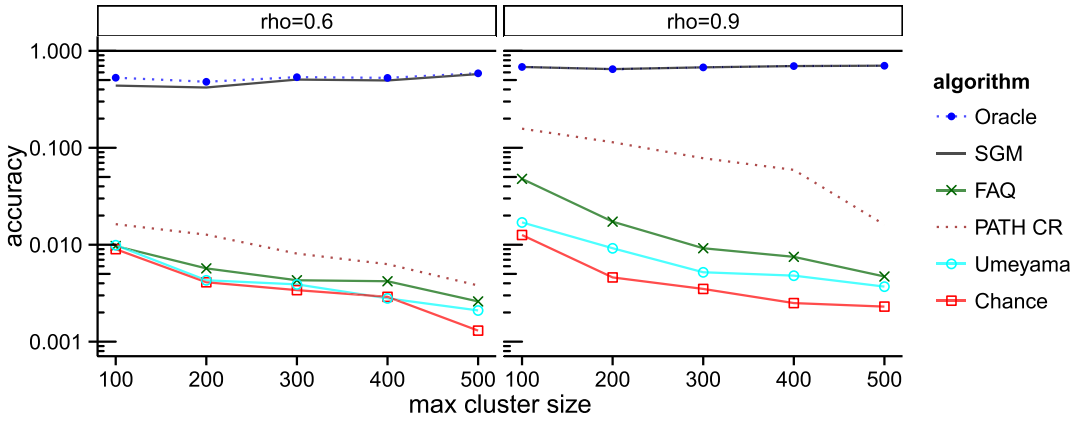


Figure 6.4: Mean matching accuracy (on the log scale) versus maximum allowed cluster size for graph matching algorithms. We simulate two pairs of ρ-correlated SBMs with $\rho \in \{0.6, 0.9\}$, $K = 10$, $\vec{n} = [100, 100, \ldots, 100]$, B as stated in the text and $m = 20$ randomly selected from the 1000 vertices. For each combination of parameters, we run 20 Monte Carlo replicates. Note that LSGM performs significantly better than other matching algorithms, and overlaps the oracle accuracy heavily. Moreover, LSGM performance is stable for different cluster sizes. This demonstrates our point of its robustness property to misspecified number of clusters. Due to scalability issues, GLAG and PATH were not run in this experiment. This simulation experiment was performed in collaboration with Henry Pao.

graph matching algorithms. Indeed, its performance is almost as good as the oracle accuracy.

6.4.4 LSGM Scalability

In this experiment, we examine the scalability of the large graph matching algorithm, particularly the scalability of the matching step. The LSGM algorithm contains three main steps: joint embedding, joint clustering and matching. The first two procedures, as mentioned in Section 6.3.2.2 and Section 6.3.3, are computationally much less costly than the the matching step. Here we explore how parallelization helps LSGM to scale well to big graph data.

We apply the LSGM algorithm on three pairs of SBMs with varying levels of correlations $\rho = 0.3, 0.6, 0.9$. Each SBM has $K = 8$ blocks with block probability matrix $B \in \{0.3, 0.6\}^{8\times 8}$ with 0.3 on the diagonal and 0.6 everywhere else, $\vec{n} = [200, 200, \ldots, 200] \in \mathbb{N}^8$, and $m = 20$ uniformly selected from the total $200 \times 8 = 1600$ vertices. We vary the number of cores $\{1, 2, 3, 4\}$, and examine the run time. The number of Monte Carlo replicates is 200 for this experiment.

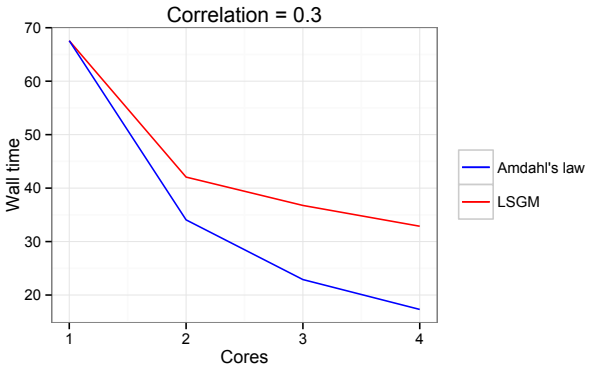
Figure 6.5 presents the resulting wall times as compared to the theoretical best possible scaling provided by Amdahl's law (run on a Genuine Intel laptop: model name: Intel(R) Xeon(R) CPU E31290 @ 3.60GHz with 4 processors). For each of the three correlation levels $\rho \in \{0.3, 0.6, 0.9\}$ and for each of 1 to 4 cores, we obtain the average runtime of each step in LSGM: embedding, Procrustes, clustering and

Cores	Embedding	Procrustes	Clustering	Matching
1	0.53	0.89×10^{-3}	0.17×10^{-2}	67
2	0.54	0.73×10^{-3}	0.18×10^{-2}	41
3	0.54	0.72×10^{-3}	0.18×10^{-2}	36
4	0.54	0.72×10^{-3}	0.15×10^{-2}	32

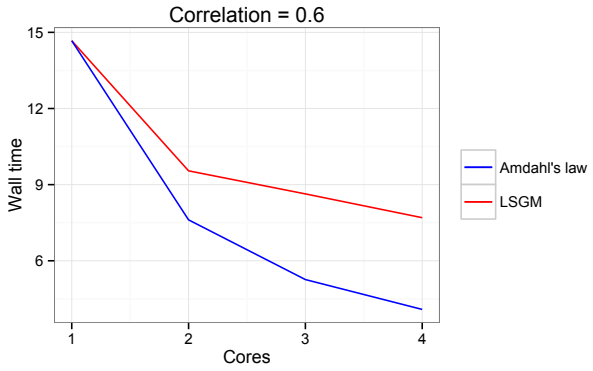
Table 6.3: Examine LSGM scalablity at $\rho = 0.3$. Mean runtime of each step for $\rho = 0.3$ for each core over 200 Monte Carlo replicates. For each selection of cores, the matching run time takes up $> 90\%$ of the total run time, and the run time for embedding, Procrustes, and clustering respectively is not a major contribution to the total run time. As the number of cores increases, the matching time decreases. This indicates the scalability of the large graph matching algorithm.

matching, as seen in Tables 6.3, 6.4 and 6.5. Indeed matching is the time-consuming aspect of the procedure, especially in the low correlation setting. Hence, as discussed before in Section 6.3.2.2, parallelizing the other components of our algorithm would gain less runtime improvements when compared to parallelizing the matching step.

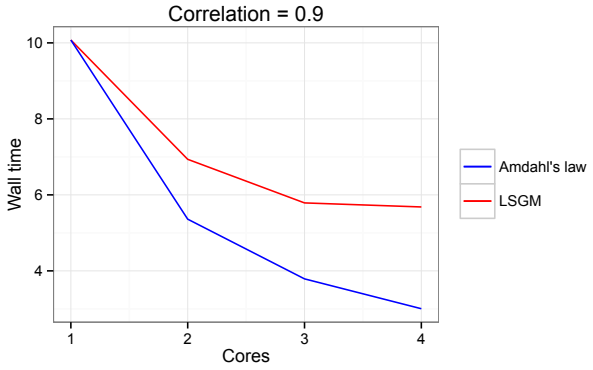
In addition, we also examine the runtime as we vary the number of clusters. We simulate two 0.9-correlated SBMs with $K = 10$, $\vec{n} = [100, 100, \ldots, 100] \in \mathbb{N}^{10}$, $B \in \{0.3, 0.6\}^{10\times10}$ with 0.3 on the diagonal and 0.6 everywhere else, and $m = 20$ randomly selected from the $100 \times 10 = 1000$ vertices. We vary the maximum allowed cluster size to be $\{100, 200, 300, 400, 500\}$. The number of Monte Carlo replicates is



(a) correlation=0.3



(b) correlation=0.6



(c) correlation= 0.9

Figure 6.5: Examine the scalability of large seeded graph matching. We consider three pairs of SBMs with varying levels of correlations $\rho = 0.3, 0.6, 0.9$. Each SBM has $K = 8$ blocks with block probability matrix B as stated in the text, $\vec{n} = [200, 200, \ldots, 200]$, and $m = 20$ uniformly selected from the 1600 vertices. We vary the number of cores $\{1, 2, 3, 4\}$ and examine the runtime. For each selected number of cores, the number of Monte Carlo replicates is 200. We plot the achieved runtime against the theoretical maximum speedup possible when parallelizing as predicted by Amdahl's law. We see a significant decrease in runtime as we include additional cores for computation.

Cores	Embedding	Procrustes	Clustering	Matching
1	0.53	0.89×10^{-3}	0.19×10^{-2}	14
2	0.53	0.73×10^{-3}	0.18×10^{-2}	8.9
3	0.54	0.72×10^{-3}	0.19×10^{-2}	8.0
4	0.54	0.73×10^{-3}	0.19×10^{-2}	7.1

Table 6.4: Examine LSGM scalablity at $\rho = 0.6$. Mean runtime of each step for $\rho = 0.6$ for each core over 200 Monte Carlo replicates. For each selection of cores, the matching run time takes up $> 90\%$ of the total run time, and the run time for embedding, Procrustes, and clustering respectively is not a major contribution to the total run time.

20, and the number of cores used for computation is 12. The mean runtime over 20 Monte Carlo replicates is presented in Table 6.6.

6.4.5 Human Brain Connectomes

We apply LSGM on the diffusion tensor MRI dataset. This dataset contains 42 brain graphs for 21 subjects, each of which has two brain graphs. For each subject, the vertices in the connectomes correspond to voxels in the diffusion tensor MRI brain mask. Edges between vertices are present if there exists at least one neural fiber bundle connecting the voxels. We downsample the voxels by $4\times4\times4$. Then the vertices in the connectomes correspond to voxel regions in the diffusion tensor MRI brain mask,

Cores	Embedding	Procrustes	Clustering	Matching
1	0.53	1.06×10^{-3}	1.06×10^{-2}	9.4
2	0.53	0.74×10^{-3}	0.20×10^{-2}	6.3
3	0.54	0.73×10^{-3}	0.21×10^{-2}	5.2
4	0.54	0.72×10^{-3}	0.20×10^{-2}	5.1

Table 6.5: Examine LSGM scalability at $\rho = 0.9$. Mean runtime of each step for $\rho = 0.9$ for each core over 200 Monte Carlo replicates. For each selection of cores, the matching run time takes up $> 90\%$ of the total run time, and the run time for embedding, Procrustes, and clustering respectively is not a major contribution to the total run time. As the number of cores increases, the matching time decreases. This indicates the scalability of the large graph matching algorithm. The matching time is lower than $\rho = 0.6$ and $\rho = 0.3$, because the graph matching problem is easier at higher correlations.

Maximum cluster size	100	200	300	400	500
Mean runtime	10.2831	24.0464	41.2820	61.8609	86.1164

Table 6.6: We present the mean runtime of LSGM versus different maximum cluster sizes $\{100, 200, 300, 400, 500\}$. We simulate two 0.9-correlated SBMs with $K = 10$, $\vec{n} = [100, 100, \ldots, 100]$, B as stated in the text, and $m = 20$ randomly selected from the 1000 vertices.

and the edges between vertices are present if there exists at least one neural fiber bundle connecting the voxel regions corresponding to the two vertices. The largest connected components (LCC) of these 42 connectomes range from 20,000–30,000 vertices. This dataset is downloadable from http://openconnecto.me/graphs. In this experiment, the between-subject connectomes are Connectome 1 and Connectome 8, and the within-subject connectomes are Connectome 8 and Connectome 29.

The LCC in Connectome 1 contains 22734 vertices, the LCC in Connectome 8 has 21891 vertices, and the LCC Connectome 29 has 22307 vertices. We match the intersection of the LCC's of Connectomes 1 and 8, and the intersection of Connectomes 8 and 29 respectively. The embedding dimension is estimated from the scree plots in both pairs, and it is estimated to be $\hat{d} = 30$. We set the maximum cluster size to be 800. Hence, recursion of clustering will occur if at least one resulted cluster size is over 800. We vary the number of seeds $m \in \{200, 1000, 2000, 5000\}$, and run 30 Monte Carlo replicates for this experiment.

Figure 6.6 shows that the matching accuracy of LSGM is significantly higher for within-subject connectomes than between-subject connectomes. This illustrates the usefulness of LSGM in detecting the similarity structures across large graphs.

We also record the runtime of the four procedures: embedding, Procrustes transformation, clustering and Matching for the brain connectome experiment as seen in Table 6.7. As expected, matching is the most time consuming step, and longer matching time corresponds to higher matching accuracy.

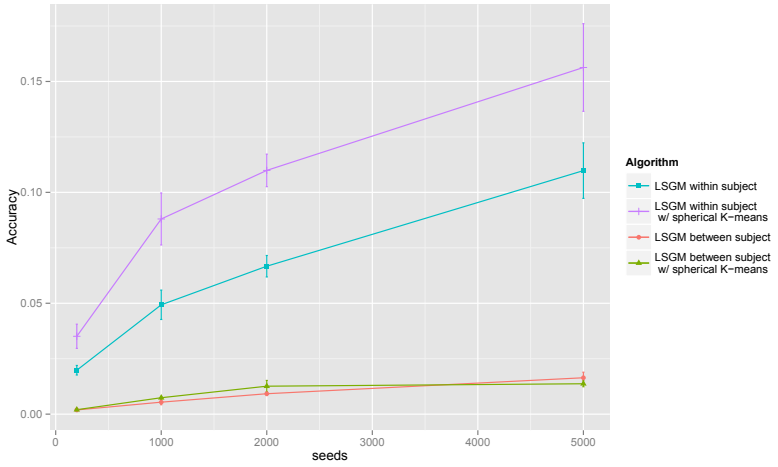


Figure 6.6: Performance of LSGM on the brain connectome. The matching accuracy for Connectomes 8 and 29 (within–subject) and for Connectomes 1 and 8 (between–subject). For the 8–29 pair, $n = 20,541$, $d = 30$. For the 1–8 pair, $n = 18,694$, $d = 30$, we cluster using K-means, reclustering any clusters of size ≥ 800. We vary the number of seeds $m = 200, 1000, 2000,$ and 5000, and run 30 Monte Carlo replicates. The error bars are within 2 standard errors.

Runtime in seconds for connectome experiment

Graph Pair	Number of seeds	Embedding	Procrustes	Clustering	Matching
08-29	200	777.11	1.94	18.16	569.97
08-29	1000	987.08	2.80	18.47	1019.73
08-29	2000	1096.84	3.26	19.29	1592.76
08-29	5000	890.89	2.90	15.19	2902.19
01-08	200	562.82	1.57	13.49	473.46
01-08	1000	754.84	2.44	15.79	883.58
01-08	2000	862.43	2.82	15.14	1495.26
01-08	5000	981.86	3.60	13.55	2698.32

Table 6.7: Runtime for LSGM on the human brain connectomes. For each of the four steps of our procedure, and each combination of seeds and connectomes, we display the average wall time measured in seconds. Matching is the most time intensive step. Embedding to $\mathbb{R}^{30}$ is the second most time intensive step. Hence, parallelization of the embedding step can also contribute to the efficiency of LSGM running speed.

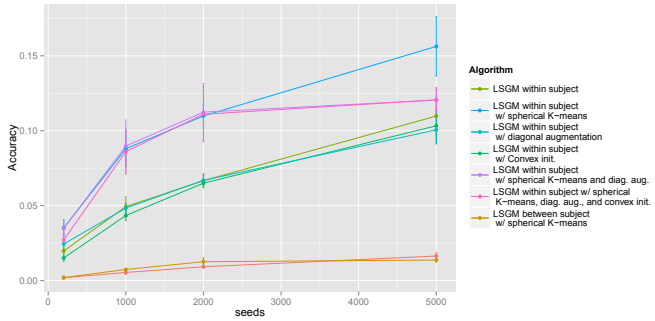
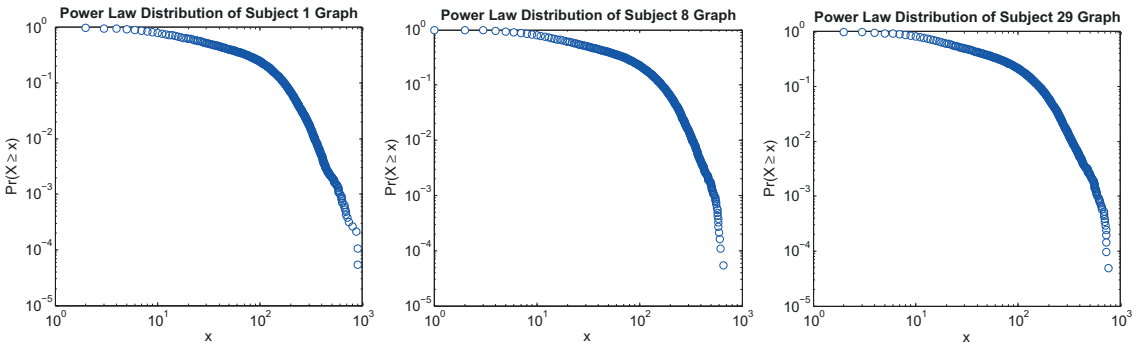


Figure 6.7: LSGM matching accuracy for within–subject and between–subject. Here we examine the performance of various LSGM modifications. For the 8–29 pair, $n = 20,541$, $d = 30$. For the 1–8 pair, $n = 18,694$, $d = 30$, we cluster using K-means, reclustering any clusters of size ≥ 800. We vary the number of seeds $m = 200$, 1000, 2000, and 5000, and run 5 Monte Carlo replicates. We see that including the step of projection onto the sphere, i.e., spherical K-means, significantly improves the matching accuracy.

In Section 6.3.2.7 we discuss three different modifications in order to improve the matching accuracy. Here we compare the performance of LSGM and three modifications of LSGM: LSGM with spherical K-means, LSGM with diagonal augmentation, LSGM with convex initialization. We see that including the step of projection onto the sphere, i.e., spherical K-means, significantly improves the matching accuracy, while other modifications decrease in matching performance. The accuracy plot is seen in Figure 6.7.

Although our theoretical framework is presented for SBMs, we would also like to examine the empirical extendibility of LSGM to other types of graphs. The degree distributions of the three connectomes 1, 8 and 29 on a log-log scale are presented

in Figure 6.8. While our theory is proven in the setting of SBM random graphs, this example shows the applicability of our method in matching graphs with heavy-tailed degree distribution. The relationship between graph degree distribution and discovering matching signals across graphs deserves further investigation.



(a) Connectome 1 (b) Connectome 8 (c) Connectome 29

Figure 6.8: Degree distribution for connectomes 1, 8 and 29. The degrees are plotted on a log-log scale, demonstrating strong evidence of a heavy-tailed degree distribution.

6.5 Discussion

In the age of big data, the ability to extract information from multiple big graph data contributes greatly to joint graph inference. The state-of-the-art SGM algorithm has complexity $O(n^3)$, which limits its feasibility to implement on big graph data. We modify the state-of-the-art seeded graph matching algorithm and propose a divide-and-conquer approach: large seeded graph matching (LSGM), to scale the algorithm to big graph data. Our proposed algorithm LSGM has complexity $O(n^2 d)$,

a remarkable improvement over $O(n^3)$. We present the steps of our proposed algorithm in detail, and prove that the LSGM algorithm performs perfectly under some assumption in the stochastic blockmodel framework.

In the numerical experiments, we explore empirically how much accuracy is lost when applying the LSGM algorithm instead of the SGM algorithm for matching small graphs. We also illustrate the robustness of our algorithm to misspecified number of clusters K, and examine the relationship between correlation, number of seeds and LSGM matching accuracy. We also demonstrate the scalability of our proposed algorithm, and illustrate that the most time intensive step, matching, is parallelizable using our algorithm, which significantly decreases the run time over SGM. In real data experiments, where LSGM is applied on two pairs of brain connectomes, LSGM discovers connectomic similarities when matching across these connectomes. Furthermore, we investigate further changes such as diagonal augmentation, projection onto the sphere, and convex initialization for SGM on the algorithm, in order to improve the empirical matching accuracy. We see that LSGM with projection onto the sphere has a significant improvement in matching accuracy compared to LSGM.

The large seeded graph matching algorithm is our first attempt to scale the seeded graph matching algorithms to big graphs within the joint graph inference framework. Here we present several research aspects worth investigating. In this chapter, we only consider the bijective graph matching framework. Hence, in the LSGM algorithm, we have to ensure that the number of vertices coming from each graph is the same.

Our attempt as discussed in Section 6.3.2.5 solves this issue, but it may not be the best way to allocate vertices. Further effort on designing an automatic clustering algorithm that guarantees to be within the bijective framework, and intelligently allocates the vertices, is worth researching. In addition, we might overcome this hurdle by enabling non-bijective matching on clusters containing different numbers of vertices from each graph. We can non-bijectively match within each cluster by padding the adjacency matrices with empty vertices to make the graphs of commensurate size ([100]), and match the resulting graphs. Vertices matched to isolates could be treated as unmatched, or we could iteratively remove the matched vertices in the larger graph and rematch the graphs, yielding a many–to–many matching.

In Chapter 5, we have presented the inferential task of vertex nomination. The problem of vertex nomination creates a nomination list so that the vertices of interest are abundant at the top of the list. Although vertex nomination is formulated under the single graph inference framework, one of our proposed nomination schemes: the likelihood maximization vertex nomination scheme, borrows from the state–of–the–art SGM algorithm to solve its maximum likelihood optimization. The likelihood maximization vertex nomination scheme inherits the computational complexity $O(n^3)$ from SGM, and thus it is limited to implement for graphs with vertices < 1500. Our proposed LSGM has complexity $O(n^2 d)$, which is a great improvement over SGM. Employing LSGM in the likelihood maximization vertex nomination scheme may scale this scheme to big graph data. This open question is also worth pondering.

Chapter 7

A Joint Graph Inference Case Study: the *Caenorhabditis elegans* Neural Connectomes

In this chapter, we present our work [18] on a joint graph inference case study for a pair of neural connectomes of the *Caenorhabditis elegans*. The *C. elegans* connectomes consist of 253 non-isolated neurons with known functional attributes, and there are two types of synaptic connectomes, resulting in a pair of graphs. We formulate our joint graph inference from the perspectives of seeded graph matching and joint vertex classification. Our results suggest that connectomic inference should proceed in the joint space of the two neural connectomes, which is of independent neuroscientific importance.

7.1 Introduction

The *Caenorhabditis elegans* (*C.elegans*) is a non-parasitic, transparent roundworm approximately one millimeter in length. The majority of *C.elegans* are female hermaphrodites. [65] first described the worm in 1900 and named it *Rhabditis elegans*. It was later categorized under the subgenus *Caenorhabditis* by [69], and then, in 1955, raised to the generic status by Ellsworth Dougherty, to whom much of the recognition for choosing *C.elegans* as a model system in genetics is attributed [77]. The long name of this nematode mixes Greek and Latin, where *Caeno* means recent, *rhabditis* means rod-like, and *elegans* means elegant.

Research on *C.elegans* rose to prominence after the nematode was adopted as a model organism: an easy-to-maintain non-human species widely studied, so that discoveries on this model organism might offer insights for the functionality of other organisms. The discoveries of caspases [99], RNA interference [38], and microRNAs [56] are among some of the notable research using *C.elegans*.

Connectomes, the mapping of neural connections within the nervous system of an organism, provide a comprehensive structural characterization of the neural network architecture, and represent an essential foundation for basic neurobiological research. Applications based on the discovery of the connectome patterns and the identification of neurons based on their connectivity structure give rise to significant challenges and promise important impact on neurobiology. Recently, there has been an increasing interest in the network properties of *C.elegans* connectomes. The hermaphrodite

C.elegans is the only organism with a fully constructed connectome [83], and has one of the most highly studied nervous systems.

Studies on the *C.elegans* neural connectomes traditionally focus on utilizing one single connectome alone [72, 83, 89, 93], although there are many connectomes available. Notably, Varshney et al. [93] discovered structural properties of the *C.elegans* neural connectomes via analyzing the graph statistics of the neural connectomes. Pavlovic et al. [72] estimated the community structure of the neural connectomes, and their findings are compatible with known biological information on the *C.elegans* nervous system.

Our new statistical approach of joint graph inference looks instead at jointly utilizing the paired chemical and electrical connectomes of the hermaphrodite *C.elegans*. We formulate our inference framework from the perspectives of seeded graph matching and joint vertex classification, which we explain in Section 7.3. This framework gives a way to examine the structural similarity preserved across multiple connectomes within species, and make quantitative comparisons between joint connectome analysis and single connectome analysis. We find that the optimal inference for the information-processing properties of the connectome should proceed in the joint space of the *C.elegans* connectomes, and using the joint connectomes predicts neuron attributes more accurately than using either connectome alone.

7.2 The Hermaphrodite *C.elegans* Connectomes

The hermaphrodite *C.elegans* neural connectomes consist of 302 labeled neurons for each organism. The *C.elegans* somatic nervous system has 279 neurons connecting to each other across synapses. There are many possible classifications of synaptic types. Here we consider two types of synaptic connections among these neurons: chemical synapses and electrical junction potentials. These two types of connectivity result in two synaptic connectomes consisting of the same set of neurons.

We represent the neural connectomes as graphs $G = (V, E)$. In a neural connectome, the vertices represent neurons, and the edges represent synapses. For the hermaphrodite *C.elegans* worm, the chemical connectome G_c is weighted and directed. The electrical gap junctional connectome G_g is weighted and undirected. This is consistent with an important characteristic of electrical synapses – they are bidirectional [75]. The chemical connectome G_c has 3 loops and no isolated vertices, while the electrical gap junctional connectome G_g has no loops and 26 isolated vertices. Both connectomes are sparse. The chemical connectome G_c has 2194 directed edges out of $279 \cdot 278$ possible ordered neuron pairs, resulting in a sparsity level of approximately 2.8%. The electrical gap junctional connecome G_g has 514 undirected edges out of $\binom{279}{2}$ possible unordered neuron pairs, resulting in a sparsity level of approximately 1.3%.

In our analysis, we are interested in the $279 - 26 = 253$ non-isolated neurons in the hermaphrodite *C.elegans* somatic nervous system. Each of these 253 neurons can be classified in a number of ways, including into 3 non-overlapping connectivity based types: sensory neurons (27.96%), interneurons (29.75%) and motor neurons (42.29%). We also assume that all graphs are undirected, unweighted and non-loopy. That is, the adjacency matrices are symmetric, binary and hollow. Here we will work with binary, symmetric and hollow adjacency matrices of the neural connectomes throughout. In particular, we symmetrize A by $A \leftarrow A + A^T$, then binarize A by threshholding the positive entries of A to be 1 and 0 otherwise, and finally set the diagonal entries of A to be zero. Indeed, we focus on the existence of synaptic connections, and the occurrence of loops is low (3 loops in G_c and none in G_g) that we can ignore it.

An image of the *C. elegans* worm body is seen in Figure 7.1. The pair of the neural connectomes are visualized in Figure 7.2. In the chemical connectome G_{c}, the interneurons are heavily connected to the sensory neurons. The sensory neurons are connected more frequently to the motor neurons and interneurons than amongst themselves. In the electrical gap junction potential connectome G_{g}, the motor neurons are heavily connected to the interneurons. The sensory neurons are connected more frequently to the motor neurons and interneurons than among themselves. The connectome dataset is accessible at `http://openconnecto.me/herm-c-elegans`. Figure 7.3 presents the adjacency matrices of the paired *C.elegans* connectomes.

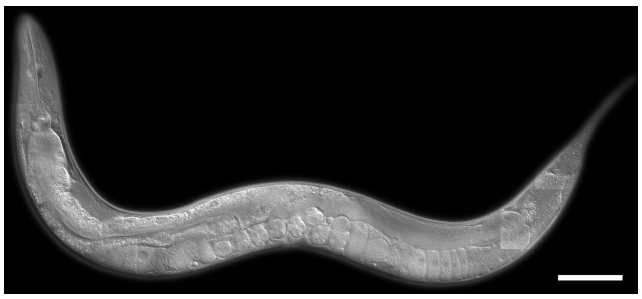


Figure 7.1: An image of the *Caenorhabditis elegans* (*C.elegans*) roundworm. The image is available at `http://post.queensu.ca/~chinsang/research/c-elegans.html`.

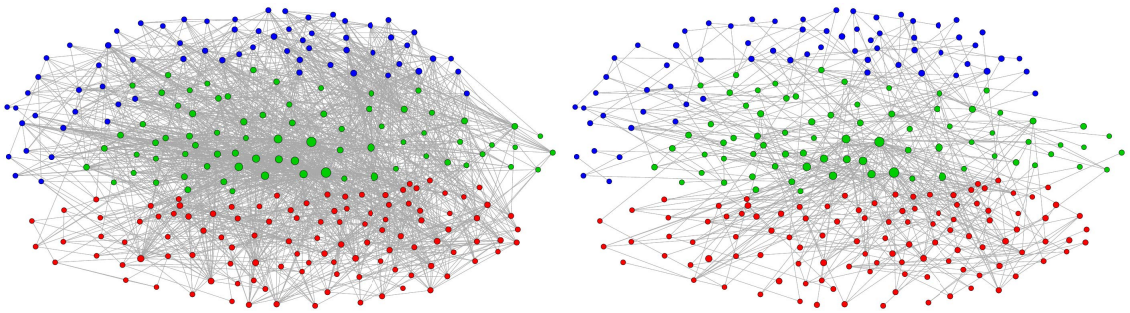


Figure 7.2: The pair of *C. elegans* neural connectomes visualized as graphs. Red nodes correspond to motor neurons, green nodes correspond to interneurons, and blue nodes correspond to sensory neurons. (Left) The chemical connectome G_{c}. (Right) The electrical gap junctional connectome G_{g}. Both synaptic connectomes are sparse, while G_{g} is much sparser than G_{c}. A similar connectivity pattern is seen across both connectomes.

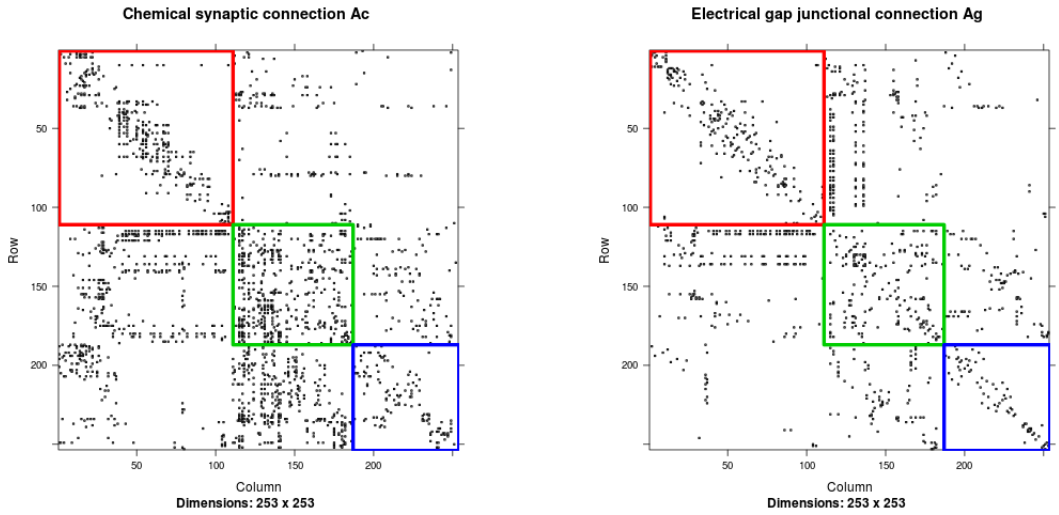


Figure 7.3: (Left): The adjacency matrix A_c of G_c sorted according to the neuron types. (Right): The adjacency matrix A_g of G_g sorted according to the neuron types. The red block corresponds to the connectivity among the motor neurons, the green block corresponds to the connectivity among the interneurons, and the blue block corresponds to the connectivity among the sensory neurons.

7.3 Joint Graph Inference

We consider an inference framework in the joint space of the *C. elegans* neural connectomes, which we refer to as joint graph inference. We focus on two aspects of joint graph inference: seeded graph matching and joint vertex classification.

7.3.1 Seeded Graph Matching

See Chapter 6.2 for a detailed description of seeded graph matching. In the following sub-section, we mainly introduce the problem of joint vertex classification.

7.3.2 Joint Vertex Classification

When we observe the adjacency matrix $A \in \{0,1\}^{n\times n}$ on n vertices and the class labels $\{Y_i\}_{i=1}^{n-1}$ associated with the first $(n-1)$ training vertices, the task of vertex classification is to predict the label Y of the test vertex v. In this case study, the class labels are the neuron types: motor neurons, interneurons and sensory neurons. In this work, we assume the correspondence between the vertex sets across the two graphs is known. Given two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ where $V = \{v_1, \ldots, v_{n-1}, v\}$, and given the class labels $\{Y_i\}_{i=1}^{n-1}$ associated with the first $(n-1)$ training vertices, the task of joint vertex classification predicts the label of a test vertex v using information jointly from G_1 and G_2.

Fusion inference merges information on multiple disparate data sources in order

to obtain more accurate inference than using only single source. Our joint vertex classification consists of two main steps: first, a fusion information technique, namely omnibus embedding methodology by [74]; and secondly, the inferential task of vertex classification.

Omnibus embedding proceeds as follows. Given G_1 and G_2, we compute their respective dissimilarity matrices $D_{G_1} \in \mathbb{R}^{n \times n}$ and $D_{G_2} \in \mathbb{R}^{n \times n}$. Then we construct an omnibus matrix $M = \begin{pmatrix} D_{G_1} & \Lambda \\ \Lambda & D_{G_2} \end{pmatrix} \in \mathbb{R}^{2n \times 2n}$, where the off-diagonal block is $\Lambda = \frac{1}{2}(D_{G_1} + D_{G_2})$. We consider classical multidimensional scaling (CMDS) embedding [10, 25, 87, 88] of M as $2n$ points into $\mathbb{R}^d$. Let $U = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \in \mathbb{R}^{2n \times d}$ denote the resulted joint embedding, where $U_1 \in \mathbb{R}^{n \times d}$ is the joint embedding corresponding to G_1, and $U_2 \in \mathbb{R}^{n \times d}$ to G_2. Our inference task is vertex classification. Let $\mathcal{T}_{n-1} := U_1([n-1], :) \in \mathbb{R}^{(n-1) \times d}$ denote the training set containing the first $n-1$ vertices. We train a classifier on $\mathcal{T}_{n-1}$, and classify the test vertex v. The algorithm is presented in Algorithm 9. A depiction of the joint vertex classification procedure is seen in Figure 7.4.

We demonstrate that fusing both pairs of the neural connectomes generates more accurate inference results than using a single source of connectome alone. We consider single vertex classification for comparison, which computes the dissimilarity matrix $D_G \in \mathbb{R}^{n \times n}$ of one graph, embeds D_G to $\mathbb{R}^d$ via CMDS, and classifies on the embedded space. The algorithm is seen in Algorithm 10. A depiction of the single vertex

Algorithm 9 Joint Vertex Classification

Goal Classify the test vertex v_n whose label is Y using both graphs jointly.

Input Let G_1 and G_2 be two graphs on vertices $\{v_1, v_2, \ldots, v_{n-1}, v_n\}$, where the first $(n-1)$ vertices are associated with known labels. Let $A_1 \in \{0,1\}^{n \times n}$ be the adjacency matrix of G_1. Let $A_2 \in \{0,1\}^{n \times n}$ be the adjacency matrix of G_2. Let D be a specified dissimilarity measure. Let d be a selected embedding dimension.

Step 1: Dissimilarity. Compute the dissimilarity matrices D_{G_1} and D_{G_2} of G_1 and G_2 respectively using D.

Step 2: Omnibus. Compute the Omnibus matrix M

$$M = \begin{pmatrix} D_{G_1} & \Lambda \\ \Lambda & D_{G_2} \end{pmatrix} \in \mathbb{R}^{2n \times 2n}, \tag{7.1}$$

where $\Lambda = \frac{1}{2}(D_{G_1} + D_{G_2})$.

Step 3: Embedding. Embed the ominibus matrix M into d-dimensional space via CMDS. $U = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \in \mathbb{R}^{2n \times d}$. $U_1 \in \mathbb{R}^{n \times d}$ is the joint embedding corresponding to G_1, and $U_2 \in \mathbb{R}^{n \times d}$ to G_2.

Step 4: Classification. Train on $U_1[1:(n-1),:] \in \mathbb{R}^{(n-1) \times d}$ and classify v_n.

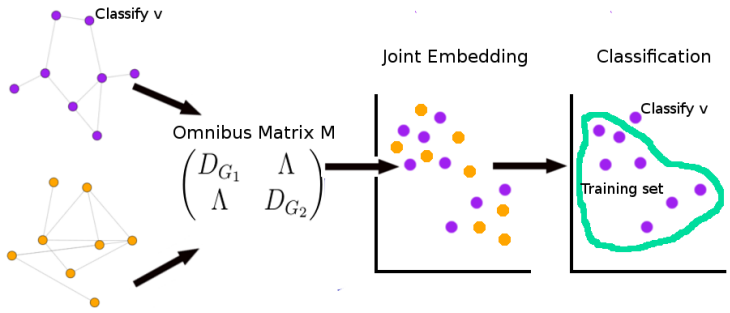


Figure 7.4: A depiction of joint vertex classification. An illustration of joint vertex classification, which embeds the joint dissimilarity matrix – the omnibus matrix – and classifies on the embedded space.

classification procedure is seen in Figure 7.5.

7.4 Discoveries from the Joint Space of the Neural Connectomes

7.4.1 Finding the Correspondence between the Chemical and the Electrical Connectomes

We apply seeded graph matching on the paired *C.elegans* neural connectomes, and discover the underlying structure preserved across the chemical and the electrical connectomes. Figure 7.6 presents the errorbar plot of the seeded graph matching

Algorithm 10 Single Vertex Classification

Goal Classify the test vertex v whose label is Y using a single graph.

Input Let G be a graph on vertices $\{v_1, v_2, \ldots, v_{n-1}, v_n\}$, where the first $(n-1)$ vertices are associated with known labels. Let $A \in \{0,1\}^{n \times n}$ be the adjacency matrix of G. Let D be a specified dissimilarity measure. Let d be a selected embedding dimension.

Step 1: Dissimilarity. Compute the dissimilarity matrix D_G of G.

Step 2: Embedding. Embed the dissimilarity D_G matrix into d-dimensional space via CMDS. $U \in \mathbb{R}^{n \times d}$.

Step 3: Classification. Train on $U[1:(n-1),:] \in \mathbb{R}^{(n-1) \times d}$ and classify v_n.

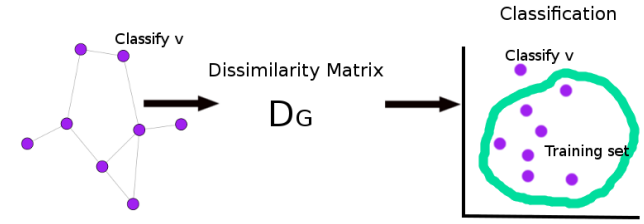


Figure 7.5: A depiction of single vertex classification. An illustration of single vertex classification, which embeds one single dissimilarity matrix, and classifies on the embedded space.

accuracy $\delta(m)$, plotted in black, against the number of seeds $m \in \{0, 20, 40, \ldots, 180\}$. For each selected number of seeds m, we randomly and independently select 100 seeding sets S_1. For each seeding set S_1 at a given number of seeds m, we apply the state-of-the-art seeded graph matching algorithm [39]. The mean accuracy $\delta(m)$ is obtained by averaging the accuracies over the 100 Monte Carlo replicates at each m. As m increases, the matching accuracy improves. This is expected, because more seeds give more information, making the SGM problem less difficult. The chance accuracy, plotted in brown dashed line, at each m is $\frac{1}{n-m}$, which does not increase significantly as m increases.

We must note two significant neurological implications based on our graph matching result. First, SGM on the pair of connectomes indicate that the chemical and the electrical connectomes have statistically significant similar neurological structure. The second significant implication is more intriguing: If the performance of SGM on the chemical and the electrical connections were perfect, then one should only need to analyze one of the paired neural connectomes, and analysis on either connectome should produce very similar results. If performance of SGM on the chemical and the electrical connections were no better than random, then it suggests that further analysis should proceed on the connectomes separately and individually. The seeded graph matching result on the *C. elegans* neural connectome is much more significant than chance but less than a perfect matching. This indicates that the subsequent inference should be performed in the joint space of both the chemical and the electrical

connectomes. This discovery is noted in [60].

7.4.2 Predicting Neuron Types from the Joint Space of the Chemical and the Electrical Connectomes

The result of SGM on the *C.elegans* neural connectomes demonstrates the advantage of inference in the joint space of the neural connectomes, and provides a statistical motivation to apply our proposed joint vertex classification approach. Furthermore, the neurological motivation of applying joint vertex classification stems from illustrating a methodological framework to understand the coexistence and significance of chemical and electrical synaptic connectomes.

We apply joint vertex classification and single vertex classification on the paired *C.elegans* neural connectomes, and compare the classification performance. The validation is done via leave-one-out cross validation. Here we do not investigate which specific dissimilarity D, embedding dimension d or classifier are optimal for our classification task. In this experiment, we consider using three specific dissimilarities: shortest path, Jaccard dissimilarity, and inverse log-weighted dissimilarity [2]. We choose support vector machine classifier with radial basis [24] for the classification step.

The three paired plots in Figure 7.7 present the misclassification errors against the

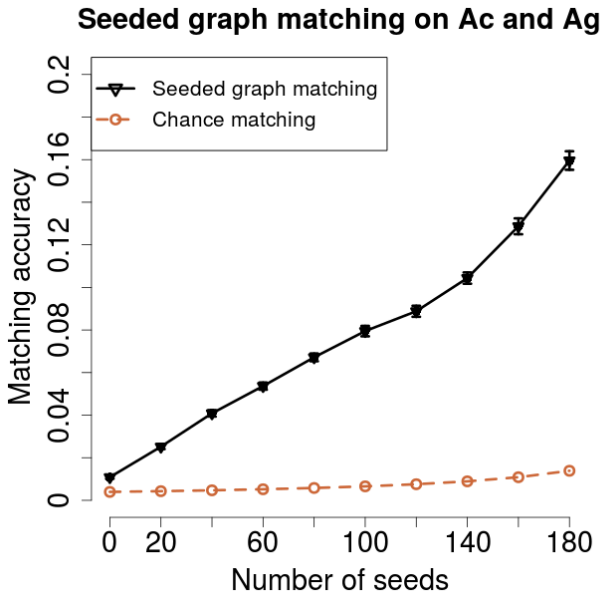


Figure 7.6: For each selected number of seeds $m \in \{0, 20, 40, \ldots, 180\}$, we randomly select 100 independent seeding sets S_1 and apply SGM. The SGM mean accuracy $\delta(m)$, plotted in black, is obtained by averaging the accuracies over the 100 Monte Carlo replicates. As the number of seeds m increases, the accuracy increases. The chance accuracy, plotted in brown dashed line, is much lower than the SGM accuracy. This suggests that a significant similarity exists between the two types of synapse connections. The SGM performance on the *C. elegans* neural connectome is much more significant than chance but less than a perfect matching, indicating the optimal inference must proceed in the joint space of both neural connectomes.

embedding dimensions $d \in \{2, 5, 8, \ldots, 116, 119\}$ for each choice of dissimilarity. For all three selected dissimilarities and for all the embedding dimensions, the joint vertex classification (plotted in black) outperforms the single vertex classification (plotted in magenta). Jaccard dissimilarity has the lowest classification error among the three selected dissimilarities.

The superior performance of the joint vertex classification over the single vertex classification has an important neuroscientific implication. In many animals, the chemical synapses co-exist with the electrical synapses. Modern understanding of co-existence of chemical and electrical synaptic connectomes suggest such a coexistence has physiological significance. We discover that using both chemical and electrical connectomes jointly generates better classification performance than using one connectome alone. This may serve as a first step towards providing a methodological and quantitative approach towards understanding the coexistent significance.

7.5 Summary and Discussion

The paired *Caenorhabditis elegans* neural connectomes have become a fascinating dataset for motivating a better understanding of the nervous connectivity systems. We have presented the unique statistical approach of joint graph inference – inference in the joint graph space – to study the worm's connectomes. Utilizing jointly the chemical and the electrical connectomes, we discover statistically significant similar-

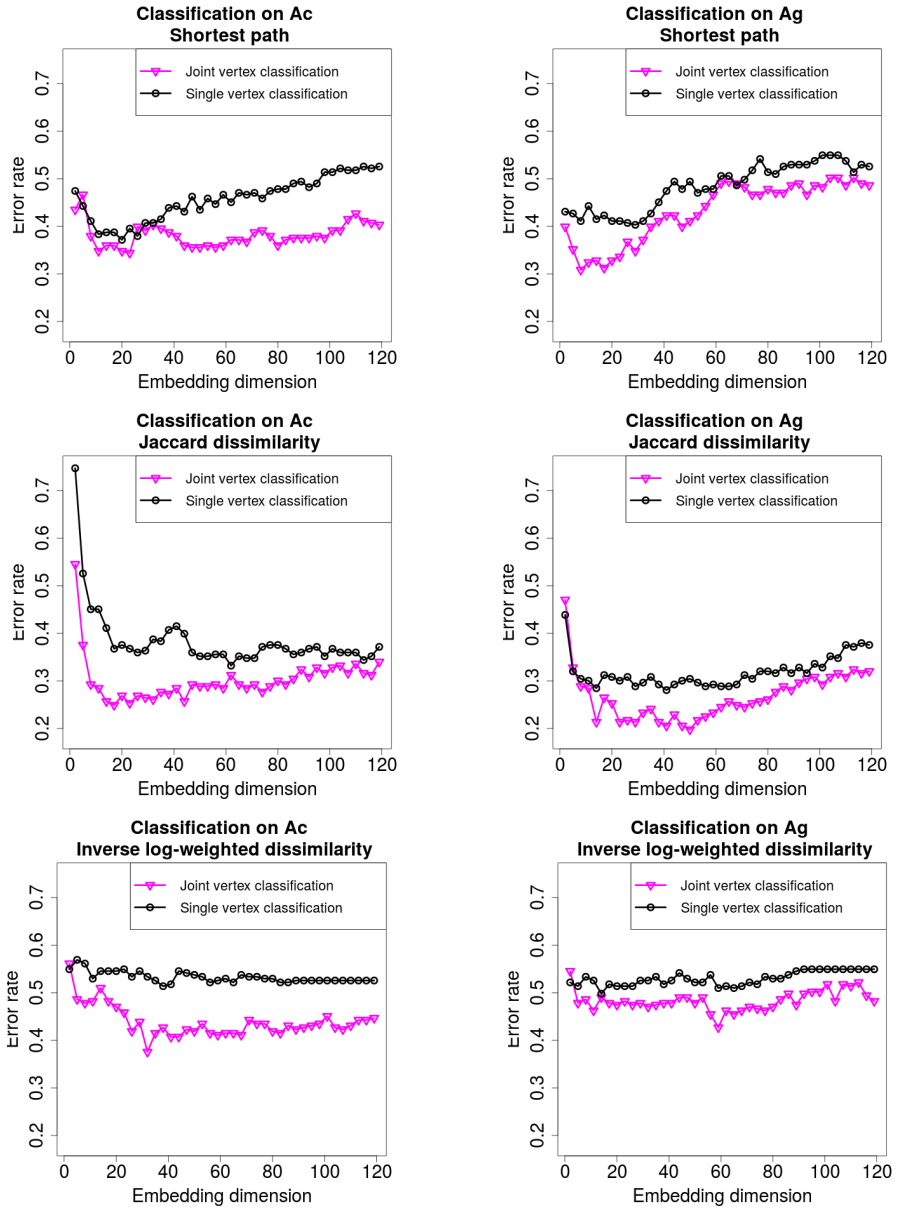


Figure 7.7: Classification performance of joint and single vertex classification for three dissimilarities: (top) shortest path, (middle) Jaccard dissimilarity, (bottom) inverse log-weighted dissimilarity. (Left) Classification on A_c. (Right) Classification on A_g. For all embedding dimensions, the error rate of joint vertex classification (magenta) is lower than the single vertex classification (black).

ity preserved across the two synaptic connectome structures. Our result of seeded graph matching indicates that the optimal inference on the information-processing properties of the neural connectomes must proceed in the joint space of the paired graphs.

The development of seeded graph matching provides a strong statistical motivation for joint vertex classification, where we predict neuron types in the joint space of the paired connectomes. Joint vertex classification outperforms the single vertex classification against all embedding dimensions for our different choices of dissimilarity measures. Fusion inference using both the chemical and the electrical connectomes produces more accurate results than using one (either one) connectome alone, and enhances our understanding of the *C.elegans* connectomes. The chemical and the electrical synapses are known to coexist in most organisms. Our proposed joint vertex classification provides a methodological and quantitative framework for understanding the significance of the coexistence of the chemical and the electrical synapses. Further development of joint graph inference is a topic of ongoing investigation in both neuroscience and statistics.

Chapter 8

Conclusion

In this dissertation, we discuss two aspects of pattern recognition on random graphs, namely single graph inference – inference on a single observed graph, and joint graph inference – inference in the joint space of two graphs. We are mainly concerned with the stochastic blockmodels, where we develop several inference methodologies for vertex classification, vertex clustering, vertex nomination, scalable graph matching and joint vertex classification.

In Chapter 2, we introduce several important random graph models which are essential for our study of pattern recognition on graphs. Details of the latent position graph, the random dot product graph, and the stochastic blockmodel are presented. We prove reparameterization of the stochastic blockmodel into the random dot product model, and present the adjacency spectral embedding method.

Under the single graph inference framework, we intend to extract information us-

ing only one single graph. In Chapter 3, we present the task of vertex classification, and propose sparse representation vertex classification. Our proposed sparse representation vertex classifier represents the test vertex as a sparse linear combination of the training vertices. This classifier is robust to data contamination, as demonstrated in our proposed contamination stochastic models. We compare the performance of sparse representation with two classifiers following adjacency spectral embedding. Our proposed classifier outperforms the other two classifiers. For stochastic block-models, when the model dimension is known, the nearest neighbor classifier following adjacency spectral embedding and the linear discriminant analysis following adjacency spectral embedding are both Bayes optimal such that they achieve the lowest possible misclassification error. However the model dimension is not known in practical graph inference. We see that the sparse representation classifier has superior and stable performance over the other two embedding-based classifiers.

In Chapter 4, we propose a vertex clustering approach, which employs adjacency spectral embedding and subsequently a model-based clustering algorithm. We focus on illustrating the practical value of our proposed approach in online advertising. We explain the basic concepts in online advertising and the business motivation for vertex clustering approach. We utilize a real online campaign dataset, and demonstrate the advantage of using our clustering algorithm to mining information that is valuable for business. We use business metrics, such as revenue, ad impressions, or website topics related to online advertising, to evaluate the clustering significance. We find that

our approach discovers more significant clusters, indicating its advantage in online advertising.

In Chapter 5, we present the inferential task of vertex nomination. Assume the graph is a realization of a stochastic blockmodel. Suppose we are interested in a block of vertices in a graph. The inferential task of vertex nomination is to create a nomination list, where the interesting vertices are abundant at the top of the list. We propose several vertex nomination schemes: canonical vertex nomination scheme, likelihood maximiazation vertex nomination scheme, spectral partitioning vertex nomination scheme, and canonical sampling vertex nomination scheme. The canonical vertex nomination scheme is the best possible scheme measured using the mean average precision. However it is limited to few tens of vertices. The likelihood maximization vertex nomination achieves good nomination performance on graphs of thousands of vertices. The canonical sampling vertex nomination scheme is a scalable version of the canonical vertex nomination, so it preserves the "gold standard" property and scales to big graphs. This concludes our presentation of single graph inference framework.

Under the joint graph inference, we discover information in the joint space of multiple graphs: in this dissertation, we consider two graphs. In Chapter 6, we present the problem of seeded graph matching (SGM) and propose our scalable seeded graph matching to large graphs. Our proposed large seeded graph matching (LSGM) algorithm is a divide-and-conquer approach, which jointly embeds two graphs, transform

them in the same space, cluster the verticces, and apply graph matching within the clusters. We compare SGM with several existing state-of-the-art graph matching algorithms, and illustrate the usefulness of seeds in improving graph matching performance. We demonstrate the scalability, robustness and the performance trade-off between SGM and LSGM in simulation. We also apply LSGM to large human brain connectomes, and discuss the values of LSGM in neuroscience.

In Chapter 7, we present a joint graph inference case study on the paired *Caenorhabditis elegans* neural connectomes. In this case study, we explore two aspects of joint graph inference, namely seeded graph matching and joint vertex classification. The result of seeded graph matching indicates that the optimal inference regarding the information processing-properties must proceed in the joint space of the neural connectomes. The result of joint vertex classification and its superior performance over single vertex classification shows that joint connectome analysis produces more accurate inference than single connectome analysis. Our proposed joint vertex classification provides a methodological and quantitative framework towards understanding the significance of the coexistence of the chemical and the electrical synapses.

Bibliography

[1] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from data*. AMLBook, 2012.

[2] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

[3] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43. ACM, 2005.

[4] Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. In *Advances in Neural Information Processing Systems*, pages 33–40, 2009.

[5] David Aldous and Jim Fill. Reversible markov chains and random walks on graphs, 2002.

[6] Avanti Athreya, Vince Lyzinski, David J Marchette, Carey E Priebe, Daniel L

Sussman, and Minh Tang. A central limit theorem for scaled eigenvectors of random dot product graphs. *arXiv preprint arXiv:1305.7388*, 2013.

[7] James Baglama and Lothar Reichel. Augmented implicitly restarted lanczos bidiagonalization methods. *SIAM Journal on Scientific Computing*, 27(1):19–42, 2005.

[8] Sivaraman Balakrishnan, Min Xu, Akshay Krishnamurthy, and Aarti Singh. Noise thresholds for spectral clustering. In *Advances in Neural Information Processing Systems*, pages 954–962, 2011.

[9] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[10] Ingwer Borg and Patrick JF Groenen. *Modern multidimensional scaling: Theory and applications*. Springer, 2005.

[11] Matthew Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear algebra and its applications*, 415(1):20–30, 2006.

[12] T Tony Cai and Lie Wang. Orthogonal matching pursuit for sparse signal recovery with noise. *IEEE Transactions on Information Theory*, 57(7):4680–4688, 2011.

BIBLIOGRAPHY

[13] Emmanuel J Candes and Terence Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

[14] Sourav Chatterjee. Matrix estimation by universal singular value thresholding. *arXiv preprint arXiv:1212.1247*, 2012.

[15] Kamalika Chaudhuri, Fan Chung Graham, and Alexander Tsiatas. Spectral clustering of graphs with general degrees in the extended planted partition model. In *COLT*, pages 35–1, 2012.

[16] Li Chen and Matthew Patton. Stochastic blockmodeling for online advertising. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intellilgence*, 2015.

[17] Li Chen, Cencheng Shen, Joshua Vogelstein, and Carey Priebe. Robust vertex classification. *Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence. arXiv preprint arXiv:1311.5954*, 2013.

[18] Li Chen, Joshua Vogelstein, Vince Lyzinski, and Carey Priebe. A joint graph inference case study: the caenorhabditis elegans neural connectomes. *Manuscript in Preparation*, 2014.

[19] Li Chen, Henry Pao, Donniell E Fishkind, Vince Lyzinski, and Carey E Priebe. Canonical sampling vertex nomination scheme for large graphs. *Manuscript in preparation*, 2015.

[20] Yudong Chen, Sujay Sanghavi, and Huan Xu. Clustering sparse graphs. In *Advances in neural information processing systems*, pages 2204–2212, 2012.

[21] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.

[22] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18(03):265–298, 2004.

[23] Glen A Coppersmith and Carey E Priebe. Vertex nomination via content and context. *arXiv preprint arXiv:1201.4118*, 2011.

[24] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[25] Trevor F Cox and Michael AA Cox. *Multidimensional scaling*. CRC Press, 2010.

[26] Wei Dai and Olgica Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Transactions on Information Theory*, 55(5):2230–2249, 2009.

[27] J-J Daudin, Franck Picard, and Stéphane Robin. A mixture model for random graphs. *Statistics and computing*, 18(2):173–183, 2008.

[28] Chandler Davis and William Morton Kahan. The rotation of eigenvectors by a perturbation. iii. *SIAM Journal on Numerical Analysis*, 7(1):1–46, 1970.

[29] Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition*, volume 31. New York: Springer, 1996.

[30] David L Donoho. For most large underdetermined systems of linear equations the minimal l_1-norm solution is also the sparsest solution. *Communications on pure and applied mathematics*, 59(6):797–829, 2006.

[31] David L Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l_1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

[32] Richard O Duda and Peter E Hart. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[33] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[34] S Dumais, G Furnas, T Landauer, S Deerwester, and S Deerwester. Latent semantic indexing. In *Proceedings of the Text Retrieval Conference*, 1995.

[35] Michael Elad and Alfred M Bruckstein. A generalized uncertainty principle and sparse representation in pairs of bases. *IEEE Transactions on Information Theory*, 48(9):2558–2567, 2002.

[36] Mário AT Figueiredo, Robert D Nowak, and Stephen J Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4): 586–597, 2007.

[37] Marcelo Fiori, Pablo Sprechmann, Joshua Vogelstein, Pablo Musé, and Guillermo Sapiro. Robust multimodal graph matching: Sparse coding meets graph matching. pages 127–135, 2013.

[38] Andrew Fire, SiQun Xu, Mary K Montgomery, Steven A Kostas, Samuel E Driver, and Craig C Mello. Potent and specific genetic interference by double-stranded rna in caenorhabditis elegans. *nature*, 391(6669):806–811, 1998.

[39] Donniell E Fishkind, Sancar Adali, and Carey E Priebe. Seeded graph matching. *arXiv preprint arXiv:1209.0367*, 2012.

[40] Donniell E Fishkind, Vince Lyzinski, Henry Pao, Li Chen, and Carey E Priebe. Vertex nomination schemes for membership prediction. *Submitted to Annals of Applied Statistics. arXiv preprint arXiv:1312.2638*, 2013.

[41] Donniell E Fishkind, Daniel L Sussman, Minh Tang, Joshua T Vogelstein, and Carey E Priebe. Consistent adjacency-spectral partitioning for the stochastic block model when the model parameters are unknown. *SIAM Journal on Matrix Analysis and Applications*, 34(1):23–39, 2013.

[42] Chris Fraley and Adrian E Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631, 2002.

[43] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[44] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[45] Richard Goldschmidt. Das nervensystem von ascaris lumbricoides und megalocephala, i. *Z wiss Zool*, 90:73–136, 1908.

[46] Rémi Gribonval and Morten Nielsen. Sparse representations in unions of bases. *IEEE Transactions on Information Theory*, 49(12):3320–3325, 2003.

[47] David H Hall and RL Russell. The posterior nervous system of the nematode caenorhabditis elegans: serial reconstruction of identified neurons and complete pattern of synaptic interactions. *The Journal of neuroscience*, 11(1):1–22, 1991.

[48] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[49] Peter D Hoff, Adrian E Raftery, and Mark S Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97 (460):1090–1098, 2002.

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

[51] Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

[52] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[53] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *Machine learning: ECML 2004*, pages 217–226. Springer, 2004.

[54] Stefan Kunis and Holger Rauhut. Random sampling of sparse trigonometric polynomials, ii. orthogonal matching pursuit versus basis pursuit. *Foundations of Computational Mathematics*, 8(6):737–763, 2008.

[55] Dominic S Lee and Carey E Priebe. Bayesian vertex nomination. *arXiv preprint arXiv:1205.5082*, 2012.

[56] Rosalind C Lee, Rhonda L Feinbaum, and Victor Ambros. The c. elegans heterochronic gene lin-4 encodes small rnas with antisense complementarity to lin-14. *cell*, 75(5):843–854, 1993.

[57] Jing Lei, Alessandro Rinaldo, et al. Consistency of spectral clustering in stochastic block models. *The Annals of Statistics*, 43(1):215–237, 2014.

[58] Wai-Hung Liu and Andrew H Sherman. Comparative analysis of the cuthill-mckee and the reverse cuthill-mckee ordering algorithms for sparse matrices. *SIAM Journal on Numerical Analysis*, 13(2):198–213, 1976.

[59] Vince Lyzinski, Daniel L Sussman, Donniell E Fishkind, Henry Pao, Li Chen, Joshua T Vogelstein, Youngser Park, and Carey E Priebe. Spectral clustering for divide-and-conquer graph matching. *IEEE Parallel Computing, Systems and Applications. To appear. arXiv preprint arXiv:1310.1297*, 2013.

[60] Vince Lyzinski, Sancar Adali, Joshua T Vogelstein, Youngser Park, and Carey E Priebe. Seeded graph matching via joint optimization of fidelity and commensurability. *arXiv preprint arXiv:1401.3813*, 2014.

[61] Vince Lyzinski, Donniell Fishkind, Marcelo Fiori, Joshua T Vogelstein, Carey E Priebe, and Guillermo Sapiro. Graph matching: Relax at your own risk. *arXiv preprint arXiv:1405.3133*, 2014.

[62] Vince Lyzinski, Donniell E Fishkind, and Carey E Priebe. Seeded graph matching for correlated Erdos-Renyi graphs. *Journal of Machine Learning Research*, (accepted), 2014.

[63] Vince Lyzinski, Daniel L Sussman, Minh Tang, Avanti Athreya, and Carey E Priebe. Perfect clustering for stochastic blockmodel graphs via adjacency spectral embedding. *Electronic Journal of Statistics*, 8(2):2905–2922, 2014.

[64] David Marchette, Carey Priebe, and Glen Coppersmith. Vertex nomination via attributed random dot product graphs. In *Proceedings of the 57th ISI World Statistics Congress*, volume 1121, page 1126, 2011.

[65] E Maupas. Modes et formes de reproduction des nematodes. *Archives de Zoologie expérimentale et générale*, 8:463–624, 1901.

[66] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[67] Katta G Murty and Santosh N Kabadi. Some NP-complete problems in quadratic and nonlinear programming. *Mathematical programming*, 39(2):117–129, 1987.

[68] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.

[69] Günther Osche. Die bedeutung der osmoregulation und des winkverhaltens für freilebende nematoden. *Zeitschrift für Morphologie und ökologie der Tiere*, 41 (1):54–77, 1952.

[70] Henry Pao. *Graph Inference and Graph Matching*. PhD thesis, Johns Hopkins University, May 2015.

[71] Yagyensh Chandra Pati, Ramin Rezaiifar, and PS Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pages 40–44. IEEE, 1993.

[72] Dragana M Pavlovic, Petra E Vértes, Edward T Bullmore, William R Schafer, and Thomas E Nichols. Stochastic blockmodeling of the modules and core of the caenorhabditis elegans connectome. *PloS one*, 9(7):e97584, 2014.

[73] Carey E Priebe, John M Conroy, David J Marchette, and Youngser Park. Scan statistics on enron graphs. *Computational & Mathematical Organization Theory*, 11(3):229–247, 2005.

[74] Carey E Priebe, David J Marchette, Zhiliang Ma, and Sancar Adali. Manifold matching: Joint optimization of fidelity and commensurability. *Brazilian Journal of Probability and Statistics*, 27(3):377–400, 2013.

[75] Dale Purves, George J Augustine, David Fitzpatrick, William C Hall, Anthony-Samuel LaMantia, James O McNamara, and Leonard E White. Neuroscience. *Sunderland, MA: Sinauer Associates*, 3, 2001.

[76] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[77] Donald L Riddle, Thomas Blumenthal, Barbara J Meyer, James R Priess, et al. Origins of the model. 1997.

[78] Karl Rohe, Sourav Chatterjee, and Bin Yu. Spectral clustering and the high-dimensional stochastic blockmodel. *The Annals of Statistics*, 39(4):1878–1915, 2011.

[79] Ron Rubinstein, Alfred M Bruckstein, and Michael Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057, 2010.

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

[81] Cencheng Shen, Li Chen, and Carey E Priebe. Sparse representation classification beyond l_1 minimization and the subspace assumption. *Submitted to Journal of Machine Learning. arXiv preprint arXiv:1502.01368*, 2015.

[82] James C Spall. Estimation via markov chain monte carlo. *Control Systems, IEEE*, 23(2):34–45, 2003.

[83] John E Sulston, E Schierenberg, John G White, and JN Thomson. The embryonic cell lineage of the nematode¡ i¿ caenorhabditis elegans¡/i¿. *Developmental biology*, 100(1):64–119, 1983.

[84] Daniel L Sussman, Minh Tang, Donniell E Fishkind, and Carey E Priebe. A con-

sistent adjacency spectral embedding for stochastic blockmodel graphs. *Journal of the American Statistical Association*, 107(499):1119–1128, 2012.

[85] Daniel L Sussman, Minh Tang, and Carey E Priebe. Consistent latent position estimation and vertex classification for random dot product graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(1):48–57, 2014.

[86] Minh Tang, Daniel L Sussman, and Carey E Priebe. Universally consistent vertex classification for latent positions graphs. *The Annals of Statistics*, 41(3): 1406–1430, 2013.

[87] Warren S Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, 17(4):401–419, 1952.

[88] Warren S Torgerson. Theory and methods of scaling. 1958.

[89] Emma K Towlson, Petra E Vértes, Sebastian E Ahnert, William R Schafer, and Edward T Bullmore. The rich club of the c. elegans neuronal connectome. *The Journal of Neuroscience*, 33(15):6380–6387, 2013.

[90] Joel A Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

[91] Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.

[92] Shinji Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):695–703, 1988.

[93] Lav R Varshney, Beth L Chen, Eric Paniagua, David H Hall, and Dmitri B Chklovskii. Structural properties of the caenorhabditis elegans neuronal network. *PLoS computational biology*, 7(2):e1001066, 2011.

[94] Joshua T Vogelstein, John M Conroy, Louis J Podrazik, Steven G Kratzer, Eric T Harley, Donniell E Fishkind, R Jacob Vogelstein, and Carey E Priebe. Fast approximate quadratic programming for large (brain) graph matching. *arXiv preprint arXiv:1112.5507*, 2011.

[95] John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2009.

[96] John Wright, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas S Huang, and Shuicheng Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044, 2010.

[97] Zai Yang, Cishen Zhang, Jun Deng, and Wenmiao Lu. Orthonormal expansion l_1-minimization algorithms for compressed sensing. *arXiv preprint arXiv:1108.5037*, 2011.

[98] Stephen J Young and Edward R Scheinerman. Random dot product graph models for social networks. In *Algorithms and models for the web-graph*, pages 138–149. Springer, 2007.

[99] Junying Yuan, Shai Shaham, Stephane Ledoux, Hilary M Ellis, and H Robert Horvitz. The c. elegans cell death gene ced-3 encodes a protein similar to mammalian interleukin-1β-converting enzyme. *Cell*, 75(4):641–652, 1993.

[100] Mikhail Zaslavskiy, Francis Bach, and J-P Vert. A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2227–2242, 2009.

[101] Peng Zhao and Bin Yu. On model selection consistency of lasso. *The Journal of Machine Learning Research*, 7:2541–2563, 2006.

[102] Mu Zhu and Ali Ghodsi. Automatic dimensionality selection from the scree plot via the use of profile likelihood. *Computational Statistics & Data Analysis*, 51(2):918–930, 2006.

Vita

Li Chen received her Bachelor's degree with summa cum laude from the University of Oregon in three years, where she majored in both pure and applied mathematics. She received her first Master's of Science degree at Oregon State University, where she majored in mathematics with a concentration in actuarial mathematics. Her master's thesis focuses on applying stochastic process modeling to risk management for non-profit organizations. Li Chen joined the PhD program in the Department of Applied Mathematics and Statistics at the Johns Hopkins University in 2010. She received her second Master's degree in Engineering in Applied Mathematics and Statistics in 2013. She received her PhD degree in Applied Mathematics and Statistics in 2015. When she is not busy studying or researching, she enjoys hiking, cooking and jogging.