GRAPH ANALYSIS COMBINING NUMERICAL, STATISTICAL, AND STREAMING TECHNIQUES

A Dissertation
Presented to
The Academic Faculty

by

James Paul Fairbanks

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Computational Science and Engineering in the
School of Computational Science and Engineering

Georgia Institute of Technology
May 2016

GRAPH ANALYSIS COMBINING NUMERICAL, STATISTICAL, AND STREAMING TECHNIQUES

ACKNOWLEDGEMENTS

Thanks be to my collaborators and mentors for without their help I would not be here. You have all been great scholars and advisers.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . . **iii**

LIST OF TABLES . . . . **vii**

LIST OF FIGURES . . . . **viii**

SUMMARY . . . . **x**

I INTRODUCTION **1**

II CONTRIBUTIONS **4**

2.1 Statistical analysis of graphs 4

2.2 Numerical methods for graph analysis 4

2.3 Algorithmic improvements 5

III BACKGROUND AND LITERATURE REVIEW **6**

3.1 Graph Theory 6

3.1.1 Social Network Features 6

3.1.2 Spectral Methods 8

3.2 Applications 13

3.2.1 Ranking and importance 15

3.2.2 Dynamic behavior analysis 15

3.2.3 Partitioning and community detection 16

3.3 Linear Algebra 17

3.3.1 Nonnegative Matrix Factorization (NMF) 20

3.3.2 Numerical Methods for Data Mining 22

3.4 Streaming Algorithm concepts 24

IV UNDERSTANDING DYNAMIC GRAPH STREAMS **28**

4.1 Insight from Streaming Features 28

4.1.1 Vertex Features 29

4.1.2 Temporal Analysis 32

4.1.3 Temporal Features 33
4.1.4 Anomaly Detection 37
4.2 Behavioral Clustering in Dynamic Graphs 41
4.2.1 Contributions 42
4.2.2 Foundations 43
4.3 Case Study - CAIDA Dataset 50
4.4 Performance Experiments 54
4.5 Conclusions and Future Work 55

V NUMERICAL METHODS FOR GRAPH PROBLEMS 57

5.1 The Spectral Sweep Cut Algorithm 57
5.2 The Stochastic Block Model 60
5.3 High Quality Partitioning from Low Quality Eigenvectors 64
5.4 Stopping Criteria for Spectral Partitioning 67
5.4.1 Contributions 69
5.5 Eigenvalue accuracy and Cheeger's inequality 70
5.6 Experiments 72
5.6.1 Experimental Design 72
5.6.2 Summary of Results 75
5.6.3 Effect of Eigengap 79
5.6.4 Alternative Stopping Criteria 80
5.6.5 Balance 81
5.7 Conclusions 83

VI SPECTRAL PARTITIONING WITH BLENDS OF EIGENVECTORS 86

6.1 Introduction 87
6.1.1 A Model Problem 90
6.1.2 Contributions 91
6.1.3 Related Work 92

6.2 Blends of Eigenvectors 94

6.2.1 Converging to a Single Eigenspace 94

6.2.2 Converging to a Subspace Spanned by Multiple Eigenspaces . 98

6.3 The Ring of Cliques 102

6.3.1 Definition 103

6.3.2 Eigenpairs of ROC Normalized Adjacency Matrix 104

6.3.3 A residual tolerance for ring of cliques 117

6.3.4 Minimal Perturbation 118

6.3.5 The Power Method 122

6.3.6 Experiment 126

6.4 Conclusions 126

VII CONCLUSION 129

REFERENCES 131

LIST OF TABLES

1 A reference table of graph matrices and vectors. 8

2 As the residual tolerance decreases the distribution of conductance values becomes more concentrated. However the minimum value seen in a sample increases. Each column represents the distribution over 30 samples. 65

3 Shows the size of each graph along with some eigenvalues. Small values of $\lambda_3 - \lambda_2$ indicate difficult problems. Graphs from the Newman and SNAP collections are abbreviated with N and S, respectively. 74

4 Some iterates of the solver are shown in detail for the graph S/web-Google. Because Theorem 2 requires that $\mu - r \geq 0$, ψ is not supplied when this condition is not met. We find that $\mu - r > 0$ is a good proxy for the stopping criterion $\phi \geq \psi$, which implies that one can use this test in order to reduce the number of times that ϕ must be computed. 76

5 Improvement shown for each graph averaged over 10 runs. The columns are the graph name, iterations until $\phi < \psi$, iterations until $r < tol$, ratio of $\phi\left(\mathbf{x}^{I_C}\right)$ to $\phi\left(\mathbf{x}^{I_F}\right)$, the conductance found when $r < tol$, and the residual upon early termination. This table shows the graphs where our convergence criterion was reached within the maximum number of iterations. 78

6 The effect of restricting to balanced cuts is shown. The max, mean, and min are taken over all graphs. ϕ_F^{10} represents the conductance of the best cut of the final computed eigenvector with at least $\frac{n}{10}$ vertices on each side. ϕ_C^{10} represents the conductance of the best balanced cut achieved with our stopping criterion. 83

7 Validation of Theorem 14 by comparing to formulas in Theorem 8. . . 123

8 Validation of Theorem 14 by comparing to vectors provided by `ARPACK`. 124

9 Table corresponding to Figure 23 with 10 iterations. Convergence to the Fiedler eigenpair is slow, yet convergence to the orthogonal complement of $\mathcal{X}_{noise}$ is rapid (column 2). 127

LIST OF FIGURES

1 Our framework combines sparse parallel graph algorithms and dense parallel linear algebra algorithms. 30

2 The distributions of logarithm of betweenness centrality and the derivative of logarithm of betweenness centrality provide information about vertices with rare behavior. 33

3 Tracing values of vertex features over time provides information about vertex behavior and changes in graph structure. 35

4 Counting vertices by sign of their finite differences at each time step. 37

5 Scatter plot matrix showing the outliers (red) and normal data (blue) 39

6 Scatter plot matrix showing the betweenness centrality outliers (red) and normal data (blue) 40

7 The temporal similarity of betweenness centrality structure between all pairs of time-steps, in internet traces data. 52

8 Longitudinal plot of a sample of vertices from each cluster identified by NMF on the betweenness centrality derived feature matrix. Each subplot represents a group of vertices that peak at the same time. The logarithm of betweenness centrality is shown on the ordinate. 53

9 NMF—BPP Scalability experiments on a 2840256×103 dense matrix 55

10 A stochastic block model with 12 blocks each of size 25 graph drawn to show structure (left). A sparsity plot of the same graph (right). . . 62

11 A histogram of the top 37 eigenvectors of the graph in Figure 10. The eigenvectors close to 1 correspond to vectors which reveal low conductance partitions. 63

12 The top eigenvectors of the graph in Figure 10 show concentration for vertices in the same block. This concentration implies that the block identities can be recovered from the spectral coordinates. 64

13 Tighter eigenresidual bounds imply tighter distributions of approximate solutions. Log scale implies that the distributions on the left have smaller variances. 66

14 The distribution of iteration reduction over the experiments. 77

15 The distribution of conductance increases over the experiments. . . . 79

16 Results for individual runs on each graph, not averaged, are plotted. The iteration savings $\frac{I_F}{I_C}$, shown on top, increase as the gap decreases and the loss in conductance $\frac{\phi_C}{\phi_F}$, shown on bottom, does not increase as the gap decreases. 81

17 The decrease in conductance and residual for SNAP/web-Notre-Dame is shown. The flat regions of the conductance curve indicate where one would stop based on $\phi(\mathbf{x}^i) - \phi(\mathbf{x}^{i+1}) = 0$. This shows that neither stopping at I_N iterations nor when the objective function stops decreasing achieves low conductance partitions for this problem, even though both satisfy $\phi(\mathbf{x}) < \sqrt{2\lambda_2}$. Using our criterion and stopping at I_C iterations does achieve a low conductance cut. 82

19 Distribution of eigenvalues of $\hat{L}$ for $\mathcal{R}_{b=8,q=32}$. The gray bars represent a histogram of the eigenvalues, including multiplicities. The red diamond represents a large multiplicity of $(n-2q)$ at $-(b-1)^{-1}$ corresponding to $\mathcal{X}_{noise}$ (see Theorem 7). There is a green interval near 1 containing the portion of the spectrum given by $\lambda_1^{(k)}$ and a blue interval near 0 containing $\lambda_1^{(k)}$ for $k = 0, 1, \dots \lfloor q/2 \rfloor$ (see Theorem 8). The open left endpoint of each interval signifies that the eigenvalue corresponding to $\lambda^{(k)}_{\lceil q/2 \rceil}$ is not present when q is odd. Small eigenvalues of the Laplacian matrix correspond to large eigenvalues of the Adjacency matrix. . . . 104

20 The eigenvectors of $\hat{A}$ are shown for $\mathcal{R}_{16,10}$. The eigenvectors with eigenvalues close to 1 (left) indicate the block structure with differing frequencies. The eigenvectors close to -1 (right) assign opposite signs to the internal vertices and corner vertices of each block. 111

21 Asymptotic estimates of spectral gaps (left) and Fiedler eigenvalues (right) for rings of cliques with parameters $b = 10, 100, 1000$ and $q = 2, 4, ..., 8096$. Lines represent leading-order terms derived in Theorems 9 and 10 and '+' represent actual eigenvalues as given by the formulas in Theorem 8. 114

22 Empirical measurements of minimal error perturbations on a log-log scale. Lower bounds are shown in the same color with dashed lines. . 120

23 First several iterations of the power method applied to $\mathcal{R}_{b=20,q=30}$. Above: sweep conductance of A reordered by sorting the 1^{st} (top-left), 2^{nd} (top middle), and 3^{rd} iterations (top-right). Horizontal axis represents which vertex to split at under the induced ordering; vertical axis is the conductance for each split on a log scale. Below: matrix sparsity plots of A reordered by sorting the 1^{st} (bottom-left), 2^{nd} (bottom-middle), and 3^{rd} iterations (bottom-right). Red lines demonstrate which edges are cut for the optimal cut in each ordering. . . . 127

Graph analysis combining numerical, statistical, and streaming techniques

James Paul Fairbanks

138 Pages

Directed by Professor David A. Bader

Graph analysis uses graph data collected on a physical, biological, or social phenomena to shed light on the underlying dynamics and behavior of the agents in that system. Many fields contribute to this topic including graph theory, algorithms, statistics, machine learning, and linear algebra.

This dissertation advances a novel framework for dynamic graph analysis that combines numerical, statistical, and streaming algorithms to provide deep understanding into evolving networks. For example, one can be interested in the changing influence structure over time. These disparate techniques each contribute a fragment to understanding the graph; however, their combination allows us to understand dynamic behavior and graph structure.

Spectral partitioning methods rely on eigenvectors for solving data analysis problems such as clustering. Eigenvectors of large sparse systems must be approximated with iterative methods. This dissertation analyzes how data analysis accuracy depends on the numerical accuracy of the eigensolver. This leads to new bounds on the residual tolerance necessary to guarantee correct partitioning. We present a novel stopping criterion for spectral partitioning guaranteed to satisfy the Cheeger inequality along with an empirical study of the performance on real world networks.

SUMMARY

Graph analysis uses graph data collected on a physical, biological, or social phenomena to shed light on the underlying dynamics and behavior of the agents in that system. Many fields contribute to this topic including graph theory, algorithms, statistics, machine learning, and linear algebra.

This dissertation advances a novel framework for dynamic graph analysis that combines numerical, statistical, and streaming algorithms to provide deep understanding into evolving networks. For example, one can be interested in the changing influence structure over time. These disparate techniques each contribute a fragment to understanding the graph; however, their combination allows us to understand dynamic behavior and graph structure. Based on the desire to use spectral features such as Pagerank, Katz Centrality, commute time, and spectral partitioning for dynamic graph analysis, this dissertation studies the utility of approximate eigenvectors for solving graph analysis problems. This leads to a detailed analysis of spectral partitioning of graphs using approximate eigenvectors.

Spectral partitioning methods rely on eigenvectors for solving data analysis problems such as clustering. Eigenvectors of large sparse systems must be approximated with iterative methods. This dissertation analyzes how data analysis accuracy depends on the numerical accuracy of the eigensolver. This leads to new bounds on the residual tolerance necessary to guarantee correct partitioning. We present a novel stopping criterion for spectral partitioning guaranteed to satisfy the Cheeger inequality along with an empirical study of the performance on real world networks.

The analysis of spectral partitioning on a model problem that leads to a constructive proof of error and residual bounds for finding the optimal partitions on the model problem. This proves that a residual tolerance smaller than $\mathcal{O}\left(n^{-5/2}\right)$ is necessary for finding the Fiedler partitions for general graphs. For this class of graphs a residual tolerance smaller than $\mathcal{O}\left(n^{-1}\right)$ is sufficient for finding the natural clusters in the graph. This understanding of the relationship between numerical errors and graph partitioning leads to a novel stopping criterion for spectral partitioning, which is validated on real-world networks.

The method of analyzing spectral partitioning presented here can be applied to more general graph analysis and data analysis problems. This work leads to improvements in algorithms for analyzing dynamic graphs as well as a deeper understanding of the impact of numerical approximation in spectral partitioning.

CHAPTER I

INTRODUCTION

This dissertation advances a novel framework for dynamic graph analysis that combines numerical, statistical, and streaming algorithms to provide deep understanding into evolving networks as well as a thorough analysis of the accuracy requirements of graph partitioning with spectral methods. Graph data is information regarding entities (vertices) and the connections between them (edges). Research in graph analysis uses graph data collected on physical, biological, or social phenomena to understand the dynamics and behavior of the agents in that system. For example, understanding which vertices are influential and changes in the influence structure over time informs design of message distribution networks. Finding subsets of vertices that are more densely connected to each other than to the rest of the graph is another problem of interest. These subsets can represent communities of users with a common interest or well connected regions of a road network such as cities. Novel graph analyses performed in this dissertation heavily utilize graph algorithms, matrix analysis, and statistics. These disparate techniques each contribute a fragment to our understanding of graphs; however, their combination constructs systems for understanding dynamic behavior and graph structure.

The methods featured in this dissertation are similar in their treatment of the vertices of a graph. Each method represents the vertices of a graph as points defined by some operation accounting for the topological information in the graph. Then, a further processing step uses these points to solve the domain problem. In some applications, the operation is to measure graph theoretic features of the vertices and then perform inference on this feature representation to classify or cluster the vertices.

For community detection and data clustering problems, the first embed the graph using spectral coordinates, and then infer cluster structure among the embedded positions.

Chapter 4 demonstrates the advantages of using this feature extraction methodology for understanding dynamic graphs. Dynamic graphs push the limits of our ability to represent and display information. The complex structure of real world networks, such as social networks, makes a two dimensional embedding of the graph difficult to construct and limited in utility. In addition to the complex structure at each point in time, a dynamic graph is undergoing evolution during analysis. This presents a real challenge for understanding the relationships between the actors in the network — between each other and over time. Feature embeddings followed by statistical and machine learning techniques provide insight into these complex temporal relationships.

Community detection and graph partitioning lead to difficult optimization problems on the graph. One approach to solving these problems is to find spectral coordinates of the vertices as defined by the graph Laplacian then partition the vertex set into groups by optimizing an objective function based on these coordinates. A significant hurdle to using this approach at the scale of large social networks is the ill-conditioning of the corresponding eigenvalue problems. It is hard to solve the eigenvalue problem accurately because these graphs lead to Laplacian matrices with eigenvalues close together.

Previous work on real world networks has shown that the computed eigenvectors are sensitive to computational error. In order to reliably partition networks, one must address the sensitivity of the problem. The key insight is that the eigenvectors are a means of producing spectral coordinates while the primary goal is to find a good partition of the graph. Unlike prior work, which computes high accuracy approximations to the eigenvectors, this dissertation directly addresses the issue of numerical accuracy and derives error bounds that ensure correct recovery of cluster structure.

Chapter 5 addresses the accuracy requirements for real world networks by developing a condition on the approximate eigenvectors providing the same theoretical guarantee as the exact eigenvectors, which obtains good performance on real world networks. Chapter 6 examines the sensitivity of computing invariant subspaces instead of eigenvectors, and applies this analysis to a model problem. The model problem determines upper bounds on the sufficient eigenvector accuracy for solving the spectral partitioning problem. The model problem approach allows the construction of useful theory for understanding the difficulty of the spectral partitioning problem. The approaches in Chapter 5 and Chapter 6 complement each other because the error bounds from Chapter 6 require a priori estimates of the spectrum.

A consistent theme of this work is using a graph processing system to generate an embedding of the vertices and then answers a graph analysis question with the best available technique applied to the embedding. This unified view of graph analysis methods allows one to combine structural graph techniques, spectral graph techniques, and statistical and machine learning techniques to solve the toughest problems in graph analysis. This dissertation provides results on this methodology and guidance for applications of these techniques by combining theoretical analysis of simple problems with experimental evaluations on real world problems.

CHAPTER II

CONTRIBUTIONS

Chapter 4 contributes to the dynamic outlier detection, dynamic behavioral clustering, and graph partitioning problems by combining streaming feature extraction with methods of statistical inference. A deeper understanding of the relationship between numerical accuracy and data analysis accuracy in graph analysis applications is provided. Specifically, Chapter 5 studies the practical aspects of this relationship, and Chapter 5 studies spectral partitioning using blends of eigenvectors.

2.1 Statistical analysis of graphs

Chapter 4 presents an empirical examination of vertex features as random variables. The conclusion of this study is that even when the stream of incoming temporal edges is not amenable to modeling using standard parametric distributions with well understood properties and estimators the induced vertex features approximately follow such distributions. This finding is important for applying the statistical techniques to dynamic graphs. Application of this framework to real world dynamic graphs contributes to dynamic outlier detection and dynamic behavioral clustering.

2.2 Numerical methods for graph analysis

Chapter 5 and Chapter 6 study the relationship between numerical accuracy and graph analysis objectives. Chapter 5 studies the dependence of partitioning quality on the numerical accuracy leading to the first meaningful stopping criterion for spectral partitioning. Section 5.4 introduces a novel parameter-free stopping criterion for iterative methods for spectral partitioning. This criterion guarantees, under the standard eigensolver assumptions, that the output partition satisfies Cheeger's bound on the conductance of the graph. This criterion allows one to find partitions without specifying a residual or error tolerance, which is important for clustering a graph with unknown spectrum. Experiments in Section 5.6 show good performance on real world networks.

Chapter 6 provides new results connecting partitioning graphs with linear combinations of low energy Laplacian eigenvectors. In general, when blends of eigenvectors can solve a data analysis problem, we provide pointwise convergence guarantees. Section 6.3 demonstrates this theory on an accessible model problem, the Ring of Cliques, by showing that the top eigenvectors recover the correct partition. This model problem is thoroughly analyzed including a derivation of the normalized adjacency eigenvalues and eigenvectors. Analysis of the structure of this graph along with the properties of the eigenvalues provides bounds on the necessary and sufficient residual tolerance for correctly recovering the cluster structure. These results combine to bridge the gap between linear algebra based data analysis and convergence theory of iterative approximation methods.

2.3 Algorithmic improvements

This work leads to improvements in practical algorithms for graph analysis. By applying ideas and techniques from various disciplines one is able to solve problems in new ways.

1. Using dense linear algebra on the output of sparse graph kernels leads to better scalability and analysis.

2. Analysis of the interaction between iterations of a numerical solver and the final objective yields improved stopping criteria.

3. Using approximate eigensolvers for partitioning with blends allows faster solvers.

CHAPTER III

BACKGROUND AND LITERATURE REVIEW

This section introduces many of the necessary concepts and surveys the state of the art. The important fields to cover are graph theory (Section 3.1), (numerical) linear algebra (Section 3.3), and streaming algorithms (Section 3.4). Section 3.2 describes motivating applications.

3.1 Graph Theory

Let $G = (V, E)$ be a graph. The vertex set is $V = \{1, 2 \ldots, n\}$ and the edge set $E \subset V \times V$. Graphs can be used to model many problems in science and engineering. This work treats the vertices as the agents and the edges as representing the connections between them. In some problems the edges can have a weight w_{ij} storing the strength of the connection. Dynamic graphs are represented as an ordered stream $e_t = (i_t, j_t)$, possibly with weights. Edge streams arise when the process creating the edges occurs over time. At any point in time $t \in \{1, 2 \ldots\}$, an algorithm can only access the edges that happened prior to t that is, $G_t = (V, E_t)$ where $E_t = \{e_1, e_2 \ldots, e_t\}$. Dynamic processes are prolific in modern data driven environments where technology allows the creation of data quickly and demands answers just as quickly.

3.1.1 Social Network Features

In the social network analysis literature, there is a long tradition of constructing functions from vertices to numbers. These functions, called vertex features here, are used to study the effect of graph structure on human behavior. A diverse literature surrounds the features, and each feature attempts to capture some aspect of an intuitive social property.

Centrality metrics on static graphs provide an algorithmic way to measure the relative importance of a vertex with respect to information flow through the graph. Higher centrality values generally indicate greater importance or influence. Betweenness centrality [31] is a specific metric based on the fraction of shortest paths on which each vertex lies. The definition is shown in Equation (1) using σ_{uw} to denote the number of shortest paths from u to w, and σ_{uvw} for the number of those paths that pass through v.

$$bc(v) = \sum_{u,w} \frac{\sigma_{uvw}}{\sigma_{uv}} \tag{1}$$

The computational cost of calculating exact betweenness centrality can be prohibitive due to the $\mathcal{O}\left(n^3\right)$ complexity of all pairs shortest paths; however, approximating by sampling betweenness centrality is tractable and produces accurate values for the highest centrality vertices [23, 3]. According to observations of scale free networks, betweenness centrality follows a heavy tail distribution. Chapter 4 examines the logarithm of the betweenness centrality score for each vertex to account for the heavy tailed distribution.

The *local clustering coefficient* of a vertex v is the number of triangles centered at v divided by the number of wedges centered at v as seen in Equation (2) [96]. The number of triangles centered at v is $triangles(v) = |\{u \sim v \sim w \sim u \mid u, v, w \in V\}|$

$$cc(v) = \frac{triangles(v)}{degree(v)(degree(v) - 1)} \tag{2}$$

The clustering coefficient is a measure of how tightly knit the vertices are in the graph.

Previous research uses a feature extraction and then feature analysis framework to study large static graphs using vertex features including Oddball [2], which performs graph anomaly detection using local (egonet) features, and RolX [40], which uses Nonnegative Matrix Factorization (NMF) on locally extracted features. This work is the first to use both global features and dynamic information. Here we are learning

Table 1: A reference table of graph matrices and vectors.

Term	Definition
Adjacency Matrix	$A_{ij} = \begin{cases} 1, & \text{if } v_i \sim v_j \\ 0, & \text{otherwise} \end{cases}.$
Degrees	$D_{ii} = \deg(v_i)$ all other entries 0.
Combinatorial Laplacian	$L = D - A$
Normalized Laplacian	$\hat{L} = I - D^{-1/2}AD^{-1/2}$
Normalized Adjacency Matrix	$\hat{A} = D^{-1/2}AD^{-1/2}$
Stochastic Propagator	$P = AD^{-1}$
Random Walk with Restart	$\alpha P + (1-\alpha)ev^T$
Perron Vector	$\hat{A}x = x$
Normalized Laplacian Kernel	$(I - \hat{A})x = 0x$
Stationary Distribution	$Py = 1y$
Eigenvalue λ, Eigenvector x	$Ax = \lambda x$
Generalized Eigenvalue Problem	$L\mathbf{x} = \lambda D\mathbf{x}$

the roles of the vertices, and the structure of those roles over time as the graph changes.

3.1.2 Spectral Methods

Spectral graph theory is the study of graphs through the techniques of matrix analysis and linear algebra. Many important theoretical results are derived this way including the Matrix Tree Theorem, and the Perron-Frobenius theorem, and also practical efficient algorithms [11, 89]. The relevant algorithms derived from graph spectra are for partitioning and ranking. Table 1 provides the definitions and connections between various graph matrices and vectors for reference.

Let A denote the adjacency matrix of an undirected graph with entries $a_{i,j}$ equal to 1 if vertex i is adjacent to vertex j. Use I to represent the identity matrix and $\mathbf{1}$ to represent the vector of ones. If D is the diagonal matrix whose entries are the degrees

$d_i = \sum_j a_{i,j} = A\mathbf{1}$, then $L = D - A$ is the combinatorial Laplacian. The combinatorial Laplacian satisfies the equation $L\mathbf{1} = 0$, which can be seen by $(L\mathbf{x})_i = d_i \sum_{j\sim i} a_{ij} = 0$

Let S be a subset of the vertex set and use $\bar{S} = V \setminus S$ to denote set complement. The sets $S, \bar{S}$ represent a cut of the graph. The edges between these sets is the edge expansion of S denoted $E(S, \bar{S})$. Graph partitioning applications focus on identifying sets S that minimize $|E(S, \bar{S})|$ relative to some normalization. The normalization is important because choosing a small set S will lead to a small edge expansion but applications require some sense of balance between the two sets.

This dissertation measures the quality of a (vertex) cut of the graph using conductance (Equation (3)), which is the surface area to volume ratio of a subset of vertices in the graph. Define $\mathrm{vol}(S) = \sum_{i,j\in S} a_{i,j}$ as the total weight of the edges within S. The conductance of a cut S is thus given by the formula [11]:

$$\phi(S) = \frac{|E(S, \bar{S})|}{\min(\mathrm{vol}(S), \mathrm{vol}(\bar{S}))} \tag{3}$$

The conductance of the graph, $\phi_G = \min_S \phi(S)$, is the minimum over all cuts S of $\phi(S)$. There are other possible objective functions such as normalized edge cut defined in Equation (4).

$$\frac{E(S, \bar{S})}{\min(|S|, |\bar{S}|)} \tag{4}$$

Normalized cut divides by the number of vertices in the smaller of the two sets. The conductance normalization is used because of its fundamental connections to Markov random walks on the graph [67].

The conductance objective is a quadratic functional $\phi : \mathbb{R}^n \to \mathbb{R}$ with a linear algebra interpretation. A set S is encoded as a vector $\mathbf{b}$ using the definition $b_i = 1$ if vertex i is in S and $b_i = 0$ otherwise. This gives a bijection between sets S and vectors $\mathbf{b} \in \{0, 1\}^n$. Using this convention we derive the edge cut $|E(S, \bar{S})| = \mathbf{b}^T L\mathbf{b}$, and the $\mathrm{vol}(S) = \mathbf{b}^T D\mathbf{b}$. This formulation expresses the objective function as a quadratic form

$$\phi(\mathbf{b}) = \frac{\mathbf{b}^T L\mathbf{b}}{\mathbf{b}^T D\mathbf{b}}, \tag{5}$$

where $\mathbf{b}^T L\mathbf{b} = \sum_{i\sim j}(b_i - b_j)^2$ and $\mathbf{b}^T D\mathbf{b} = \sum_i D_{ii}b_i^2 = \sum_{i\in S} deg(v_i)$.

By relaxing the constraint of integer assignment we minimize over vectors $x \in \mathbb{R}^n$ as in Equation (6). The constraint that $x \perp 1$ follows from the fact that $\mathbf{1}^T L\mathbf{1} = 0$.

$$\min_{\mathbf{x}\perp 1} \frac{\mathbf{b}^T L\mathbf{b}}{\mathbf{b}^T D\mathbf{b}} \tag{6}$$

The Fischer-Courant theorem implies that the minimum value is the smallest nonzero generalized eigenvalue, and the minimizer in a connected graph is the second smallest eigenvector, also called the Fiedler vector [11]. This leads to the generalized eigenvalue problem

$$L\mathbf{x} = \lambda D\mathbf{x} \tag{7}$$

A change of variables allows one to apply theorems and computational tools for symmetric eigenvector problems to this generalized eigenvalue problem. A pair $\lambda, \mathbf{y}$ solves the generalized eigenequation $L\mathbf{y} = \lambda D\mathbf{y}$, if and only if the pair $\lambda, \mathbf{x} = D^{-\frac{1}{2}}\mathbf{y}$ solves Equation (8)

$$\hat{L}\mathbf{x} = \lambda\mathbf{x}. \tag{8}$$

Also, the vector $D^{\frac{1}{2}}\mathbf{1}$ lies in the kernel of $\hat{L}$ since $\hat{L}D^{\frac{1}{2}}\mathbf{1} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}}D^{\frac{1}{2}}\mathbf{1} = L\mathbf{1} = 0$. Another reason to use this normalization is that for graphs with skewed degree distributions the high degree vertices can dominate the spectrum of A [68]. The normalized Laplacian can recover the skewed degree planted partition model [17].

This dissertation focuses on the *normalized adjacency* matrix $\hat{A} = D^{-1/2}AD^{-1/2}$ and the *normalized Laplacian* $\hat{L} = I - \hat{A}$. *Eigenvalues* and *eigenvectors* are defined as the solutions to the equation $A\mathbf{x} = \lambda\mathbf{x}$. The vectors $\mathbf{x}$ are the eigenvectors and the scalar values λ are the eigenvalues. The eigenvalues and eigenvectors are intrinsic quantities of the matrix A. For symmetric real matrices, the eigenvalues are always real and there exists an *orthonormal basis* $\{\mathbf{q}_i \mid i \in 1 \ldots n\}$ to represent the eigenvectors. The vectors $\mathbf{q}_i, \mathbf{q}_j$ are orthogonal $\mathbf{q}_i^T \mathbf{q}_j = 0$ when $i \neq j$ and unit norm $\|\mathbf{q}_i\|_2 = 1$. Each $\mathbf{q}_i$ corresponds to an eigenvalue λ_i. When eigenvalues are repeated this basis is not unique and algorithms must choose a basis for representation.

When the matrix is a graph Laplacian $\hat{L}$, all eigenvalues are nonnegative. When the graph is connected, $\hat{L}$ has a unique 0 eigenvalue and we sort the eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \lambda_n$ and label the associated eigenvectors $\mathbf{q}_1, \mathbf{q}_2 \ldots \mathbf{q}_n$. Undirected graphs have symmetric Laplacian matrices which implies the existence of an orthogonal basis for the eigenvectors. In terms of matrix decomposition $\hat{L} = Q \Lambda Q^T$, where $QQ^T = I$ and Λ is a diagonal matrix with nonnegative entries. The properties of the Laplacian eigenvalues and eigenvectors determine whether spectral partitioning will find good cuts and how difficult said cuts are to find.

Spectral partitioning algorithms use eigenvectors of the normalized Laplacian to embed the vertices of G into a low dimensional space. An algorithm for partitioning vector space data is then applied to the embedded vertices. The simplest such algorithm is to embed into one dimension and use a threshold cut. However, this does not guarantee the minimum value of the objective is satisfied. One approach to partitioning with one dimensional embeddings is to take an optimal sweep cut.

For any vector x let $S_x(t) = \{i \mid x_i > t\}$ be the sweep cut at threshold t. For every cut S the vector $\mathbf{x} = e_S - e_{\bar{S}}$ expresses S as a sweep cut. We call the conductance of a vector the conductance of the minimal sweep cut of that vector: $\phi(\mathbf{x}) = \min_t \phi\left(S_x^T\right)$. The optimal sweep cut of a vector can be found with one pass over the edges of the graph as in Algorithm 3. We represent the graph partitioning problem as minimizing $\phi(\mathbf{x})$ over all $\mathbf{x}$. This work focuses on the spectral partitioning algorithms that use a sweep cut of the eigenvector $\mathbf{q}_2$ of $\hat{L}$ to partition the graph.

One reason to use conductance to measure success when applying spectral methods is bounds on the optimal conductance derived from the eigenvalues of the graph.

Cheeger's Inequality (Theorem 3) bounds the relaxation error in terms of the exact eigenvalue λ_2. Theorem 3 guarantees that exact eigenvectors provide a sweep cut with conductance less than $\sqrt{2\lambda_2}$.

Theorem 1. *[11] Let $\hat{L}$ be the normalized Laplacian of a connected graph with degree matrix D. If $\hat{L}\mathbf{x} = \lambda_2\mathbf{x}$ then $\mathbf{y} = D^{-\frac{1}{2}}\mathbf{x}$ has a sweep cut S such that $\phi(S) = \phi\left(D^{-\frac{1}{2}}\mathbf{x}\right) \leq \sqrt{2\lambda_2}$.*

Theorem 3 guarantees that small eigenvectors of the graph Laplacian reveal low conductance cuts of the graph. Section 5.5 examines the effects of approximation of the eigenvector on the guarantee provided by Cheeger's inequality.

Since general eigenvalue problems cannot be solved exactly, we must use approximations. Section 3.3 and Chapter 5 study the impact of numerical error in eigenvectors when applying spectral partitioning. For a recent survey of the graph partitioning problem see [8]. A tutorial on spectral clustering for data mining can be found in [93].

Pagerank and the personalized version is traditionally defined in terms of computing the stationary distribution of a random walk on the graph. The pagerank problem can be solved as a linear system [19]. The intuition behind this comes from viewing the power method as summing powers of the adjacency matrix and the summation formula $\sum_{i=1}^{\infty} x^k = {}^1/_{1-x}$. Solution of Equation (9) gives the desired personalized page rank vector.

$$(I - \alpha P - \alpha \mathbf{v}\mathbf{d}^T)\mathbf{z} = (1-\alpha)\mathbf{v} \tag{9}$$

By application of the Sherman-Morrison formula the equation can be reduced to Equation (10).

$$(I - \alpha P^T)\mathbf{z} = \mathbf{v}, \bar{\mathbf{z}} = \frac{\mathbf{z}}{\|\mathbf{x}\|_1} \tag{10}$$

and 1-normalizing the result. Spectral partitioning and Pagerank calculation, heavily utilize numerical linear algebra to achieve graph analysis goals. The error analysis and general method in Chapter 6 could be applied to Pagerank in order to study how numerical accuracy affects the ranking given by a Pagerank solver. Section 3.3 reviews some concepts.

3.2 Applications

Large graphs are found in many domains including the analysis of social networks, document collections, and transportation networks. Many applications of interest depend on deriving insight into the behavior of individuals within the network or the large scale structure that defines the network. Applications aimed at understanding vertex behavior focus on computing properties such as clustering coefficients and rankings such as Pagerank and betweenness centrality. One method for understanding the large scale structure of the network is to decompose the vertex set into a partition whose parts are called communities. The driving assumption of this work is that vertices that are more connected to each other than to the rest of the graph are behaving in a subsystem. Identifying these subsystems will enable one to better understand the individuals participating in the network. Finding good partitions of these graphs is a challenging data analysis task.

The primary focus of this dissertation is on the analysis of social networks and information networks. The growth of internet technology has vastly expanded the ability of researchers and organizations to acquire information on the interactions between users and machines.

Social networks such as Twitter and Facebook represent a large portion of information transfer on the Internet today. Each new post provides a small amount of new information about the dynamics and structure of the network of human interaction. New posts reveal connections between entities and possibly new social circles or topics of discussion. Social media is a large and dynamic service; at its peak, Twitter recorded over 13,000 Tweets per second [91] and revealed that the service receives over 400 million Tweets per day on average [92]. Social media events — such as two users exchanging private messages, one user broadcasting a message to many others, or users forming and breaking interpersonal connections — can be represented as a graph in which people are vertices and edges connect two people representing the event between them. The nature of the edge can vary depending on application, but one approach for Twitter uses edges to connect people in which one person "mentions" the other in a post. The edge is marked with a timestamp that represents the time at which the post occurred. The format of a Twitter post makes this information accessible. Because of the nature of Twitter, we do not use deletions, and the edge weights count the number of times that one user has mentioned the other.

Previous research shows that Twitter posts reflect valuable information about the real world. Human events, such as breaking stories, pandemics, and crises, affect worldwide information flow on Twitter. Trending topics and sentiment analysis can yield valuable insight into the global heartbeat.

A number of crises and large-scale events have been extensively studied through the observation of Tweets. Hashtags, which are user-created metadata embedded in a Tweet, have been studied from the perspective of topics and sentiment. Hashtag half-life was determined to be typically less than 24 hours during the London riots of 2011 [33]. The analysis of Twitter behavior following a 2010 earthquake in Chile revealed differing propagation of rumors and news stories [66]. Researchers in Japan used Twitter to detect earthquakes with high probability [80]. Analysis of Twitter data can track the prevalence of influenza on a regional level in real time [82]. Betweenness centrality analysis applied to the H1N1 outbreak and historic Atlanta flooding in 2009 revealed highly influential tweeters in addition to commercial and government media outlets [23].

Understanding the internet as a graph provides information about major geopolitical events. Dainotti et al. [16] conducts a thorough analysis of the internet blackout caused by the Egyptian government during the Arab Spring revolution. Understanding patterns of internet connections yields insight into the methods of agents acting to restrict the flow of information. IP network data collected by the Center for Applied Internet Data Analysis (CAIDA) with the UCSD Network Telescope reveals the progress of a botnet composed of 3 million unique IP addresses, in scanning IP space [15]. Understanding how the botnet traversed the internet allows constructs inferences into the logic of the malware that produced the botnet. An analysis of IP network data and graph metrics is presented by Henderson et al. [39] focused on the relationships between different metrics. These applications connect this research to large scale and important human endeavors such as social networks, natural disasters, internet censorship, and cyber-security.

3.2.1 Ranking and importance

Many attempts at quantifying influence have been made. Indegree, retweets, and mentions are first-order measures, but popular users with high indegree do not necessarily generate retweets or mentions [9]. These first-order metrics are traditional database queries that do not take into account topological information. Pagerank and a low effective diameter reveal that retweets diffuse quickly in the network and reach many users in a small number of hops [56]. Users tweeting URLs that are judged to elicit positive feelings are more likely to spread in the network, although predictions of which URL will lead to increased diffusion are unreliable [4].

3.2.2 Dynamic behavior analysis

Dynamic behavior analysis [77] involves understanding the state of vertices in a graph over time. A set of functions define the state such as the number of neighbors, number of participating triangles, Pagerank, or centrality of a vertex. The behavior of the vertex is defined as the temporal pattern of these functions for that vertex. In [26], we found that nonnegative matrix factorization can be used to find clusters of vertices with the same behavior. The goal of dynamic behavior analysis is to understand how the activity of individual vertices evolves over time in contrast to work studying the evolution of the entire network [59].

3.2.3 Partitioning and community detection

Partitioning graphs is useful improving computational performance, distributed computing, understanding communities in social networks, and identifying subsystems of biological systems. The performance of Dijkstra's algorithm for finding shortest paths in graphs can be improved by first precomputing a partition of the graph and then applying a modified Dijkstra's algorithm that performs fewer edge traversals. This method shows large speedup for real world road networks [69]. METIS is used to partition matrices for efficient distributed computing [49].

Detecting communities leads to insight into the behavior of subgroups of agents in the network. For example detecting communities in the social networks of bottle-nose dolphins leads to an understanding of the social behavior of dolphin pods [63, 62]. Metabolic networks contain functional units that can be found with community detection [38]. Community detection can produce hierarchies of communities which provide more information than a single partition [70]. Distributed computation requires that the problem to be solve is decomposed into independent pieces of work that can be performed independently. These diverse applications lead to a variety of methods, objectives, and software.

Statistical physics champions the modularity maximization method which uses a functional on partitions that measures the additional in intracluster density and intercluster sparsity relative to a background null model of independence. Because finding an optimal modularity partition is NP-Hard, the modularity maximization approach uses heuristics to find the partition which maximize the modularity functional. Modularity maximization has been thoroughly studied including a fundamental resolution limit [30], and practical approximation algorithms [6].

The Markov chain and random walk community prefers to optimize measures such as conductance, because of its deep connection to mixing times of Markov chains [45, 74, 67].

Section 5.2 shows the application of spectral partitioning to a class of graphs with known community structure called the stochastic block model. It is known that the eigenvectors of the adjacency matrix can recover the labeling of a graph drawn from the stochastic block model [65]. This research focuses on recovering the hidden assignment of the vertices into blocks. Some approaches to graph partitioning involve modeling the edge set directly and designing estimators directly for these model parameters [28, 95, 77]. The statistical hypothesis testing field defines a hypothesis test for detecting localized increases in activity within a temporal graph [95]. This work emphasizes a particular generative model of the data and leads to a streaming computation of the test statistic. Groups of vertices with communication density much higher than a typical group are considered significant according to this test. Community detection and graph partitioning appear in a broad class of applications where the data heterogeneity in the form of substructures.

3.3 Linear Algebra

Matrix factorizations are useful for many tasks and provide a unified view of many matrix analysis algorithms [34]. For example, Gaussian Elimination for solving linear systems builds a factorization $A = LU$ revealing triangular structure for solving linear equations quickly. Such algorithms spend most of their time constructing the factorization and then the problem is solved by leveraging the factorization efficiently. The eigenvalue decomposition and singular value decomposition reveal the spectrum of the matrix and allow for the rapid computation of matrix functions including analytic functions such as the matrix exponential. Operations such as vertex ranking or spectral partitioning only need part of the appropriate factorization. For example, Pagerank only requires computing the first eigenvector of the random walk with restart matrix. Spectral partitioning uses some number of the minimal eigenvectors of the graph Laplacian. Low Rank approximation is a general method of computing a factored form of a matrix which is close in the appropriate norm to the original data matrix. These ideas and techniques have found success in many areas of data analysis [5, 34, 72]. Matrix analysis algorithms appear throughout this dissertation. In Section 4.2 the nonnegative matrix factorization is used to determine groups of vertices which have similar temporal behavior. Chapter 5 uses eigenvectors of the Laplacian to find low conductance partitions of graphs and derive approximation bounds necessary to recover cluster structure.

Direct methods are used for both dense and sparse problems. The performance on sparse problems depends heavily on the amount of fill-in that occurs. They typically follow a fixed pattern of operations and produce an answer after a fixed number of steps. For sparse problems that are highly structured such as banded problems sparsity aware direct methods are sufficient [35]. For sparse problems without regular structure, such as those arising from graph analysis, direct methods have too much fill-in for practical application to large problems.

For large sparse problems, iterative methods are the only scalable solution technique. Iterative methods take an approximate solution to the problem and iteratively improve it until reaching the desired level of accuracy. The number of iterations depends strongly on the characteristics of the problem. We focus on these methods because they are more widely applicable to the sparse systems needed for graph analysis. Access to a matrix vector multiplication routine is usually sufficient to apply iterative methods.

Throughout the text vector norms are used to measure distance. We use $\|\mathbf{x}\| = \|\mathbf{x}\|_2 = (\sum_i x_i^2)^{-\frac{1}{2}}$, $\|\mathbf{x}\|_\infty = \max_i x_i$, and $\|\mathbf{x}\|_1 = \sum_i |x_i|$. The norm of a matrix A

is either the operator norm $\|A\| = \max_{\|\mathbf{x}\|_2} \mathbf{x}^T A\mathbf{x}$ or the Frobenius norm $\|A\|_F = \sum_{i,j} a_{i,j}$. The distance between two vectors $\mathbf{x}, \mathbf{y}$ is denoted $\|\mathbf{x} - \mathbf{y}\|$.

For example, Algorithm 1 depicts the power method which is an algorithm for computing extremal eigenvectors of matrices. Convergence of the power method can be analyzed in terms of the distribution of energy in the basis of eigenvectors of the matrix. The algorithm as written below guarantees that the residual is less that ϵ. This is expected to happen in $\mathcal{O}\left(\log(\lambda_2/\lambda_1)\log\epsilon^{-1}\right)$ steps [97]. For linear

```
**Data:** an $n \times n$ matrix $M$, tolerance $\epsilon$, iteration limit k
**Result:** an approximate extremal eigenvector $\mathbf{v}$
$\mathbf{v} \leftarrow$ randn($n$);
$\mathbf{v} \leftarrow \mathbf{v}/\|\mathbf{v}\|$;
for $i$ in 1:k do
    $\mathbf{x} \leftarrow M\mathbf{v}$;
    $\mu \leftarrow \mathbf{v}^T\mathbf{y}$;
    if $\|M\mathbf{v} - \mu\mathbf{v}\| < \epsilon$ then
        return $\mathbf{v}$;
    end
    $\mathbf{v} \leftarrow \mathbf{x}/\|\mathbf{x}\|$;
end
return $\mathbf{v}$
```

Algorithm 1: The Power Method

systems, the *condition number*, $\frac{\|A\|}{\|A^{-1}\|}$ is the appropriate measure of sensitivity of the system [35]. For extremal eigenvalue problems the relevant measure is the spectral gap $|\lambda_1 - \lambda_2|$ [73].

Iterative methods can generate solutions to arbitrary approximation factors. Both runtime and solution accuracy increase with the number of iterations performed. Iterative methods [58, 79] for solving the eigenvector problem $A\mathbf{x} = \lambda\mathbf{x}$ have been shown to provide fast approximate solutions. For example, the implicitly restarted Arnoldi method (IRAM) allows one to solve for a small number of eigenvalues of a linear operator A [83]. A function that evaluates the action of A on arbitrary vectors along with $O(n(k+p) + (k+p)^2)$ space is sufficient to use the method. A practical implementation of the Arnoldi method, which is commonly used in practice, can be found in [58].

For a matrix A and a unit vector $\mathbf{x}$ define two numerical quantities of interest, the error and the residual. The error in an eigensolver is measured as $e = \left\| \mathbf{x} - \mathbf{q}\mathbf{q}^T\mathbf{x} \right\|$, the norm of the projection of the vector onto the eigenspace. The residual r is defined as $\|A\mathbf{x} - \mu\mathbf{x}\|$ where $\mu = \frac{\mathbf{x}^T A\mathbf{x}}{\mathbf{x}^T\mathbf{x}}$ is the Rayleigh Quotient of $\mathbf{x}$. The error represents distance between the computed solution and the true solution. Residual measures how close the computed solution is to solving the eigenequation. The goal of an eigensolver is to produce an $\mathbf{x}$ that satisfies $e = 0$. Since this cannot be done exactly (for general problems), the error and residual evaluates the quality of a solution. In practice, the error cannot be measured directly, and we must rely on theorems relating the residual to the error. The standard approach is to iterate until r is less than a prescribed tolerance chosen by the user. One cost of using iterative methods is that the level of accuracy necessary to solve a problem must be chosen by the user or dictated by the application. Section 5.4 addresses this choice for spectral partitioning.

3.3.1 Nonnegative Matrix Factorization (NMF)

Paatero and Tapper [72] formulates the Nonnegative Matrix Factorization problem as “Positive Matrix Factorization”. Lee and Seung [57] defines Kullback-Leibler (KL) Divergence as the closeness of the input matrix to the product of low rank factors. This leads to the Multiplicative Updates algorithm which is straightforward to implement. However, Gonzales and Zhang [36] proves that the Multiplicative Updates algorithm lacks converge guarantees. Convergence to a stationary point is one of the desiderata for optimization algorithms, and C.J. Lin [61] proposes a projected gradient descent based algorithm that converges to a stationary point. Similarly, Cichocki et al. [13, 12] proposes the Hierarchical Alternating Least Squares (HALS) algorithm which is independently described in Ho et al. [42] as rank-one residue iteration (RRI). Kim, He and Park [51] gives a unified framework based on Block Coordinate Descent that explains convergent NMF algorithms. Kim and Park [52] proposes a fast greedy active set based method to solve the NMF problem with convergence guarantees. The greedy active set method is the fastest convergent algorithm in the literature. Given the sufficient literature for NMF algorithms, we address graph applications of NMF.

Much of the literature on social networks using NMF assumes the input to be an adjacency matrix where each element $A_{i,j}$ represents the strength of connection between node i and node j. The most important applications of NMF on adjacency matrices are graph clustering and community detection. Kuang, Yun, and Park [55] applies NMF to the symmetric adjacency matrix of an undirected graph to cluster the vertices. Wang et al. [94] detects communities in social networks with NMF. Psorakis, Robert, Ebden, and Sheldon [76] proposes a Bayesian framework with NMF for detecting overlapping communities in networks. Yang and Leskovec [99] applies NMF to extremely large social networks to detect overlapping communities.

The primary applications of NMF for social network analysis are clustering and community detection; however, Tong and Lin [87] examines a related problem of anomaly detection in social networks using a nonnegative matrix factorization with a nonnegativity constrained additive residual representing the anomalous edges. The additive component is sparse and nonnegative thus containing edges which deviate from the low rank structure of the adjacency matrix. By applying NMF to recursively extracted local vertex features, Henderson et al. [40], discovers the roles of vertices within the structure of a static graph. This role extraction (RolX) method distinguishes network role discovery from network community discovery. NMF provides natural explanation of network role discovery.

Irregular memory access, difficult load balancing, and difficult partitioning on distributed memory systems. are perpetual difficulties of traversing large and irregular graphs such as social networks. Factorizing the adjacency matrix of a large and irregular social network directly suffers the same problems. To overcome these difficulties, Chapter 4 proposes an approach combining high performance graph processing algorithms with parallel, dense, linear algebra algorithms to extract insight from the graph.

3.3.2 Numerical Methods for Data Mining

Many data mining algorithms are phrased as optimization problems with numerical solutions [54]. The solution to the original data mining problem depends on the accuracy of the solution to the induced numerical problem, and many theoretical results quantify the relationship between the exact solution to the numerical problem and the quality of the solution to the data mining problem. However, there is little work evaluating the quality of a data mining solution produced by an approximate solution to the numerical problem. In this paper, we address this topic for spectral partitioning. Spectral partitioning is performed in two steps. First, one or more vectors approximating some eigenvectors of a graph matrix are computed and then those vectors are used to partition the graph. The eigenvector computation step is often treated as a primitive operation without considering the trade-off between runtime and accuracy. This is the case in [48], which evaluates the running time and quality, in terms of conductance, of both spectral and other partitioning algorithms. Pothen et al. [75], when applying spectral partitioning to distributed memory sparse matrix computation, recognized the value of low accuracy solutions. Since solving sparse linear systems is the computational goal, producing an accurate solution to the eigenvalue problem in order to solve a single linear system is not feasible. Thus, understanding the relationship between the error in eigenvalue approximation and the error in the original data mining problem is important. This work addresses the effect of the error in the computed eigenvalue on the quality of the sweep cuts of the computed eigenvector. Allowing more error in the eigenvector computation improves runtime performance with a potential loss in partition quality. Section 5.4 provides a stopping criterion with the same guarantees as the true eigenvectors. Section 5.6 shows that this stopping criterion yields good performance on real world networks. If the partition quality is not affected too greatly, trading quality for performance can be useful, especially for computationally expensive problems on large datasets.

Although eigenvectors produced by these methods are approximations, the impact of the error of these approximation techniques on the error of the original data mining solution has not been sufficiently studied. When eigenvectors of a kernel matrix are approximated with the power method and then k-means is applied to these approximations to cluster the graph, the k-means objective function is well approximated when using approximate eigenvectors [7]. The bounds given in [7] depend on using the k eigenvectors to partition into k parts and depend on the kth spectral gap. On the approximate eigenvectors, k-means is faster and sometimes more accurate in terms of normalized mutual information compared to using exact eigenvectors.

Instead of using k-way partitioning, this work focuses on partitioning into two clusters based on sweep cuts of a single approximate eigenvector. Because two way partitioning can be used recursively to find small communities, successfully partitioning into two groups is sufficient. The effects on multilevel partitioning [49], multiway partitioning, and local methods to improve the cut are beyond the scope of this work.

Other work focuses on the impact of probabilistic sampling error on data mining quality. In the context of Graham (kernel) matrices, Huang et al. [44] study the effect of perturbing the original data points on the spectral partitioning method. A similar topic is pursued in [98], where data points are quantized to reduce bandwidth in a distributed system. This work differs because it treats the data as correctly observed and evaluates error in the iterative solver.

When applied to large graphs, the Laplacian eigenvalues can be very close together implying that small residual tolerances are necessary to give the same pointwise guarantees. Small residual tolerance lead to many iterations and long run times. Chapter 5 studies the impact of numerical error in order to understand what accuracy levels are necessary to recover the data mining information and to derive stopping specific to the spectral clustering problem.

In the problem of graph partitioning, multiple good partitions may exist. In a resource constrained environment, one would like to be able to recover one of these near optimal partitions while expending as few resources as possible. Chapter 6 shows that finding vectors which produce these near optimal partitions is much less expensive than highly accurate approximations to the eigenvectors.

3.4 Streaming Algorithm concepts

Early work on the theory of streaming algorithms involves summarizing data streams. In a seminal paper by Flajolet and Martin [29], the data is presented in a streaming context and the number of distinct elements must be counted. The algorithms in this field are streaming but the analysis of that data is not necessarily temporal. Feigenbaum et al. [27] have contributed to one model of streaming graph analysis by considering the "semi-streaming model" where graphs are presented "as a stream of edges in adversarial order" and the goal is to compute properties of the graph in one or sub-linearly many passes over the edge stream. This semi-streaming model takes the perspective of a fixed graph with limited access to the data. The work addresses the theoretical issues in computing solutions to "classical graph problems" with necessary approximations due to the constraints on accessing the edges.

Another way to handle a large graph $G = (V, E)$ is to consider it as $|V|$ points in $|V|$ dimensional space, where V is the vertex set, and then use dimensionality reduction techniques. These methods are used in the sketching data structures found in Ahn, Guha and McGregor [1], where linearity of the sketch gives rise to incremental and dynamic updates to the data structure. A more traditional way to apply dimensionality reduction to graph analysis is to apply a matrix approximation to the adjacency matrix directly and use the approximation to study the graph. One example of this work is Colibri-D [88].

In the *massive streaming data analytics model* [22], we view the graph of social media events as an un-ending stream of new edge updates. For a given interval of time, we have the static graph, which represents the previous state of the network, and a sequence of edge updates that represent the new events that have taken place since the previous state was recorded. An update can take the form of an insertion representing a new edge, a change to the weight of an existing edge, or a deletion removing an existing edge.

Previous approaches have leveraged traditional, static graph analysis algorithms to compute an initial metric on the graph and then a final metric on the graph after all updates. The underlying assumption is that the time window is large and the network changes substantially so that the entire metric must be recomputed. In the massive streaming data analytics model, algorithms react to much smaller changes on smaller time-scales.

Given a graph with billions of edges, inserting 100,000 new edges has a small impact on the overall graph. An efficient streaming algorithm recomputes metrics on only the regions of the graph that have experienced change. This approach has shown large speed-ups for clustering coefficients and connected components on scale-free networks [22, 24].

This work builds on the high performance software package for streaming graph analysis called `STINGER`, which runs on large shared memory parallel computers. The key data structure is a blocked adjacency list that allows for efficient insertion, modification and deletion of edges [1]. The paradigm for an algorithm using `STINGER` is to perform initial work using a static graph that is possibly empty along with parameters to the algorithm. Then as edges are inserted from a stream, the algorithm (kernel)

[1]See http://www.stingergraph.com/ for code and data

maintains a data structure on the graph as changing edges necessitate changes to the data structure. These edges are received as batches which allow for parallelism. After each update a vertex feature is computed for each vertex in parallel and this is transmitted to another process that is awaiting the values of the vertex feature. The `STINGER` package allows for coordination between kernels so that multiple vertex features can be computed simultaneously without duplicating the effort and memory requirements of updating and storing the graph. This coordination is handled by running a main server which transmits the new edge updates to each kernel and then synchronizes the stages of computation.

Each algorithm produces a vector of length $|V|$ that is stored for analysis. The computation after each batch considers all edges in the graph up to the current time. As the graph grows, the memory will eventually become exhausted, requiring edges to be deleted before new edge insertions can take place. We do not consider this scenario, but propose a framework by which we can analyze the graph in motion.

An evaluation of the performance of the `STINGER` platform and various vertex features is presented in [21]. The time for various algorithms is measured by running them on an Intel Xeon E5–2670 with 16 hyperthreaded cores and 64 GiB of main memory. A fully operational server which is concurrently performing the data ingest and each feature computation is used to give an approximation of a realistic workload in operation. The list of computed features is degree velocity, Pagerank, and approximate betweenness centrality, where degree velocity is the change in degree for each vertex. The updates are performed in batches of 5000 to expose parallelism, because inserting a single edge into an adjacency list leads to a serial computation. Large batches also amortize the overhead of communicating the edges between processes. Since the computation of different kernels takes different amounts of time per batch and `STINGER` preserves the temporal ordering of the edge stream, we cannot allow faster algorithms to run ahead of the slower algorithms. This produces an overall system that takes as long as the slowest algorithm.

In an experiment, `STINGER` ingested a total of 53 batches of 5000 edges from the Hurricane Sandy Twitter dataset. The average time required to insert and update 5000 edges in the `STINGER` data structure was 2.89 milliseconds with a median of 2.85 milliseconds. This yields an update rate of 1.73 million updates per second. The degree velocity kernel had an average update time of 24.2 milliseconds per 5000 edges and a median update time of 24.4 milliseconds, and an update rate of 207,000 updates per second. The Pagerank kernel had an average update time of 132 milliseconds per 5000 edges and a median update time of 126 milliseconds, with a capacity to process 38,000 updates per second. The betweenness centrality kernel had an average update time of 214 milliseconds per 5000 edges and a median update time of 193 milliseconds, processing 23,400 updates per second. The more complex information takes longer to extract. The time to update degree velocity is closer to the raw update time than to the time to update Pagerank and betweenness centrality. Global information about the graph is more expensive than local information, but this additional computation provides deeper insight into the vertex behavior.

CHAPTER IV

UNDERSTANDING DYNAMIC GRAPH STREAMS

This chapter proposes a general technique for applying statistical methods to streaming graph data. Despite an algorithmic approach to streaming data, we lack statistical methods to reason about the dynamic changes taking place inside the network. These methods are necessary to perform reliable anomaly detection in an on-line manner. Section 4.1 analyzes the values of vertex features in order to understand the dynamic properties of the graph. This allows us to leverage existing techniques from statistics and data mining for analyzing time series. Behavioral clusters of vertices can be found by applying a clustering technique to these features after they have been computed by a streaming graph processing system such as `STINGER`. Section 4.2 applies NMF to the extracted features in order for form clusters of vertices with similar temporal behavior. These techniques are evaluated on two real world data sets a collection of tweets regarding Hurricane Sandy and a sample of internet traces provided by CAIDA.

4.1 Insight from Streaming Features

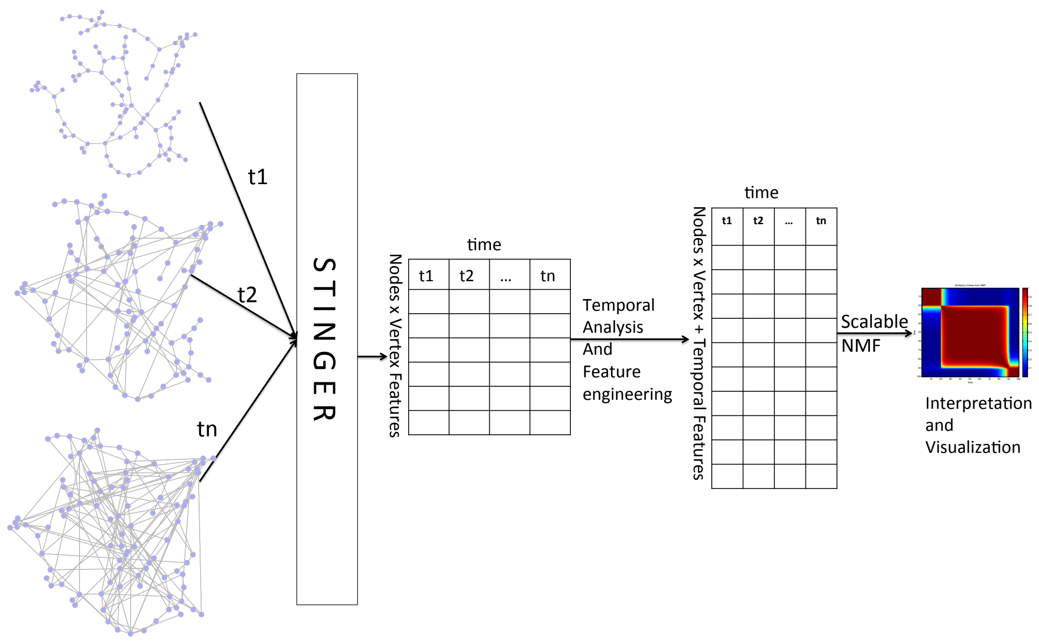
There are many domains of data analysis that can be modeled with the graph abstraction. In particular we are interested in social networks and internet connection networks. These networks are collections of interactions occurring in complex patterns. Analyzing these patterns is essential to leveraging the information contained in these networks. Because the most important networks are the networks that are in heavy use right now, methods to understand temporal patterns in dynamic networks are important. Some networks do not naturally handle deletions, for example Twitter and IP networks where messages are sent and received. In these cases we count the number of messages as the edge weight. With a dynamically changing graph where only those edges occurring in the past can be accessed, there are a new set of temporal queries to answer. This work contributes to the analysis of modern graph problems that only appear when the edge set is fluctuating over time. We provide insight into applications of temporal data analysis techniques to large data sets that are well represented by the dynamic graph abstraction.

The availability of big data has driven an adoption of large scale statistical techniques, both classical and modern. These techniques are not immediately applicable to graph data and this leaves analysts separated from their familiar software tools. In order to connect graph analysis and statistical reasoning, we introduce vertex features which can be calculated efficiently and then analyzed using familiar large scale statistical software tools. This connection is bidirectional because statistical analysis of vertex features informs the computation of additional features. The curse of dimensionality indicates that applying vectorial techniques directly to a large graph will be difficult and plagued by overfitting. The observed difficulty of writing scalable parallel graph algorithms for scale-free and irregular graphs advises against writing inferential and mathematical code to analyze the graphs directly. In this framework we address this gap by first applying non-inferential graph code to generate vectorial data that is statistically well behaved, then applying a state of the art vectorial technique to this data, which provides insight into the original graph. A representation of this framework is presented in Figure 1.

4.1.1 Vertex Features

We define a *graph kernel* as an algorithm that builds a data structure or index on a graph.[1] We define a *vertex feature* as a function from the vertex set to the real numbers. Graph kernels are analogous to high performance computing kernels such

[1]This is distinct from a kernel function that compares the similarity of two graphs.



t1
t2
tn
STINGER
Nodes x Vertex Features
time
t1
t2
...
tn
Temporal Analysis And Feature engineering
Nodes x Vertex + Temporal Features
Scalable NMF
Interpretation and Visualization

Figure 1: Our framework combines sparse parallel graph algorithms and dense parallel linear algebra algorithms.

as applying a stencil operator. These low level computation kernels are applied to build scientific software for answer questions about physical systems. Vertex features are analogous to data features in machine learning where they provide a set of observations describing each element of the dataset. In the context of a dynamic graph, we use temporal vertex feature to refer to any vertex feature that captures temporal or dynamic information. For example, a connected components algorithm is a graph kernel because it creates a mapping from the vertex set to the component labels. The function that assigns each vertex the size of its connected component is a vertex feature. The function that assigns to each vertex the number of new vertices in its component at time t is a temporal vertex feature. Graph kernels can be used as subroutines for the efficient computation of vertex features. Any efficient parallel implementation of a vertex feature will depend on efficient parallel graph kernels. Another example of a kernel-feature pair is breadth-first search (BFS) and the eccentricity of a vertex, which is the maximum distance from v to any vertex in the connected component of v. The eccentricity of v can be computed by measuring the height of the BFS tree rooted at v.

Vertex features are mathematically useful ways to summarize the topological information contained in the edge set. Each feature compresses the information in the graph; however by compressing it differently, an ensemble of vertex features can extract higher-level features and properties from the graph. Each feature also defines a sense in which two vertices are similar. For example, two vertices with the same degree have the same neighborhood size, and two vertices with the same clustering coefficient have the same local triangle structure.

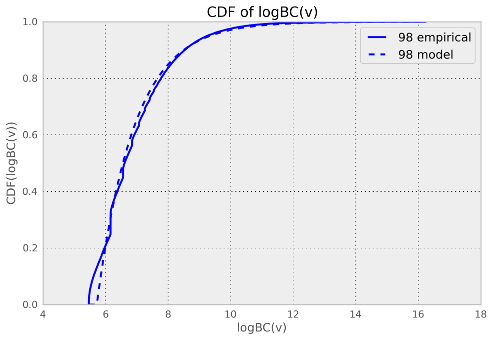
One implication of this framework for graph analysis is that the computation of these vertex features will produce a large amount of extracted data from the graph. As shown in Figure 1, the data for each feature can be stored as an $|V| \times |T|$ array, which is indexed by vertex set V and time-steps $T = t_1, t_2, \ldots, t_n$. These dense matrices are amenable to parallel processing using techniques from high performance linear algebra. In Section 4.2.2.4 we apply both Nonnegative Matrix Factorization (NMF) and Singular Value Decomposition (SVD) in order to infer the temporal relationships between the vertices. Once we have created these dense matrices of features, we can apply large scale data analysis techniques in order to gain insight from the graph in motion, mainly to study the rise and fall of influential nodes over time. Another advantage of the vertex feature approach is the ability to visualize the large temporal graph. Typical visualizations attempt to show the nodes and connections on the plane capturing the spatial relationship of the nodes. Using vertex features, one can see the relationship between the behavior of the vertices without bombarding the user with all of the edges. This is demonstrated along with our observations in Section 4.3.

4.1.2 Temporal Analysis

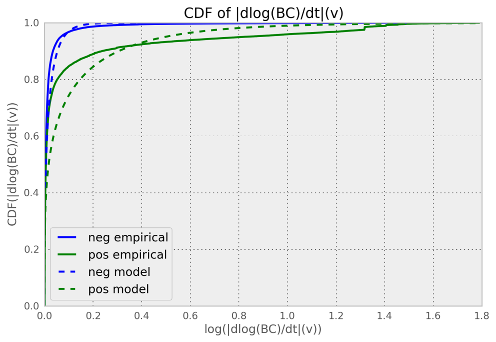
In order to explain our temporal analysis and construction of useful vertex features, we introduce a running example using Twitter data. Specifically, we assembled a corpus around a single event maximizing the likelihood of on-topic interaction and interesting structural features. At the time, there was great concern about the rapid development of Hurricane Sandy. Weather prediction gave more than one week of advanced notice and enabled us to build a tool chain to observe and monitor information regarding the storm on a social network from before the hurricane made landfall through the first weeks of the recovery effort. For diversity we also demonstrate these techniques with applications to CAIDA data that was collected passively on internet traffic.

In order to focus our capture around the hurricane, we selected a set of hashtags (user-created metadata identifying a particular topic embedded within an individual Twitter post) that we identified as relevant to the hurricane. These were *#hurricane-sandy, #zonea, #frankenstorm, #eastcoast, #hurricane,* and *#sandy.* Zone A is the evacuation zone of New York City most vulnerable to flooding.

The data set includes 1.4 million public Twitter posts starting from the day before the storm made landfall as an edge list. This edge list includes any mentions of one user by another as well as cases where a user "retweeted" or reposted another user's post, because retweets mention the author of the original Tweet similar to a citation. The dataset included over 1,238,109 mentions from 662,575 unique users. We construct a graph from this stream in which each username is represented as a vertex. The file contains a set of tuples containing two usernames which are used to create the edges in the graph. The temporal ordering of the mentions is maintained through the processing tool chain resulting in a temporal stream of mention events encoded as graph edges. The edge stream is divided into batches of 10,000 edge insertions for our analysis. As each batch is applied to the graph, we compute the betweenness centrality, local clustering coefficient, number of closed triangles, and



(a) The cumulative distribution function for logarithm of betweenness centrality empirical (solid) and exponential best fit (dashed)



(b) CDF of the derivative evaluated at time 98. Vertices with increasing Betweenness centrality are separated to show the difference in distribution.

Figure 2: The distributions of logarithm of betweenness centrality and the derivative of logarithm of betweenness centrality provide information about vertices with rare behavior.

degree for each vertex in the graph.

4.1.3 Temporal Features

In order for this framework of connecting graph algorithms to machine learning algorithms to be applied, one must first choose a set of vertex features to compute. We discuss here how one can study a direct vertex feature, as well as temporal features that can be derived from the direct features.

4.1.3.1 Betweenness Centrality

Centrality metrics on static graphs provide an algorithmic way to measure the relative importance of a vertex with respect to information flow through the graph. Higher centrality values generally indicate greater importance or influence. Betweenness centrality is described in Section 3.1. In order to account for the heavy tail, we examine the logarithm of the betweenness centrality score for each vertex. Since many vertices have zero or near zero betweenness centrality we add one before taking the logarithm and discard vertices with zero centrality.

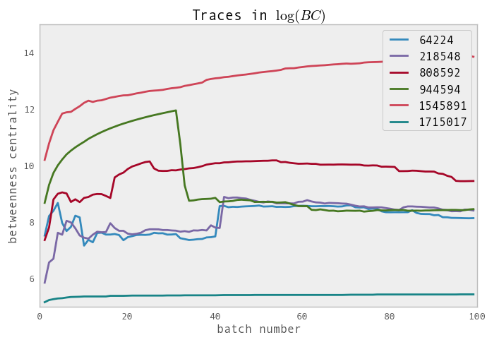
For the Twitter data set after inserting 98 batches of edges we find that the right half of the distribution of logarithm of betweenness centrality can be modeled as an exponential distribution. The cumulative distribution function (CDF) is well approximated by $F(x) = 1 - exp\left[-\lambda(x - x_0)\right]$ where the maximum likelihood estimates for the location and rate parameters are $x_0 = 5.715$ $\lambda = 1.205$ respectively. Since betweenness centrality estimates are more accurate for the high centrality vertices [32], we focus our analysis on the vertices whose centrality is larger than the median.

Figure 2a shows both the empirical CDF and the modeled CDF for the log(betweenness centrality). It is apparent in the figure that the exponential distribution is a good fit for the right tail. We can use the CDF to assign a probability to each vertex, and these probabilities can be consumed by an ensemble method for a prediction task. This will allow traditional machine learning and statistical techniques to be combined with high performance graph algorithms while maintaining the ability to reason in a theoretically sound way. Such distributional analysis connects the analysis of graph topology to the well studied fields of parameter estimation and nonparametric density estimation. Understanding the distribution of these features allows us to choose transformations of the data that will be helpful in follow-on analysis.

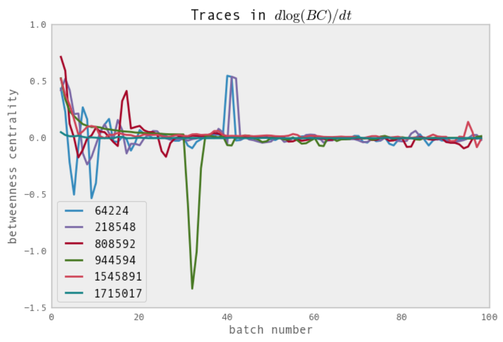
Figure 3a, traces the value of betweenness centrality for a selection of vertices over time. In the sociological literature, this corresponds to a longitudinal study. It is clear that there is a significant amount of activity for each vertex. Such a longitudinal study of vertices can be performed for any vertex feature that can be devised for graphs.

4.1.3.2 Finite Differences

Tracking the derivatives of a feature can provide insight into changes that are occurring in a graph in real time. Equation 11 defines the (discrete) derivative of a vertex feature for a vertex v at time t using the following equation where $b(t)$ is the number of edges inserted in batch t. Note that the derivative of a vertex feature is also a



(a) Traces of betweenness centrality value for selected vertices over time.



(b) The derivative of the logarithm of betweenness centrality values for selected vertices.

Figure 3: Tracing values of vertex features over time provides information about vertex behavior and changes in graph structure.

vertex feature.

$$\frac{df}{dt}(v,t) = \frac{f(v,t+1) - f(v,t-1)}{b(t)+b(t+1)} \tag{11}$$

When concerned about maximizing the number of edge updates that can be processed per second, fixing a large batch size is appropriate. However when attempting to minimize the latency between an edge update and the corresponding update to vertex features, the batch size might vary to compensate for fluctuations in activity on the network. Dividing by the number of edges per batch accounts for these fluctuations. For numerical or visualization purposes one can scale the derivative by a constant.

The data can be examined in a cross section by examining the distribution of the derivatives at a fixed point in time. Figure 2b shows the CDF of $\frac{d}{dt}|log(BC)|(v)$ at batch 98 on a log scale, where the vertices are grouped by the sign of their derivative. The separation of the two curves show that the distribution of increases in logarithm of betweenness centrality is different than the distribution of decreases in logarithm of betweenness centrality. By counting the number of vertices of each group, we determined that the most vertices decrease in betweenness centrality in this batch.

For example, Figure 3b shows the derivative of logarithm of betweenness centrality. These traces indicate that changes in the betweenness centrality of a vertex are larger and more volatile at the beginning of the observation and decrease in magnitude over time. The reason for taking logs before differentiation is that it effectively normalizes the derivative by the value for that vertex.

Because the temporal and topological information in the graph is summarized using real numbers, we can apply techniques that have been developed for studying measurements of scientific systems to graphs. Since the derivative of a vertex feature is another vertex feature, these derivatives can be analyzed in a similar fashion. By estimating the distribution of $\frac{df}{dt}$ for any feature we can convert the temporal information encoded in the graph into a probability for each vertex.

By measuring dynamic properties of the graph change-points in the edge stream can be detected as in Figure 4 Section 4.1.4 applies this method behavioral outliers, that is, a set of vertices whose behavior is significantly different from the bulk of the vertices.

Once we construct the vertex features for each time coordinate, we are able to take finite differences of a time series. We can measure the overall activity of a graph according to a statistic such as clustering coefficient by counting the number of vertices that change their value in each direction. For clustering coefficient this is shown in Figure 4. One observation is that more vertices have increasing clustering coefficient than decreasing clustering coefficient. We also learn that only a small fraction of vertices change in either direction. Monitoring these time series could alert an analyst or alarm system that there is an uptick in clustering activity in the graph. The increase in the number of vertices with increasing clustering coefficient around batch 70 corresponds to October 30th, which is the day after the storm passed through New Jersey. By studying these changes we can describe changes to the system at the graph level, which complements our knowledge of the specific behavior of the individual vertices.

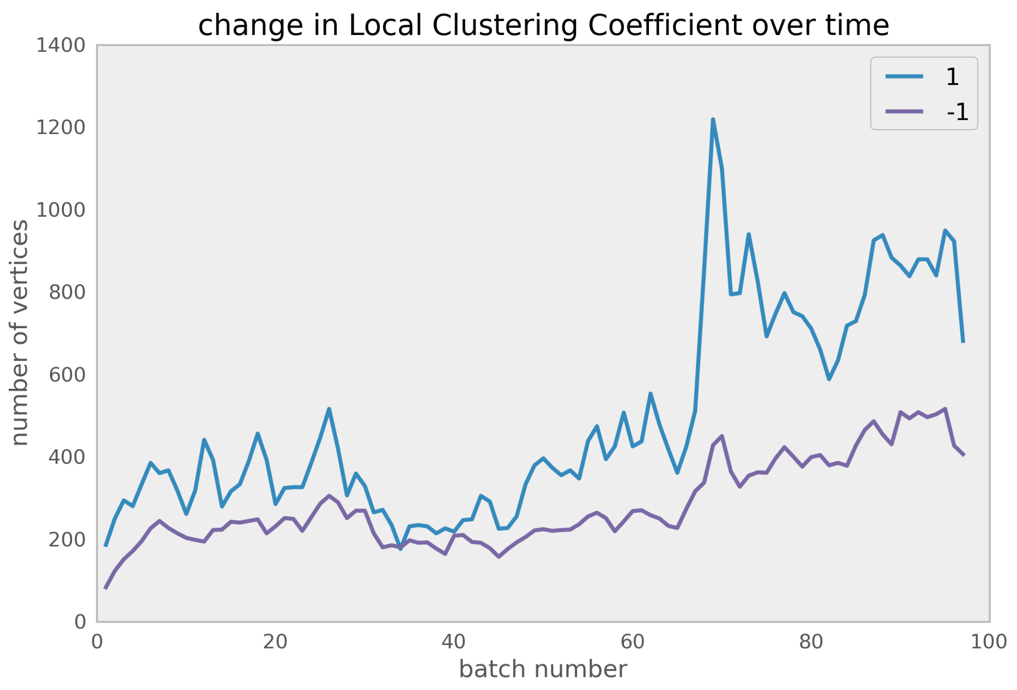


Figure 4: Counting vertices by sign of their finite differences at each time step.

4.1.4 Anomaly Detection

Because anomaly detection is a vague problem, we focus on the more well-defined problem of outlier detection. For the purposes of this work, the outliers of a data set are defined as the points that appear in the lowest density region of the data set.

One method for finding outliers is to assume that the data are multivariate Gaussian and use a robust estimate of mean and covariance [78] — a method known as the elliptic envelope. This is appropriate when the data is distributed with light tails and one mode. The one class support vector machine (SVM) can be used to estimate the density of an irregular distribution from a sample. By finding the regions with low density, we can use an SVM to detect outliers [81]. This nonparametric technique is well suited to this data analysis task.

We seek a method to apply these multivariate statistical methods to our temporal graph data. Because we have been computing the triangle counts and local clustering coefficient for each vertex in an on-line fashion, each vertex has a time series. This time series can be summarized by computing moments.

We extract the mean and variance of the original local clustering coefficient series. In order to capture temporal information, we use the finite differences of the local clustering coefficient and extract the mean and variance. Summary statistics for the differences are taken over non zero entries because most vertices have no change in local clustering coefficient at each time step. These summary statistics are used as features that represent each vertex. This is an embedding of the vertices into a real vector space that captures both topological information and the temporal changes to the network. This embedding can be used for any data mining task. Here we use an outlier detection algorithm to illustrate the usefulness of this embedding.

Once these features are extracted, the vertices can be displayed in a scatter plot matrix. This shows the distribution of the data for each pair of features. These scatter plots reveal that the data is not drawn from a unimodal distribution. Because the robust estimator of covariance requires a unimodal distribution this eliminates the elliptic envelope method for outlier detection.

Using a single class support vector machine with the Gaussian Radial Basis Kernel, we are able to estimate the support of the data distribution. Because the SVM is sensitive to scaling of the data, we whiten the data so that is has zero mean and unit standard deviation. By grouping the data into inliers and outliers, we see that the two distributions are distinct in feature space.

Figure 5 shows a scatter matrix with the inlying vertices in blue and the outliers in red. We can see that in any pair of dimensions some outliers are mixed with inliers. This indicates that the SVM is using all of the dimensions when forming a decision boundary. The diagonal plots show normalized histograms in each dimension with inliers in blue and outliers in red. These histograms show that the distribution of the inliers differs significantly from the distribution of the outliers. This indicates that the SVM is capturing a population that is distinct from the majority population in

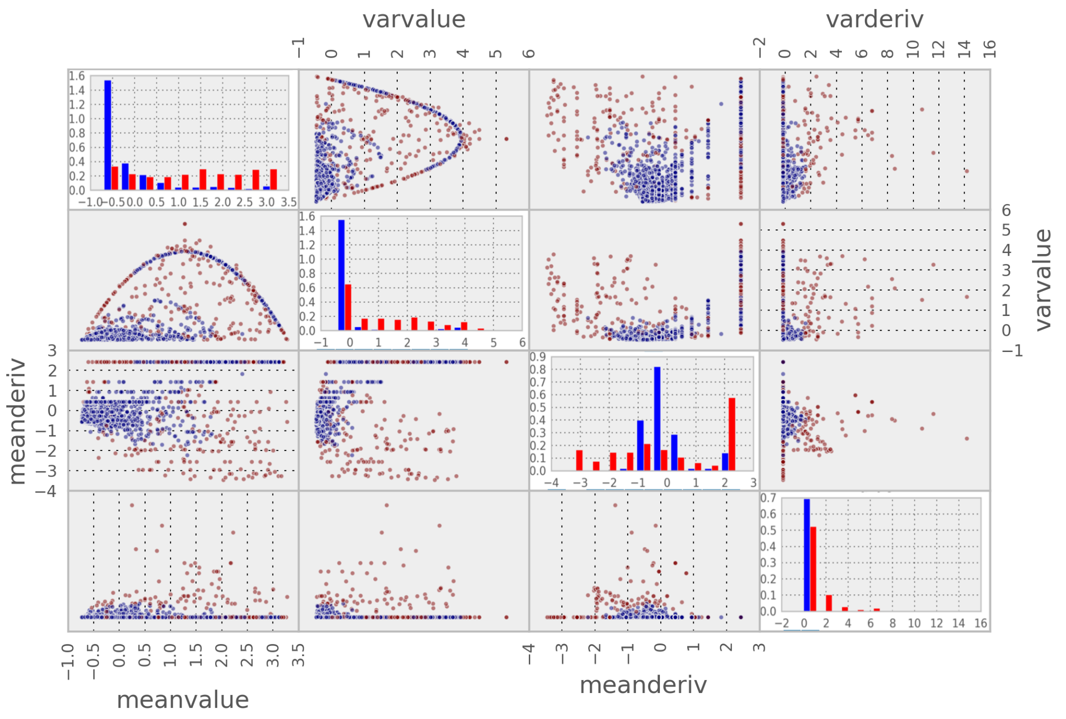


Figure 5: Scatter plot matrix showing the outliers (red) and normal data (blue)

terms of their dynamic clustering coefficient behavior.

Considering the betweenness centrality of vertices represents the vertices in terms of their influence over time. The influence behavior as measure by the mean logarithm of betweenness centrality value and variance in logarithm of betweenness centrality over time has a distribution well approximated by a two dimensional Gaussian distribution as seen in Figure 6. Figure 6 shows a scatter matrix with the inlying vertices in blue and the outliers in red as found by an elliptical envelope method.

Inspection of the results indicate that the outliers are primarily accounts associated to news agencies and jokes. The news agencies are spreading information that is useful to the public and this analysis indicates that their messages are being spread effectively. Joke twitter accounts are popular because their comment is highly entertaining. According to these methods, joke accounts are also effective in spreading their messages. The news agencies are found in the high mean and low variance region of the data while the joke accounts are in the high variance region. This matches the

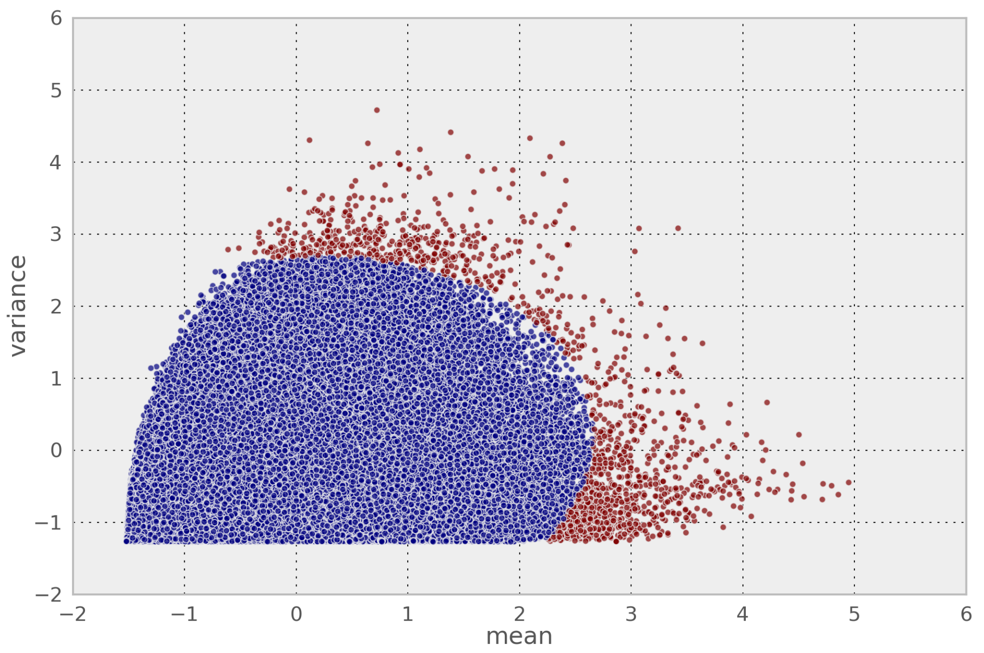


Figure 6: Scatter plot matrix showing the betweenness centrality outliers (red) and normal data (blue)

intuition that news agencies are consistently influential over time while the jokes become popular quickly and then lose their influence rapidly. While it would be possible to curate a list of known news agencies and look for them in the data using the text of the tweets and metadata, a goal of graph analysis is to use topological information to find the users who effectively spread their messages. Finding known news agencies without a priori looking for them supports the value of these methods.

Through the use of temporal features, we are able to learn about the behavior of the individual vertices and changes to the graph as a whole. These features allow analysts to study the vertices using visualization techniques that avoid the complex topology of the graph instead conveying more compact summaries of the vertex behavior. This feature based approach also allows the cutting edge techniques from statistics and machine learning to apply to graph analysis without inventing new complex problems.

4.2 Behavioral Clustering in Dynamic Graphs

Another study using this framework clusters the vertices based on their temporal behavior. The extracted features are designed to capture influence, and the clustering produced by nonnegative matrix factorization recovers groups of vertices which rise and fall in influence together. Experiments show the parallel scalability of this approach. In this work we show that the vertex features, as explained above, can be used to understand the vertex behavior as well as the behavior of the entire graph as a whole. The nonnegative factorization of these feature matrices provides a clustering of the vertices into groups and a segmentation of the edge stream into phases, which are two important data analysis tasks.

This section contributes a method for combining sparse parallel graph algorithms with dense parallel linear algebra algorithms in order to understand dynamic graphs including the temporal behavior of vertices. This method is the first to cluster vertices in a dynamic graph based on arbitrary temporal behaviors. In order to successfully implement this method, we develop a feature based pipeline for dynamic graphs and apply Nonnegative Matrix Factorization (NMF) to these features. We demonstrate these steps with a sample of the Twitter mentions graph as well as a CAIDA network traffic graph.We contribute and analyze a parallel NMF algorithm presenting both theoretical and empirical studies of performance. This work can be leveraged by graph/network analysts to understand the temporal behavior cluster structure and segmentation structure of dynamic graphs.

In this work we show that the vertex features, as explained above, can be used to generate an understanding of the vertex behavior as well as the behavior of the entire graph as a whole. The nonnegative factorization[2] of these feature matrices provides a clustering of the vertices into groups and a segmentation of the edge stream

[2]Matrix factorization and low rank approximation are used interchangeably for consistency with the literature.

into phases, which are two important data analysis tasks. These feature matrices are broadly applicable and many applications are beyond the scope of this paper, including tensor factorizations which will provide latent feature based understanding of the three way interaction between the vertices, the different features, and the time-steps.

- Graph analyst can use traditional statistics and modeling to understand the data and then apply an advanced technique to solve a particular problem. The first step allows for the design of useful features also called feature engineering which is an integral step of the machine learning and data analysis workflow.
- Vertex level temporal changes can be connected to graph level temporal changes. Latent features from a matrix factorization are graph level features. Discovering explicit graph features is difficult.
- Treating the vertices of a graph as n points in n dimensional space and applying a vector data analysis technique ignores the complex topological information in the graph. This method uses graph theoretic knowledge and computations to represent each vertex in a vector space and use vector data analysis on these points. This approach allows us to leverage parallel graph algorithms that are not aware of probabilistic or machine learning techniques, and familiar dense linear algebra instead of performing inference on sparse graph data structures.
- Smooth temporal changes in the graph are found with SVD. NMF produces clear demarcations of graph level changes revealing phases of graph activity.

4.2.1 Contributions

With the relevant literature in mind, we contribute a broad framework for connecting high performance graph algorithms to large scale data analysis techniques. Our method is the first to find behavioral vertex clusters in a dynamic graph. We present a feature based pipeline for dynamic graphs and apply Nonnegative Matrix Factorization (NMF) to these features, which reveals vertex clusters and phases of network activity over time.

The algorithms used in this paper present good performance in both theory and practice. Three important tasks about temporal graphs are clustering the vertices, segmenting the edge stream and visualizing changes to the graph. The low rank approximation method used in this paper provides answers to all three of these important questions.

Section 4.1.2 takes a stream of Twitter posts ("Tweets") from the time surrounding the landfall of Hurricane Sandy, a tropical storm that hit the Northern Atlantic coast of the United States, and forms a temporal social network of mentions. We compute graph metrics, including betweenness centrality and Pagerank, in a streaming manner for each batch of new edges arising in the network. Statistical analysis of these features leads to the construction of additional features. In Section 4.3, we describe some insights into cluster structure in the CAIDA Network derived by NMF. The performance of our parallel NMF algorithm is empirically demonstrated in Section 4.4 and validated our theoretical analysis of parallel NMF in Section 4.2.2.2

4.2.2 Foundations

In this section, we present the necessary foundations for describing our feature based graph analysis pipeline. We discuss our representation of a graph in terms of vertex features, our parallel platform for computing such features, the relevant algorithm for NMF used to detect temporal clusters, and methods for comparing the similarity of graphs over time.

4.2.2.1 Block Principal Pivoting Algorithm for NMF

After computing vertex features using STINGER, we will apply NMF in order to make inferences about vertex and graph behavior. Here we discuss the NMF problem for general matrices F. Given a non-negative matrix $F \in \mathbb{R}_+^{m\times n}$, the problem of Non-negative Matrix Factorization (NMF) is to find two matrices $W \in \mathbb{R}_+^{m\times k}$ and $H \in \mathbb{R}_+^{k\times n}$ such that $F \approx WH$. Formally the NMF problem can be defined as

$$W^*, H^* = \underset{W,H}{\arg\min} \|F - WH\|_F^2 \quad \text{s.t.}, W \geq 0; H \geq 0; \tag{12}$$

The NMF problem is non-convex to solve W and H together. However, if we assume one of them is given, solving the other is a convex problem. Hence, we alternatively solve two sub problems of finding W and H until a stopping criteria.

$$H \leftarrow \underset{H\geq 0}{\arg\min} \|WH - F\|_F^2 \text{ and } W \leftarrow \underset{W\geq 0}{\arg\min} \left\|H^T W^T - F^T\right\|_F^2. \tag{13}$$

By taking transposes, we see that the algorithm for finding W is the same as the algorithm for finding H, thus we focus our attention on solving for H. The columns can be partitioned into independent blocks and each block can be solved for with a concurrent NNLS with multiple right hand sides. We leverage the fact that if $\mathcal{I}$ is a partition of the index set, we can expand the Frobenius norm as shown in Equation 14.

$$\|WH - F\|_F^2 = \sum_{I\in\mathcal{I}} \sum_{i\in I} \|Wh_i - f_i\|_2^2 \tag{14}$$

There are different algorithms for solving the above NMF problem. Also there are many variants depending on the characteristics of the input matrix such as symmetric [55], bounded [47] etc. For a general non-negative input matrix, the most common algorithms are multiplicative update [57], Hierarchical Alternative Least Squares (HALS) [12] and Block Principal Pivoting [52]. Kim, He and Park [51], present a detailed comparison and the properties of these algorithms using Block Coordinate Framework. For this paper, we are using BPP as it is the fastest and scalable NMF algorithm.

$$\sum_{I\in\mathcal{I}} \|WH_I - F_I\|_F^2 \quad ; \text{where,} H_I \in \mathbb{R}_+^{k\times|I|} \tag{15}$$

We can decompose the Frobenius norm into a sum over columns and consider only a subset of the columns, which gives equation (15). Minimization of the above expression is a non-negative least squares (NNLS) problem with multiple right hand sides. In general the NNLS problem with multiple right hand sides is

$$\min_{X \geq 0} \|CX - B\|_F^2 \tag{16}$$

The Block Principal Pivoting (BPP) algorithm listed as Algorithm 2 for the above problem is discussed by Kim and Park [52]. We are using this algorithm because it has scalable performance as demonstrated in 4.4. Here we briefly explain the algorithm which is an iterative algorithm inspired by the active set method. If we knew which indices correspond to nonzero values in the optimal solution, then computing the optimal solution is an unconstrained least squares problem on these indices. Call the set of indices i such that $x_i = 0$ the active set and the remaining indices the passive set. The BPP algorithm works to find this active set and passive set. Since the above problem is convex, the correct partition of the optimal solution will satisfy the Karush-Kuhn-Tucker (KKT) condition. The BPP algorithm greedily swaps indices between the active and passive sets until finding a partition that satisfies the KKT condition. In the partition of the optimal solution, the values for indices that belong to the active set will be zero. The values of the indices that belong to the passive set are determined by solving the least squares problem using normal equation [3] restricted to the passive set.

We have all the necessary building blocks to explain our parallel multicore NMF algorithm using ANLS-BPP. Broadly the algorithm has two major components. (a) Given $W \geq 0, F \geq 0$, find a non-negative $H \geq 0$ and then given F and H, find a W, alternating until the stopping criteria is satisfied. (b) Partitioning columns of H and

[3]The solution to least squares problem $||Ax - b||_2^2$ is obtained by solving the system of linear equations for $A^T Ax = A^T b$

W calling $ANLS-BPP$ with each partition.

```
input  : Matrix $F \in \mathbb{R}_+^{m\times n}$, target rank $k$
output: Matrices $W \in \mathbb{R}_+^{m\times k}$, $H \in \mathbb{R}_+^{k\times n}$
Initialize W as nonnegative random matrix ;
while stopping criteria not met do
    // Find matrix H given F,W
    $H \leftarrow$ findOptimal $(F, W)$;
    // Find matrix W given F,H
    // Transpose the returned matrix
    $W \leftarrow$ findOptimal$(F^T, H^T)^T$;
end
function findOptimal($F \in \mathbb{R}_+^{m\times n}$,$W \in \mathbb{R}_+^{m\times k}$)
    $\mathcal{I} \leftarrow$ Partition $(1, \ldots, n)$;
    for $I \in \mathcal{I}$ do in parallel
        // Assign the returned matrix from BPPNNLS to the index $I$
           of $H$
        // BPPNNLS with multiple right hand sides from [52]
        // According to equation (16), $C = W$ and $B = F_I$
        $H_I \leftarrow$ BPPNNLS $(W, F_I)$;
    end
```
Algorithm 2: Multicore NMF Algorithm

4.2.2.2 NMF Parallelism Theory

The $ANLS-BPP$ routine in Algorithm 2 is an iterative method. It requires the computation of few matrix matrix products once and a least square for each iteration of the method. The matrix matrix multiplication or level-3 BLAS cost is $O(mkn+k^2m)$ and the time spent over all iterations is $O(k^4(m+n))$. The upper bound comes from the fact that the active set method can take at most k iterations in order to find a solution of length k and we are solving for m NNLS vectors for W and n NNLS vectors for H. Each iteration of the active set method requires at most a Cholesky decomposition for $k\times k$ matrix and a two triangular solves involving $k\times k$ matrix. Since the parallelization is over the number of solution vectors, we can see a parallel run time on p processors that is $O(k^4m/p)$ when solving for W and $O(k^4n/p)$ when solving for H where p is limited to m or n respectively. The asymptotic cost to compute the product of a $m \times k$ matrix with an $n \times k$ matrix is well known as $O(mkn/p)$ in shared memory parallel computers. This indicates that for rank k decompositions where k^3 exceeds either m or n, the limiting step in the computation is the k least squares step for the larger of W or H. In detail assume $m > k^3 > n$, then $mkn > nk^4$ so matrix multiply exceeds the cost of the k least squares solve for H. However, $mkn < mk^4$ implying that the cost of least squares for W exceeds the matrix multiplication cost. In our case there are more vertices than time-steps and so the k least squares is the dominant cost when k^3 exceeds n. One implication of this analysis is that the chosen number of communities effects not only the overall runtime of the algorithm but also which step is the dominant cost. We study these runtime considerations empirically in Section 4.4

4.2.2.3 Graph Similarity Metrics

One task when analyzing a dynamic graph, is to determine how much the graph has changed over a period of time. We will use several measures to address this task and draw some conclusions about their relative merit. The simplest measure determines the similarity without looking at which edges we insert but only the number of edges inserted and the size of the graph. We define the relative impact of a batch of insertions as the number of edges in the batch divided by the average size of the graph during the insertions. Letting the $|E_i| = bi$ be the number of edges in the graph at time i, which corresponds to a batch size of b edges, we obtain Equation 17 for the relative impact of batches i to $i + s$.

$$RI(i, i+s) = \frac{2bs}{bi + b(i+s)} = \frac{s}{i + s/2} = \left(\frac{i}{s} + \frac{1}{2}\right)^{-1} \tag{17}$$

The relative impact allows us to reason a priori about what the behavior of a measure of graph similarity over time. If we fix a gap or delay of size s then the impact of each batch decreases over time. Also as the delay s increases, the relative impact increases. The similarity of a dynamic graph to future instances should behave as the inverse of the relative impact of those edge insertions. Thus similarity should weaken as the gap size increases and similarity should strengthen as the size of the graph increases.

Additionally, we can define a measure of similarity for any set of vertex features. Let $F \in \mathbb{R}^{m\times n}$ be a feature matrix, where each row represents a vertex and each column represents a time stamp, $(F^T F)_{ij}$ represents the similarity of the graph at time i to the graph at time j. These similarities are the inner product (cosine) similarities one gets by representing the graph as a point in n dimensional space where the vth coordinate is the value of the vertex feature for vertex v. These similarities are specialized to particular vertex features and their nature is defined by the behavior captured by the chosen vertex features.

Using matrix factorization $F \approx WH$ we can get another similarity measure defined as $H^T H$. This similarity matrix takes account of the relationships between the vertex feature at different vertices. The nature of the factorization determines the behavior of this similarity. For example, the Singular Value Decomposition (SVD) for a given rank $F \approx U\Sigma V^T$ gives two orthogonal matrices U, V and a nonnegative diagonal matrix Σ that best approximate F in the two norm among rank k matrices. We can compute a similarity based on this factorization as $F^T F \approx V\Sigma^T\Sigma V^T$. By definition of the SVD, this similarity will contain a smooth approximation of the similarity and will account for the interrelation of the vertices. The orthogonal vectors found by the SVD are those that best represent the variance in the data. Each vector well approximates the variance of the entire data set. According to Kuang, Yun, and Park [55] NMF is a form of clustering, where $W \in \mathbb{R}^{m\times k}$ is a set of basis vectors such that each basis vector is a representative of a cluster, and $H \in \mathbb{R}^{k\times n}$ represents the distribution of every data point over these k clusters. Hence, H identifies clusters in time. Thus $H^T H$ will represent the graph similarity accounting for the clustered structure of the vertex set. In contrast to the SVD, each representative of a cluster is predominantly determined by the members specific to that cluster. We observe the differences between the SVD and NMF in Section 4.3. These similarity measures are useful ways to compare each snapshot of the graph to previous and future snapshots.

4.2.2.4 Temporal Graph Analysis using NMF

Here we reiterate the connections among NMF, vertex feature extraction and graph analysis. Our objective is to study the behavioral changes of vertices and the graph over time. Towards this end, we construct vertex features using highly scalable infrastructure such as `STINGER`. We study the vertex changes over time using the exploratory data analysis techniques explained in the previous section. Given the vertex features and the temporal features, in this section we explain our novel method to understand behavioral clusters and temporal changes in the graph using NMF.

Identifying and constructing explicit features of graphs that are good for understanding temporal changes is difficult. In order to avoid constructing explicit features, we construct implicit (latent) features through the low rank approximation of the original feature matrix. These implicit features capture the structure of the vertex features extracted from the graph. Given the temporal edge stream containing N vertices, we form a matrix $F \in \mathbb{R}_{+}^{m \times n}$ matrix where n is the number of time-steps and $m = d|V|$ is the number of vertex and temporal features times the number of vertices, as explained in previous section. Given this matrix F, we build graph features of dimension k over n time-steps, such that these latent features are good representation for the graph as a whole. It is important to appreciate the difference between vertex features and latent features. Vertex features are constructed from the graph structure such as between centrality, Pagerank, and clustering coefficient, whereas latent features are implicitly defined from the values of these vertex features.

In our scenario, we use NMF to separate the input matrix $F \in \mathbb{R}_{+}^{m \times n}$, which relates the vertex features to the time-steps, into $W \in \mathbb{R}_{+}^{m \times k}$, which relates the vertex features to the latent features, and $H \in \mathbb{R}_{+}^{k \times n}$, which relates the latent features to the time-steps. This process discovers k latent features that mediate the interactions between vertices and time-steps. The matrix W reveals a clustering on vertices, and the matrix H provides a representation for studying the temporal changes in the graph as a whole. Since a direct clustering of F would not produce both pieces of information concurrently, we choose a low rank approximation approach. In the next section we show a case study from a real world graph and demonstrate the usefulness of W, H for understanding the temporal behavior in the dynamic graph.

4.3 Case Study - CAIDA Dataset

In this section, we present the observation and analysis of temporal clusters generated by NMF. As illustrated in Figure 1, we are using the explicit vertex features matrix as input to the NMF. The output of NMF gives the temporal clusters of nodes that aids in visualizing the rise and fall of influential nodes. Experiments on IP network data demonstrate the utility of these vertex feature matrices. The STINGER library is used to extract the betweenness centrality and Pagerank for each vertex at each time-step. Because this is a bipartite network we do not find any triangles. The CAIDA passive traces form an IP network where vertices are host IP addresses and an edge means that a packet was sent from one host to another. This graph has timestamped edges extracted from the packet capture (`pcap`) data. CAIDA releases this data in an anonymized form under an acceptable use policy and it can be obtained from them [86]. [4] Edges are processed as undirected for the purpose of graph kernels. We examine logarithm of betweenness centrality, its squared discrete derivative, and the exponentially weighted moving average and exponentially weighted moving standard deviation. These derived statistics capture the temporal nature by finite differences and the distributional aspects by capturing the center and spread of the distribution of recently observed data. These features are arranged as a matrix $F \in {\mathbb{R}_+}^{m \times n}$. Since

[4] `http://www.caida.org/data/passive/passive_2014_dataset.xml`

we perform only edge insertions we use exponentially weighted moving averages to discount the historical data. If more augmenting features are found to be interesting, they can be computed without impacting the behavior of the graph algorithmic code.

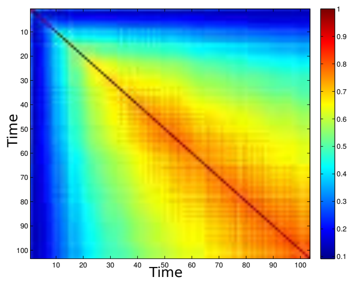
In this context, the matrix $F^T F$ (normalized so that $(F^T F)_{i,i} = 1$) gives similarity of the graph over time. The construction of $F^T F$ as the similarity between timesteps, implies that two timesteps are similar if for each feature and each vertex the values are similar. The unnormalized version of $F^T F$ is show in Equation (18).

$$(F^T F)_{s,t} = \sum_{v \in V, k \in \{1 \dots d\}} f_k(v,s) f_k(v,t) \tag{18}$$

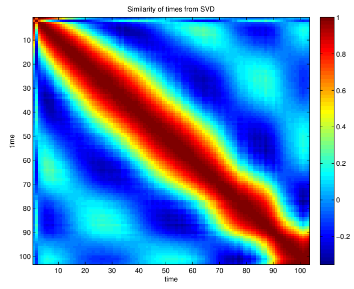
The formula in Equation (18) is a covariance between timesteps. By normalizing so that the diagonal entries are 1, we get the correlation version.

The nature of the graph similarity is determined by the choice of feature. If one is interested in the influence structure of the graph, then influence metrics such as betweenness centrality and Pagerank can be used. The i, jth entry of this matrix is the similarity between the graph at time t_i to time t_j. When we examine this matrix in Figure 7a all we see is the large scale trend. By factorizing this matrix into low rank factors, we reveal a more refined picture of the dynamics. Figure 7b shows the similarity after using the Singular Value Decomposition (SVD) $F = U\Sigma V^T$ to account for the intervertex dependencies.

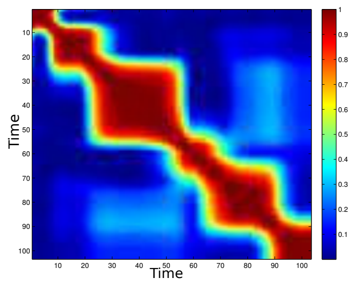
The approximate block structure of $H^T H$ indicates a clustering on the rows and columns. The NMF has determined that certain time-steps belong together. Because we are inserting edges into the graph, we are not surprised that the clusters are identified as contiguous subsets of the time domain. By comparing these similarity matrices to the formula for relative impact given in Equation 17, we see that all three of these similarity functions agree with the dependence on s. As the gap between two observations i and $i + s$ increases, the similarity of the graph at those two times decreases. In the case of the basic similarity that does not account for vertex interactions, we see only this trend effect. The SVD accounts for the vertex interactions and



(a) $F^T F$ does not account for any dependence between the vertices or between the derived features. This shows only the large scale trend.



(b) The SVD accounts for intervertex and interfeature dependence in a smooth manner.



(c) The H factor from NMF gives an approximately block diagonal structure. This shows clusters in time.

Figure 7: The temporal similarity of betweenness centrality structure between all pairs of time-steps, in internet traces data.

gives a more specific similarity. However, the real difference occurs when we examine the cluster structure of the data. The NMF similarity $H^T H$ gives a discrete sense of similarity. If two time-steps are in the same phase (cluster of time-steps), then the similarity is very high, and if they are in different clusters the similarity is very low. This validates the claim that NMF produces a segmentation of the edge stream into phases of activity. Pivoting our attention to W, the left hand factor, we discover the clusters of vertices. The index of the largest element in each row of W indicates to which cluster that vertex belongs. Returning to the longitudinal study of vertices and grouping the vertices into their clusters, Figure 8 shows the plot of logarithm of betweenness centrality for each vertex over time, which reveals a clear pattern. Most elements of each cluster peak in betweenness centrality around the same time. The timing and duration of these peaks correspond to the diagonal blocks of $H^T H$. We have clustered vertices into groups that rise and fall in influence together. The similarly matrix $V\Sigma^T\Sigma V^T$, is smoother as every data point is projected onto the basis U representing the entirety of the data. Both the SVD and NMF approximations

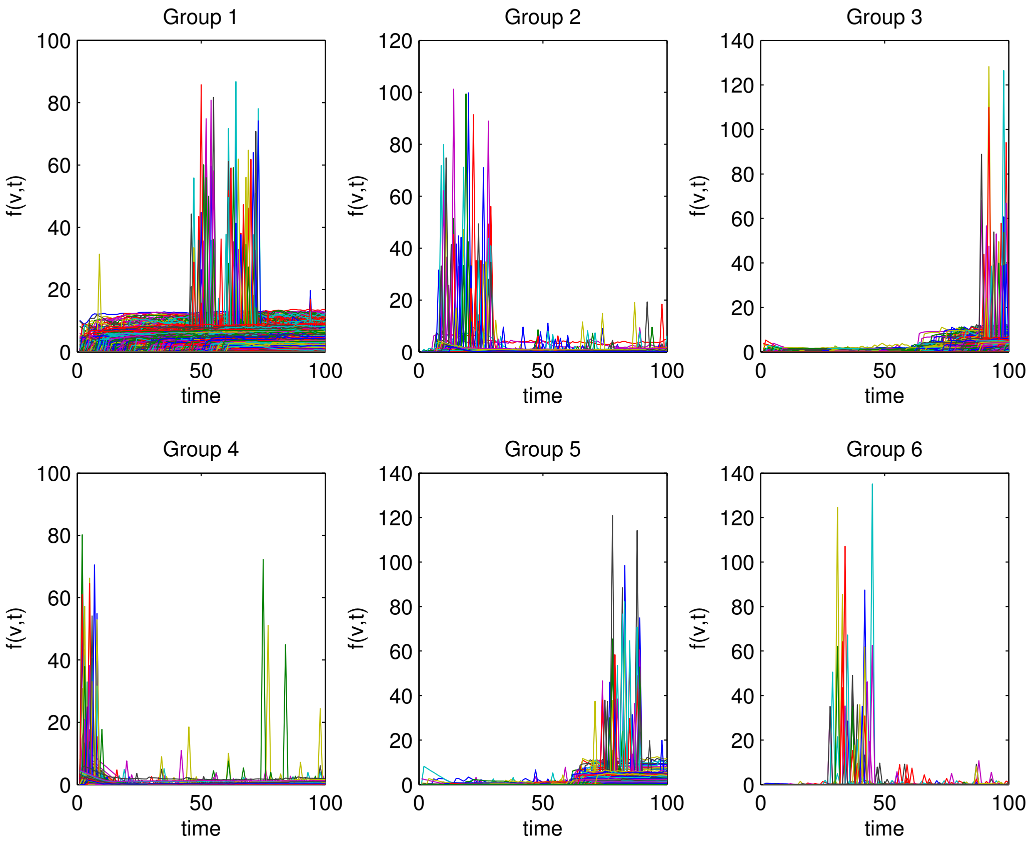
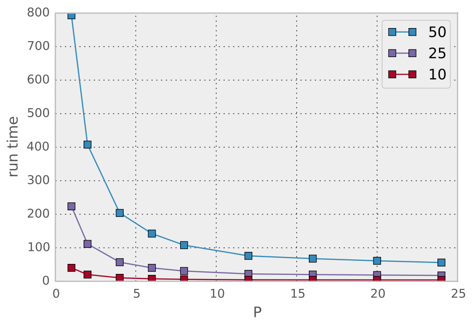


Figure 8: Longitudinal plot of a sample of vertices from each cluster identified by NMF on the betweenness centrality derived feature matrix. Each subplot represents a group of vertices that peak at the same time. The logarithm of betweenness centrality is shown on the ordinate.

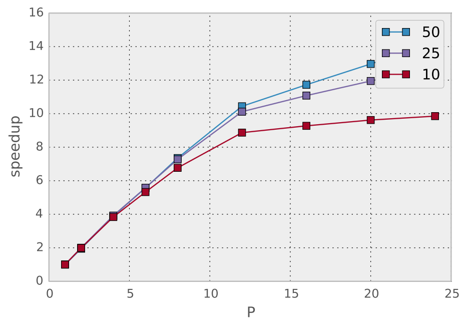
account for the interdependence of vertices and features, thus revealing more information than $F^T F$. In the next section we present the scalability experiments of the NMF algorithm.

4.4 Performance Experiments

In order to validate the application of NMF to graph analysis, we give empirical evidence that the parallel BPP algorithm presented in Section 4.2.2.1 is scalable. The experiments are conducted on a dual socket Intel Xeon CPU E5-2620 machine clocked at 2.00GHz. Each socket has 6 cores along with hyperthreading, hence there are 12 cores in total and 24 threads of execution. Our input matrix is a dense matrix of size 2840256x103, where 103 represents number of time-steps and 2840256 are 8 different features observed over each of 355032 vertices. One of the parameters for BPP is the rank k which determines the number of clusters and for practical applications, k is chosen on the order of 10's. Since the rank of the matrix is 103 (the number of independent columns) we show experiments where $k = 10, 25, 50$. For this experimentation, our code uses the Intel MKL implementation of BLAS and LAPACK. Since our machine has 12 physical cores but can support 24 threads with hyperthreading, we choose the number of threads as 1, 2, 4, 6, 8, 12, and 24 so that we also study the performance when the number of threads is more than number of physical cores. In the Figure 9, the number of cores is presented on the x-axis and the running time (Figure 9a) and speedup factor (Figure 9b) is shown on the y-axis. From these graphs we observe linear scaling up to the number of physical cores on the system. When hyperthreading is used, the speedup is no longer linear, as we achieve a speedup of approximately 10 on 12 cores and a speedup of approximately 12 on 24 cores. From these experiments we also infer that the algorithm achieves less parallel efficiency when forming a very low rank approximation of the matrix. Since the work per core is $O(k^4(m+n)/p)$ we expect more parallel efficiency for larger values of



(a) NMF Runtime



(b) NMF Speedup

Figure 9: NMF—BPP Scalability experiments on a 2840256×103 dense matrix

k than for very small values of k. According to the BPP algorithm, when we find H, the majority of the time is taken towards computing the matrix multiplications $W^T F$ and $W^T W$, which surpasses the time needed for computing the small matrix H of size 103xk. Where as, in the case of computing W, the computational effort for the matrix multiplications $H^T H$ and HF^T was smaller than for the iteration to find W. Thus the effort for finding every vector w_i is very scalable with multiple threads and this advantage is more pronounced for $k = 50$. The observed linear scaling with better efficiency for larger values of k confirms the behavior predicted by theory. One implication of this parallel performance analysis is that the partitioning of matrices for distributed memory processing must account for the asymmetry between the left and right factors.

4.5 Conclusions and Future Work

We contribute a novel framework for making inferences about dynamic vertex behavior based on streaming graph computations. This framework provides both a principled method of finding anomalous vertices and a new method for detecting clusters of vertices based on temporal behavior. The inferred temporal behavior is interpretable, easy to visualize, and found without explicitly searching for a predefined pattern. We developed a successful feature based pipeline for dynamic graph streams and applied NMF to these features. The computed low rank approximation is useful for several graph analysis tasks including temporal similarity at the graph level, vertex clustering and graph temporal segmentation. We contribute a theoretical and empirical evaluation of a new parallel algorithm for NMF based clustering, which scales up to 12 cores and exhibits good performance up to 50 clusters, even on graphs with large vertex sets. These techniques yield an algorithm which is linear in the number of vertices m and time-steps n when the number of temporal behavior clusters is $O((m+n)^{1/4})$. Application of this method to Twitter data reveals patterns, change points, anomalous vertices. A case study analyzing CAIDA internet traces provides understanding into the behavior of the ubiquitous internet connection graph. This feature based pipeline takes advantage of longitudinal studies of vertex behavior over time to inform the generation of derived features. A low rank approximation to the matrix of features produces clusters of vertices that rise and fall in influence together. The clusters of vertices correlate with sharp boundaries in time which form phases of the network's evolution. Our approach loosely couples parallel graph algorithms with machine learning algorithms, which allows the application of both systems effectively.

CHAPTER V

NUMERICAL METHODS FOR GRAPH PROBLEMS

When conducting any scientific or engineering task that involves approximate computations, the question of accuracy arises. It is impossible to execute the computations of continuous mathematics exactly in a computer. In order to solve the problems of continuous mathematics with computers, approximations must be made. One aspect of numerical analysis is to derive algorithms that give accurate approximations. Another aspect is to understand the properties of these approximations. This chapter studies numerical approximation for the graph partitioning problem. The goal is to compute good partitions of a graph efficiently. The spectral method is commonly used for this problem. When applying the spectral method we ask how accurate does the numerical solution need to be in order to provide correct partitioning of the graph. Section 5.1 discusses some details of the spectral partitioning method. Section 5.2 illustrates the spectral method for partitioning graphs from the stochastic block model. Section 5.3 provides a motivational example of using approximate solutions to a numerical problem in order to solve a graph partitioning problem. Section 6.2 studies the usages of approximate eigenvectors to partition graphs in detail, and Section 5.4 develops a new stopping criterion based on conductance. This chapter explores how analysis of numerical methods applied to graph analysis problems can improve our understand of both fields.

5.1 The Spectral Sweep Cut Algorithm

Spectral partitioning is a family of algorithms for finding good cuts of a graph.

1. Construct a graph matrix M one of $A, \hat{A}, L, \hat{L}, P$.

2. Compute some eigenvectors of M.

3. Use these vectors to compute a partition $V = \uplus_i S_i$.

These steps can be repeated to find smaller communities if necessary. The choice of $A, \hat{A}, L, \hat{L}, P$ depends on the structure of the analysis goals, data distribution, and chosen numerical methods. For recovering the stochastic block model parameters A provides the best estimator [65]. For finding local partitions the random walk matrix P is used [53]. This work uses $\hat{L}$ and $\hat{A}$ for finding global partitions as in [11].

A multidimensional embedding formed by computing more eigenvectors of the graph matrix allows one to make a multiway cut in one step, but is more difficult to analyze because there is a larger design space for spatial partitioning in multiple dimensions. This dissertation focuses on partitioning the graph into two parts using sweep cuts (Algorithm 4).

One way to apply spectral bisection to a graph given a vector $\mathbf{x}$ is to divide the graph at a threshold such as 0 or the median value of x_i. This method is sensitive to small errors because if the true value of $|x_i|$ is close to the threshold, then a small perturbation in $\mathbf{x}$, as measured in the two norm, can push x_i over the threshold. One solution to this problem is to compute the conductance of all n possible cuts, and take the optimal cut. These sweep cuts give partitions with at most the same conductance as using a fixed threshold, and often give much better cuts. An efficient algorithm for computing the conductance of all possible sweep cuts can be derived from the definition of conductance in Equation (3). Lemma 1 gives a recurrence relation for the cut size.

Lemma 1. *The edge cut size follows the recurrence relation*

$$E(S_i, \bar{S}_i) = E(S_{i-1}, \bar{S}_{i-1}) + \sum_{j \geq i} a_{i-1j} + \sum_{k<i-1} a_{k,i}.$$

Proof. Let $E_i = E(S_i, \bar{S}_i)$ be defined as $E_i = \sum_{k<i} \sum_{j \geq i} a_{kj}$

$$\begin{aligned} E_i - E_{i-1} &= \sum_{k<i} \sum_{j \geq i} a_{kj} - \sum_{k<i-1} \sum_{j \geq i-1} a_{kj} \\ &= \sum_{j \geq i} a_{i-1j} + \sum_{k<i-1} \sum_{j \geq i} a_{kj} - \sum_{k<i-1} \sum_{j \geq i-1} a_{kj} \\ &= \sum_{j \geq i} a_{i-1j} + \sum_{k<i-1} a_{k,i} + \sum_{k<i-1} \sum_{j \geq i-1} a_{kj} - \sum_{k<i-1} \sum_{j \geq i-1} a_{kj} \\ &= \sum_{j \geq i} a_{i-1j} + \sum_{k<i-1} a_{k,i} \end{aligned}$$

Thus, $E_i = E_{i-1} + \sum_{j \geq i} a_{i-1j} + \sum_{k<i-1} a_{k,i}$ and the conclusion follows. □

One must compute for all i the volume of the vertices $v_1 \ldots v_i$ according to $\mathrm{vol}(S_i) = \sum_{k \leq i} \sum_{k \leq i} a_{ij}$.

Lemma 2. *The volume of $S = \{1 \ldots k\}$ follows the recurrence relation*

$$V(S_k) = \sum_{j \leq k} a_{k,j} + \sum_{i \leq k-1} a_{i,k} + V(S_{k-1}).$$

Proof. Let $V_k = V(S_k) = \sum_{i \in S_k} \sum_{j \in S_k} a_{ij}$.

$$V_k = \sum_{i=1}^{k} \sum_{j=1}^{k} a_{ij} = \sum_{j=1}^{k} a_{kj} + \sum_{i=1}^{k-1} \sum_{j=1}^{k} a_{ij} = \sum_{j=1}^{k} a_{kj} + \sum_{i=1}^{k-1} a_{ik} + V_{k-1}$$

□

From these recurrence relations, algorithm 3 provides an efficient algorithm from computing the conductance of every possible sweep cut.

The sweep cut algorithm can be used to partition once an ordering of the vertices has been given. A convenient way to construct the ordering is to use an approximate eigenvector and sort the vertices according to their spectral coordinates. That is when using a vector $\mathbf{x}$ $v_j \mapsto v_j$ if $j = |\{k \mid x_k > x_i\}|$. According to Theorem 3 this method is guaranteed to approximate the true minimum conductance cut of the graph. Let `eigensolve`(G, r, ϵ) be a function that computes the second smallest eigenvector of the graph Laplacian of G, such as the power method, Algorithm 1, or IRAM [58].

```
**Data:** a graph G labeled $v_1 \dots v_n$
**Result:** an array of conductances $\phi\{v_1 \dots v_i\}$
$A \leftarrow \texttt{adjacency(}G\texttt{)}$;
$\phi = \texttt{zeros(}n\texttt{)}$;
$e = \texttt{zeros(}n\texttt{)}$;
for $i$ in $2{:}n$ do
    $e_i = \mathsf{sum}(A_{i-1,i:n}) + \mathsf{sum}(A_{1:i-1,i})$;
    $\mathsf{vol}_i = \mathsf{vol}_{i-1} + \mathsf{sum}(A_{i,1:i-1}) + \mathsf{sum}(A_{1:i,i})$;
    $\phi_i = e_i / \min(\mathsf{vol}_i, n - \mathsf{vol}_i)$;
end
```

Algorithm 3: Sweep Cut

Also let `argsort`(**v**) be the function returning a permutation of the indices of **v** that sorts the values of **v** with the corresponding function `permute(`*G,s*`)` applying that permutation to a graph. Algorithm 4 provides an algorithm for partitioning a graph G into two parts given the functions `eigensolve()` and `argsort()`, which yields the permutation that sorts an input array.

```
**Data:** a graph G, a tolerance $\epsilon$
**Result:** a partition S, conductance $\phi(S)$
$n = \mid V \mid$;
$\mathbf{x}^0 = \texttt{randn}(n)$;
$\mathbf{x} = \texttt{eigensolve}(A, \mathbf{x}^0, \epsilon)$;
$\phi = \texttt{zeros}(n)$;
$\mathbf{s} = \texttt{argsort}(\mathbf{x})$;
$\phi = \texttt{sweep(permute(}G, \mathbf{s}\texttt{))}$;
$j, \phi_{\min} = \arg\min_i \phi_i$;
return $s_{1:j}, s_{j+1:n}$
```

Algorithm 4: Spectral Sweep Cut

Theses operations define a particular algorithm spectral sweep cut Algorithm 4. The remainder of this work uses spectral partitioning to refer to this particular algorithm.

5.2 The Stochastic Block Model

When studying the graph partitioning problem, a common class of problems to study is the stochastic block model, where the vertices in the same block are stochastically equivalent meaning they are interchangeable with no effect on the probability distribution [43]. The stochastic block model is a latent space model where the latent space is discrete. The model is parameterized by the set of blocks $C_1 \cup C_2 \cup \ldots C_q = V$, and the block interaction coefficients B. In a linear algebra interpretation the block assignment of the vertices is given by the matrix $Q \in \{0,1\}^{n \times q}$, with q denoting the number of blocks.

$$Q_{ik} = \begin{cases} 1 & : i \in C_k \\ 0 & : i \notin C_k \end{cases}$$

The probability of an edge between a vertex in C_k and a vertex in C_l is given by the block interaction coefficient $B_{k,l}$. This can be represented in matrices as

$$P_{ij} = \left[Q^T B Q\right]_{ij} = \sum_k \sum_l Q_{ik} B_{kl} Q_{lj} \tag{19}$$

The adjacency matrix $A \in \{0,1\}^{n \times n}$ is drawn from independent Bernoulli trials. The probability is given in equation 20.

$$P(A) = \prod_{i \leq n} \prod_{j \leq n} P_{ij}^{a_{ij}} (1 - P_{ij})^{1 - a_{ij}}. \tag{20}$$

When restricting to undirected graphs, one draws only from the upper triangle of indices satisfying $i < j$.

These block models are expressive enough to model diverse community structure. One such community structure is shown in Figure 10, which depicts a random stochastic block model graph both with a graph drawing and the sparsity plot. A consistent estimator of the block identities can be derived from the eigenvectors of the adjacency matrix of a graph drawn from the stochastic block model under certain conditions on B [64]. One condition is that the block association matrix B must be positive semidefinite. If the intrablock probabilities are larger that the sum of the interblock probabilities for each row of B, the Gershgorin Circle Theorem [35] implies B is positive semidefinite. The remaining conditions relate to the density of the graph, the separation of the eigenvalues and the separation of the rows of $Q\sqrt{B}$. For a wide

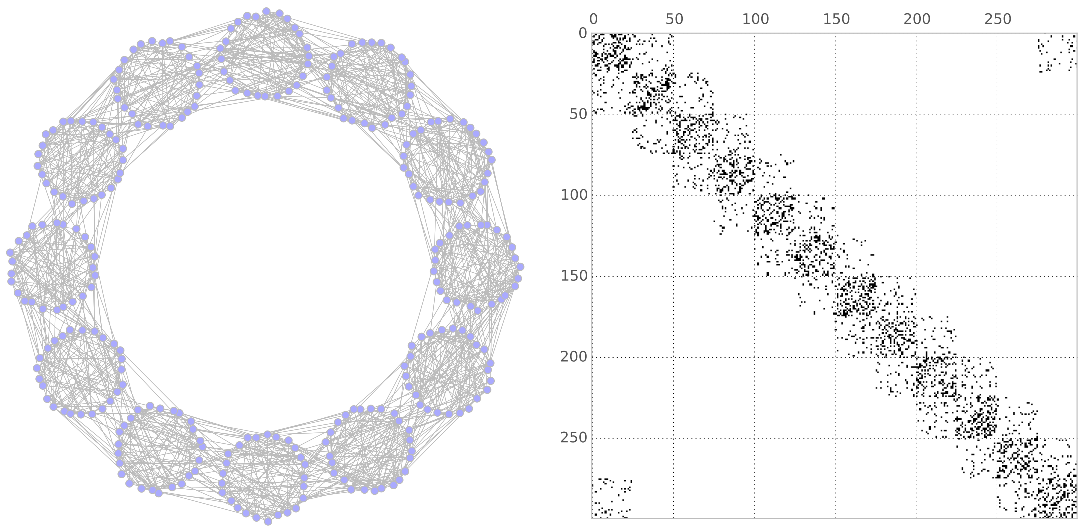


Figure 10: A stochastic block model with 12 blocks each of size 25 graph drawn to show structure (left). A sparsity plot of the same graph (right).

variety of parameters for the stochastic block model the eigenvectors can give perfect partitioning of the observed graphs.

As can be seen from Equation (19), the matrix of probabilities has at most q nonzero eigenvectors. When passing from the probability matrix P to the sampled adjacency matrix A, this low rank property implies that A has at most q large eigenvalues and at least $n - q$ small eigenvalues. When applying spectral sweep cut, Algorithm 3, to partition a stochastic block model graph, the goal is to recover a partition into two groups that does not split any of the blocks $C_1, C_2, \ldots C_q$. If Algorithm 4 is applied recursively and at each level no block is split, then all of the blocks will be recovered on the final level. This motivates convergence analysis of methods computing an approximation to a linear combination of dominant eigenvectors. Section 6.2 shows that when the adjacency matrix has only a small number of large eigenvalues, a linear combination of these eigenvectors are easily computed. In order to show that this approach is sufficient to recover graph partitions Section 6.3 examines a highly structured graph with this property.

For an example of a stochastic block model graph refer to Figures 10-12 The

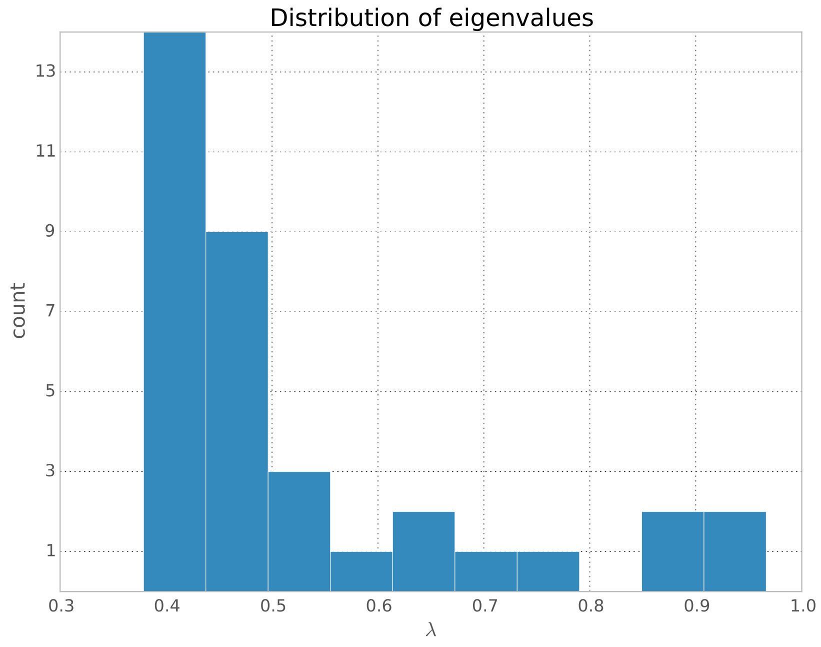


Figure 11: A histogram of the top 37 eigenvectors of the graph in Figure 10. The eigenvectors close to 1 correspond to vectors which reveal low conductance partitions.

graph in Figure 10 is drawn from a stochastic block model where B is circulant that is $B_{i,j} = B_{i-1,j-1}$ modulo q. The diagonal entries of B are large relative to the off diagonal entries. This implies that most vertices will have more neighbors in their community compared to outside of their community, which matches the intuitive definition of a community. The distribution of eigenvalues for the example graph can be seen in Figure 11. The adjacency eigenvalues are distributed such that q eigenvectors are close to 1 and the remaining $n-q$ eigenvectors are small in magnitude. In Figure 12 we see that the adjacency eigenvectors of a stochastic block model graph are concentrated into blocks Figure 12. An extreme form of this concentration is exploited in Section 6.3 to prove results about the numerical accuracy requirements of spectral partitioning.

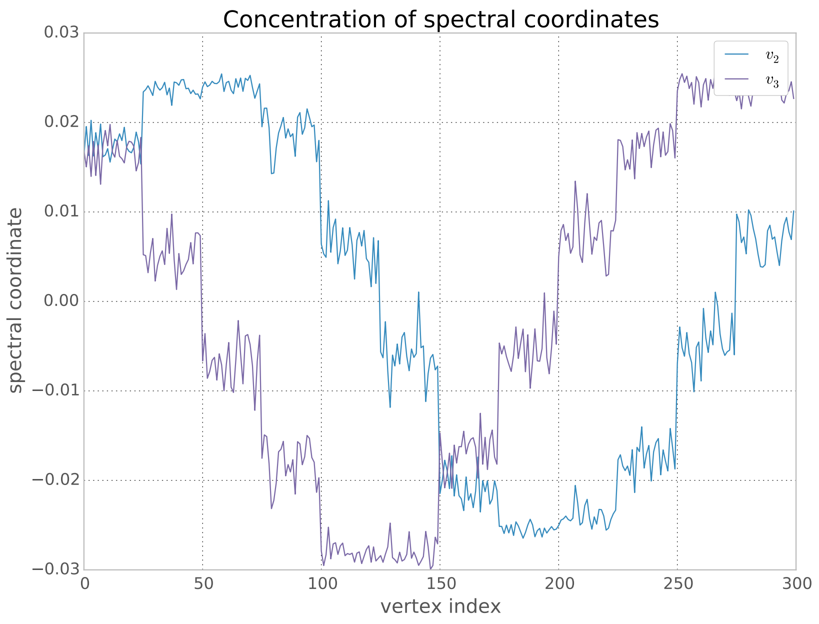


Figure 12: The top eigenvectors of the graph in Figure 10 show concentration for vertices in the same block. This concentration implies that the block identities can be recovered from the spectral coordinates.

5.3 High Quality Partitioning from Low Quality Eigenvectors

This section provides a motivational example of applying numerical analysis ideas to a graph problem yielding insight into a computational process. For some graphs, low-accuracy eigenvectors can outperform high-accuracy eigenvectors. Eigenvector solvers such as `ARPACK` [58] use random seed vectors to compute a candidate eigenvector. This candidate eigenvector is guaranteed to satisfy some approximation bound and depends on the chosen random seed. We can see the effects of treating the eigenvector returned from `ARPACK` as a random variable.

Fix a graph G and take an ensemble of random seeds $\mathbf{r}_1 \ldots \mathbf{r}_k$, then the output of `ARPACK` applied to this ensemble is $\mathbf{x}_1 \ldots \mathbf{x}_k$ where $\mathbf{x}_i = p(\hat{L}^{-1})\mathbf{r}$ using `ARPACK`'s `SM` setting and a prescribed tolerance ϵ. From $\mathbf{x}_i$ we can determine the best sweepcut $S_i, \bar{S}_i$ and compute the conductance $\phi(\mathbf{x}_i)$. In this way an eigensolver induces a distribution of conductances at tolerance ϵ.

On a $G_{n,p}$ random graph, we see that the cut size is more concentrated for tighter tolerances than for looser tolerances [25]. As we decrease the tolerance ϵ, we are more likely to get the same partition from the vectors in the ensemble. Since the goal is to find the minimum cut of the graph, we can take several approximate eigenvectors and take the best induced cut. Table 2 shows the distribution of cut size when an ensemble of eigenvectors are computed. The minimum of the distribution is the best cut induced at tolerance ϵ.

Results of this experiment conducted on an Erdős-Rényi random graph Laplacian with tolerance $10^{-2}, 10^{-4}, 10^{-8}$ are shown in Table 2. The smallest conductance value is found for $\epsilon = 10^{-2}$ which indicates that the low accuracy solution is outperforming the high accuracy solution.

Table 2: As the residual tolerance decreases the distribution of conductance values becomes more concentrated. However the minimum value seen in a sample increases. Each column represents the distribution over 30 samples.

Distribution of conductance sizes ϕ			
	1e-2	1e-4	1e-8
mean	0.317831	3.156708e-01	3.156708e-01
std	0.001842	2.258405e-16	2.258405e-16
min	0.314898	3.156708e-01	3.156708e-01
25%	0.315849	3.156708e-01	3.156708e-01
50%	0.318569	3.156708e-01	3.156708e-01
75%	0.318669	3.156708e-01	3.156708e-01
max	0.321631	3.156708e-01	3.156708e-01

In order to understand this phenomenon we examine the distribution of computed eigenvectors. Figure 13 shows the distribution of $\left\| |\mathbf{x}| - \bar{|\mathbf{x}|} \right\|$ where $\bar{|\mathbf{x}|}$ is the average solution produced[1] The quantity $\left\| |\mathbf{x}| - \bar{|\mathbf{x}|} \right\|$ represents the deviation of each solution from the average solution. As the computation becomes more accurate, this distribution becomes more concentrated.

This behavior is possible because we are relaxing the min cut partitioning problem

[1]The absolute values are necessary to account for $A\mathbf{x} = \lambda\mathbf{x}$ can be satisfied by both $\mathbf{x}$ and $-\mathbf{x}$.

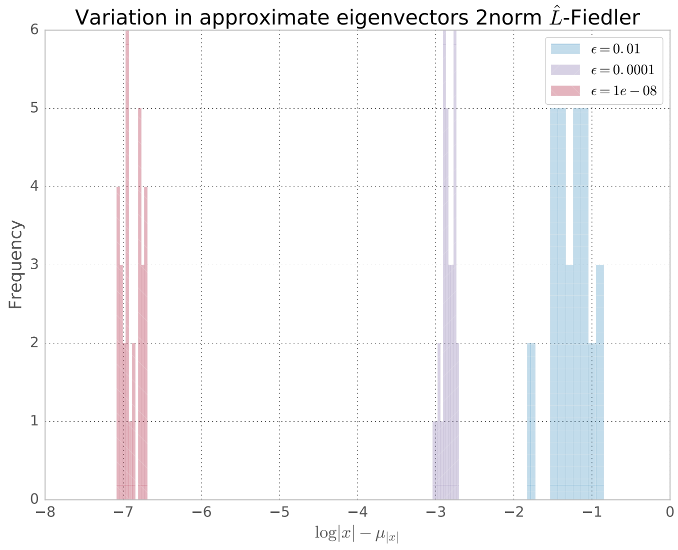


Figure 13: Tighter eigenresidual bounds imply tighter distributions of approximate solutions. Log scale implies that the distributions on the left have smaller variances.

to the eigenvector problem and then rounding. With two balls centered at the same point with different radii $r < R$, the best integer solutions in the larger ball are better than the best integer solutions in the smaller ball. This is a consequence of if $S \subset T$ are sets, then $\min_{x \in S} f(x) \geq \min_{x \in T} f(x)$. The experiments show that in this case, the intuition extends to sampling. Further experimentation shows that this is not an efficient method for accelerating spectral partitioning in general graphs. This work has been published in a poster "Discovering Block Structure in Graphs with Approximate Eigenvectors" at SIAM-CSE 2015.

This example is presented to motivate how numerical analysis can be applied to graph analysis problems to gain insight into the phenomena observed in algorithms. The observation is that low accuracy approximate eigenvectors can sometimes outperform high accuracy approximate eigenvectors for the graph partitioning problem. This phenomenon can be understood by considering the eigensolver as a random process that samples from a distribution over vectors. The remainder of this chapter discusses ways that numerical analysis can be applied to the spectral partitioning problem in order to understand how numerical error affects the quality of the graph partitions.

This work is relevant because we have a large literature on how to approximately solve data analysis problems using the exact solution of numerical problems, and a large literature on approximately solving numerical problems, but we do not have a firm grasp on the relation between accuracy in the numerical solution and the approximation quality of the data analysis solution. This leads to practical difficulties such has how to choose convergence tolerances for eigensolvers when they are used for spectral partitioning. When using numerical solvers for problems such as studying the behavior of systems governed by partial differential equations (PDEs), we know that there is a baseline error that comes from modeling the continuous problem with a finite set of points. This discretization error allows one to use the properties of the PDE and the number of points to determine at what accuracy the solution is close enough to the continuous solution. For data analysis applications of numerical methods, there has not been sufficient research into understanding the appropriate convergence tolerances.

Section 5.4 gives an analysis using Cheeger's inequality that improves the state of the art in choosing convergence tolerances. Keep in mind that the desired accuracy of a numerical solution depends on the context of its application. With the knowledge of the downstream data analysis problem, one can choose numerical solvers with different properties that are advantageous to that data analysis problem.

5.4 Stopping Criteria for Spectral Partitioning

The relationship between numerical accuracy and data mining quality is not thoroughly understood. This section shows that analyzing numerical accuracy and data mining quality together can lead to algorithmic improvements. Specifically, we study spectral partitioning using approximate eigenvectors of the normalized graph Laplacian as described in Algorithm 4. We introduce a novel, theoretically sound, parameter free stopping criterion for iterative eigensolvers designed for graph partitioning. On a corpus of social networks, we validate this stopping criterion by showing the number of iterations is reduced by a factor of 4.15 on average, and the conductance is increased by only a factor of 1.24 on average. Regression analysis of these results shows that the decrease in the number of iterations needed is greater for problems with a small spectral gap, thus our stopping criterion helps more on harder problems. Experiments show that alternative stopping criteria are insufficient to ensure low conductance partitioning on real world networks. While our method guarantees partitions that satisfy the Cheeger Inequality, we find that it typically beats this guarantee on real world graphs.

This work fits into a larger framework of studying how knowledge of the data mining task can shape our choice of numerical procedure. It is already common to see the computer architecture or networking capabilities of a distributed system shape the choice of numerical algorithm [46]. In some applications numerical solutions are used such that errors compound, but in other applications the numerical solution is used in a way that reduces error. For instance in spectral partitioning, one spends many cycles computing high accuracy numerical solutions to an equation only to round the solution vector. In order to make efficient design choices in our algorithms, we must consider the accuracy requirements we face. This chapter shows that for spectral partitioning, the numerical accuracy required is much lower than that which is typically assumed by users of these solvers. The default solver accuracy for Matlab's eigs is machine ϵ times $\|A\|$ which is approximately 10^{-15} in double precision arithmetic, this accuracy is often necessary for many scientific applications, but is not necessary for this particular data mining application. The goal of this chapter is to understand the effect of eigensolver accuracy on partitioning quality. Stopping criterion for iterative methods are an important facet of this relationship. Without a good stopping criterion, an iterative method will either take too few iterations and fail to solve the problem, or take too many iterations and waste computational resources. We introduce a stopping criterion for computing eigenvectors which combines conductance and estimates of numerical error.

Under the standard assumptions on eigensolvers, Section 5.5 analyzes eigenvalue accuracy in the context of spectral partitioning to derive a condition on approximate eigenvectors that provides the same theoretical guarantees as sweep cuts of the exact eigenvectors. Section 5.6 shows this new stopping criterion reduces the number of iterations compared to traditional stopping criteria on real world networks from the Newman and SNAP collections. This provides a parameter free convergence criterion that is theoretically sound and empirically verified. Graphs can have multiple partitions of similar quality and throughout this chapter we assume that an application finds any of them to be sufficient. In this chapter we show that, using this new stopping criterion, we can compute approximate eigenvectors which induce nearly optimal graph partitions. These approximate eigenvectors are computed faster than approximation using the classical stopping criterion based on approximation error. Therefore, we decrease running time while sacrificing little quality. Many of these insights can generalize to other applications where a numerical method solves a data mining problem, such as using personalized PageRank [19] to rank vertices in a graph, commute times [20] to compute a metric distance on the vertices, or the heat equation on a graph [10] to construct low conductance local cuts.

5.4.1 Contributions

Sections 5.5-5.6 make the following contributions:

1. A novel parameter free stopping criterion for spectral partitioning with both

theoretical and experimental support.

2. Demonstrations that alternative stopping criteria are too weak to ensure high quality partitions.

3. Evidence that this method works when restricting to balanced sweep cuts.

4. Guidance on practical choices for the residual tolerance parameter of eigensolvers for this problem.

5.5 Eigenvalue accuracy and Cheeger's inequality

The Cheeger inequality guarantees that exact eigenvectors provide a sweep cut with conductance less than $\sqrt{2\lambda_2}$. The remainder of this section derives a stopping criterion providing the same guarantee for approximate eigenvectors. As one iterates a numerical solver, they can compute the conductance of each iterate. Once a partition with conductance less than $\sqrt{2\lambda_2}$ is found, the solver can stop while satisfying the same guarantee provided by the exact solution. However, when running the solver, the true value of λ_2 is unknown. Thus one cannot use this comparison as a stopping criterion directly. One way to create a stopping criterion is to use μ as an estimate of λ_2 then stop when $\phi(\mathbf{y}) \leq \sqrt{2\mu}$; however, this criteria does not ensure that $\phi(\mathbf{y}) \leq \sqrt{2\lambda_2}$ as $\phi(\mathbf{y})$ can fall between $\sqrt{2\mu}$ and $\sqrt{2\lambda_2}$. Section 5.6.4 discusses further some alternative stopping criteria found to be unreliable for this problem.

To derive a stopping criterion that does guarantee $\phi(\mathbf{y}) \leq \sqrt{2\lambda_2}$, we find a lower bound on $\sqrt{2\lambda_2}$ is sufficient. The lower bound must be computable from information available to the solver. Both the eigenresidual r and the Rayleigh Quotient μ are computable without knowledge of the true eigenvalue and thus form the inputs to the stopping criterion.

Theorem 2. *Let $\hat{L}$ be the normalized graph Laplacian of a connected graph, and $\mathbf{x}$ be a unit vector orthogonal to $D\mathbf{1}$ and $\mu = \mathbf{x}^T\hat{L}\mathbf{x}$ and $\mathbf{y} = D^{-\frac{1}{2}}\mathbf{x}$. If $\mu - \lambda_2 < |\mu - \lambda|$*

for all other eigenvalues λ*, then* $\phi(\mathbf{y}) < \sqrt{2(\mu - r)} = \psi(\mathbf{x})$ *is a stopping criterion that guarantees* $\phi(\mathbf{y}) < \sqrt{2\lambda_2}$*.*

Proof. First the fact that $\sqrt{2\alpha}$ is an increasing function gives, for any positive ϵ, $\mu - \epsilon < \lambda_2$ implies $\sqrt{2(\mu - \epsilon)} < \sqrt{2\lambda_2}$. We show that $r > |\mu - \lambda_2|$ directly. Using the eigendecomposition of $L = Q\Lambda Q^T$, let $\mathbf{z} = Q^T\mathbf{x}$. Since $\mathbf{x} \perp \mathbf{q}_1$, $z_1 = 0$. From the hypothesis that $|\mu - \lambda_2|$ is minimal, one sees

$$r^2 = \|(\Lambda - \mu I)\mathbf{z}\|^2 > (\lambda_2 - \mu)^2 \sum_i z_i{}^2 = (\lambda_2 - \mu)^2.$$

So $r > |\lambda_2 - \mu|$, and $\mu - r < \lambda_2$. Thus under these conditions we know that $\sqrt{2(\mu - r)} < \sqrt{2\lambda_2}$. Since all terms on the left hand side are known to the solver at each iteration, this is a valid stopping criterion. □

This stopping criterion is the first stopping criterion for spectral partitioning that does not require the implementation to specify a chosen parameter value. Unlike prior methods there is no choice of acceptable error that must be considered and no choice of tolerance exposed to the user. This simplifies practical application of this method for graph analysis.

The assumption that μ is closer to λ_2 than to any other eigenvalues implies that ϵ is less than the spectral gap, $\delta = \lambda_2 - \lambda_3$, of the matrix. Thus under the standard assumption that the Ritz value is close enough to the desired eigenvalue, we have a stopping criterion bounding $\phi(\mathbf{y})$.

When the eigenvalue is computed exactly, this bound coincides with the original Cheeger inequality. While this theorem does not imply that further iteration of the eigensolver will not reduce $\phi(\mathbf{y})$, it gives a condition under which it is possible that further iteration will not reduce $\phi(\mathbf{y})$. The vector output will always satisfy the $\sqrt{2\lambda_2}$ guarantee of Theorem 3 as long as the assumptions are true. The experiments in Section 5.6.2 show for many graphs, termination according to this new criterion leads to only a small increase in the conductance of a partition. For some graphs the partition produced by the lower quality approximate has lower conductance than the partition produced by the higher quality approximation.

A valid concern is that ensuring that one satisfies $\sqrt{2\lambda_2}$ is too weak of a guarantee for practitioners. We see that the final partitions are much better than the guarantee. The cycle and the hypercube show that for some graphs the eigenvectors achieve a partition with conductance $\Omega(\sqrt{2\lambda_2})$ [90]. Thus for a method applicable to all graphs, this is the best guarantee possible. However, if one knows the graph in question comes from a family where $\phi(G) \leq f(\lambda)$ where $f(x)$ is an increasing function, then one could apply the technique used in the proof of Theorem 2 by checking if the conductance of the current iterate is less than $f(\mu - r) < f(\lambda)$. Planar Graphs [84] are such a family of graphs. Experiments in Section 5.6 show that this stopping criterion typically gives partitions that outperform the guarantee by at least a factor of $\lambda/5$.

5.6 Experiments

Iterative eigensolvers such as IRAM were developed for solving problems from physics and engineering. Thus they are designed to quickly minimize error and residual. However when using eigenvectors for graph partitioning, minimizing the error to the true eigenvectors is less important than finding an $\mathbf{x}$ minimizing $\phi(\mathbf{x})$. This leads to an experiment showing the conductance of the optimal sweep cut approaches the minimal value before the eigenresidual is small and that our approach returns such a vector with a low conductance.

5.6.1 Experimental Design

While spectral partitioning finds a sweep cut with low conductance, it does not guarantee the minimum possible conductance. Therefore, for our experiments, we compute a baseline eigenvector and partition using the standard approach of iterating until the residual is within an application determined tolerance. We use a tolerance of $\|A\mathbf{x} - \mu\mathbf{x}\| < 10^{-6}$, but stop if the number of iterations reaches 800. Let I_F denote the corresponding number of eigensolver iterations. The conductance of the sweep cut of this approximate eigenvector is our baseline conductance ϕ_F For our experiments, we compare this standard approach using residual tolerance to the stopping criterion from Theorem 2. The restart parameter and maximum number of iterations was chosen manually at 15 to balancing time and memory constraints. For 34 of the graphs, the first time $\phi(\mathbf{x}) < \psi(\mathbf{x})$, hypothesis of Theorem 2 is not satisfied, but by taking one more step the number of such graphs drops to 17. Because stopping at the next iteration after $\phi(\mathbf{x}) < \psi(\mathbf{x})$ has a small impact on the average number of iterations needed and leads to a large decrease in the average conductance that we find, we use this iteration, represented by I_C, in our experiments. ϕ_C represents the conductance of the sweep cut after I_C iterations. We use I_N to represent the first iteration where $\phi(\mathbf{x}) < \sqrt{2\lambda_2}$.

Experiments are conducted on matrices from the Newman [71] and the SNAP [60] collections. These include graphs from co-purchasing, citation, co-authorship, road, autonomous systems, and online networks[2]. Table 3 includes the size of each graph along with the first 3 eigenvalues, which gives a sense of the difficulty of the various problems. From error analysis in [73] we know that when the small eigenvalues of $\hat{L}$ are close together, the eigenvectors are difficult to compute accurately. The problems range from the small and well conditioned N/lesmis to the large and ill conditioned S/web-Google.

For the purpose of experimentation, we compute functions of each iterate[3]. These include the conductance ϕ, the Rayleigh quotient μ, and the residual $r = \|Ax - \mu x\|$. These measurements can be seen in detail in Table 4 for a single graph. We also show the lower bound on the Cheeger bound $\psi(x) = \sqrt{2(\mu - r)}$ from Theorem 2. From

[2]Adjacency matrices are made symmetric by taking $A + A^T$. Because we can find connected components faster than solving the eigenequation, we restrict to the largest connected component of each graph

[3]Requesting the approximate eigenvector at each step also prompts using an alternative implementation of IRAM over the standard ARPACK.

Table 3: Shows the size of each graph along with some eigenvalues. Small values of $\lambda_3 - \lambda_2$ indicate difficult problems. Graphs from the Newman and SNAP collections are abbreviated with N and S, respectively.

name	$\|V\|$	$\|E\|$	λ_2	λ_3	λ_4
N/adjnoun	112	850	0.35604	0.37559	0.39457
N/as-22july06	22963	96872	0.01936	0.02418	0.02621
N/astro-ph	16706	242502	0.00328	0.00383	0.00478
N/celegansneural	297	2345	0.19524	0.25629	0.33204
N/cond-mat	16726	95188	0.00718	0.00729	0.00816
N/cond-mat-2003	31163	240058	0.00427	0.00606	0.00788
N/cond-mat-2005	40421	351382	0.00428	0.00484	0.00616
N/dolphins	62	318	0.03952	0.23435	0.24662
N/football	115	1226	0.13680	0.18292	0.22509
N/hep-th	8361	31502	0.00558	0.00888	0.01010
N/karate	34	156	0.13227	0.28705	0.38731
N/lesmis	77	508	0.08813	0.09222	0.15107
N/netscience	1589	5484	0.00303	0.00850	0.00993
N/polblogs	1490	19025	0.08144	0.10904	0.13685
N/polbooks	105	882	0.03780	0.17589	0.24433
N/power	4941	13188	0.00027	0.00055	0.00043
S/amazon0302	262111	1234877	0.00029	0.00083	0.00135
S/amazon0312	400727	3200440	0.00425	0.00361	0.00153
S/amazon0505	410236	3356824	0.00070	0.00078	0.00085
S/amazon0601	403394	3387388	0.00036	0.00085	0.00088
S/as-735	7716	26467	0.03349	0.04254	0.04635
S/as-caida	31379	106762	0.01120	0.01826	0.01939
S/ca-AstroPh	18772	396160	0.00629	0.01038	0.01706
S/ca-CondMat	23133	186936	0.00719	0.00803	0.01622
S/ca-GrQc	5242	28980	0.00187	0.00206	0.00367
S/ca-HepPh	12008	237010	0.00178	0.00400	0.01009
S/ca-HepTh	9877	51971	0.00312	0.00718	0.00801
S/cit-HepPh	34546	421578	0.01572	0.02300	0.03220
S/cit-HepTh	27770	352807	0.01839	0.02391	0.02598
S/email-Enron	36692	367662	0.00353	0.00454	0.00668
S/email-EuAll	265214	420045	0.00010	0.00213	0.00234
S/Oregon-1	11492	46818	0.03290	0.04820	0.04958
S/Oregon-2	11806	65460	0.02919	0.04191	0.04551
S/p2p-Gnutella04	10879	39994	0.02189	0.08147	0.17245
S/p2p-Gnutella05	8846	31839	0.10959	0.14601	0.15297
S/p2p-Gnutella06	8717	31525	0.12074	0.13797	0.17567
S/p2p-Gnutella08	6301	20777	0.04037	0.08103	0.09628
S/p2p-Gnutella09	8114	26013	0.03515	0.08965	0.12833
S/p2p-Gnutella24	26518	65369	0.06600	0.09155	0.11575
S/p2p-Gnutella25	22687	54705	0.02837	0.04100	0.06361
S/p2p-Gnutella30	36682	88328	0.06124	0.08551	0.09075
S/p2p-Gnutella31	62586	147892	0.05989	0.06141	0.06811
S/soc-Epinions1	75888	508837	0.00479	0.01323	0.01573
S/soc-sign-epinions	131828	841372	0.01123	0.01575	0.01717
S/soc-sign-Slashdot081106	77357	516575	0.02105	0.02175	0.03069
S/soc-sign-Slashdot090216	81871	545671	0.01723	0.01769	0.03069
S/soc-sign-Slashdot090221	82144	549202	0.01723	0.01769	0.03069
S/soc-Slashdot0811	77360	905468	0.01247	0.01529	0.01772
S/soc-Slashdot0902	82168	948464	0.01174	0.01321	0.01772
S/web-Google	916428	5105039	0.00043	0.00044	0.00064
S/web-NotreDame	325729	1497134	0.00248	0.00182	0.00134
S/web-Stanford	281903	2312497	0.00002	0.00008	0.00016
S/wiki-Talk	2394385	5021410	0.01680	0.01941	0.02410
S/wiki-Vote	8297	103689	0.10055	0.16859	0.24277

Theorem 2 we know that if $\phi < \psi$ then $\phi < \sqrt{2\lambda}$, which means that our partition is within the guarantee provided by the Cheeger inequality. A dash in Table 4 indicates $\mu - r < 0$ and thus the stopping criteria cannot be satisfied.

Because $\hat{L} = I - \hat{A}$, the eigenvalues of $\hat{L}$ are the same as one minus the eigenvalues of A with the same eigenvectors. Thus we can solve for the largest eigenvalues of $\hat{A}$ and then compute the corresponding smallest eigenvalues of $\hat{L}$. This allows us to avoid the need for a linear solver for $\hat{L}x = b$. We can also work with vectors orthogonal to $\mathbf{q}_1 = D^{-1/2}\mathbf{1}\left\|D^{-1/2}\mathbf{1}\right\|^{-1}$ by iterating with the linear operator $M = \hat{A} - \mathbf{q}_1\mathbf{q}_1^T$. The desired eigenvalue and eigenvector of the Laplacian correspond to the largest eigenvalue of M. This improves the performance of the solver while allowing us to compute the appropriate Laplacian eigenvalues and eigenvectors.

5.6.2 Summary of Results

Table 5 compares, for all graphs, our stopping criterion to the standard criterion of iterating until $\|r\| < 10^{-6}$. Values for each graph are averaged across 10 runs, with the last row showing the average across all graphs. The values in Table 5 differ from those seen in Figure 16 because the latter shows results from individual runs. We can see that stopping at I_C results in a conductance less than five times the final conductance for every graph. On average across all graphs the conductance resulting from our approach is only 1.24 times greater and only 0.24 times as many iterations are needed or a reduction by a factor of 4.15. This is the primary result of the experiments. This stopping criteria reduces the number of iterations significantly without creating a large loss in the achieved conductance.

Table 4 shows a sample of iterations in detail for the S/web-Google graph. Although we show such detailed results for only one graph, these observations apply to many of the large graphs seen in the experiment. Table 4 also shows that the approximate eigenvalue μ is monotonically decreasing, but the minimal conductance of a

Table 4: Some iterates of the solver are shown in detail for the graph S/web-Google. Because Theorem 2 requires that $\mu - r \geq 0$, ψ is not supplied when this condition is not met. We find that $\mu - r > 0$ is a good proxy for the stopping criterion $\phi \geq \psi$, which implies that one can use this test in order to reduce the number of times that ϕ must be computed.

iter	ϕ	μ	$\sqrt{2\mu}$	ψ	r
20	0.02283	0.00881	0.13276	–	0.01900
30	0.00571	0.00445	0.09435	–	0.01110
40	0.00399	0.00277	0.07438	–	0.00755
50	0.00297	0.00193	0.06209	–	0.00584
60	0.00203	0.00144	0.05372	–	0.00423
70	0.00153	0.00117	0.04837	–	0.00347
80	0.00143	0.00101	0.04493	–	0.00267
90	0.00100	0.00090	0.04248	–	0.00234
110	0.00100	0.00074	0.03859	–	0.00194
120	0.00062	0.00069	0.03703	–	0.00169
130	0.00062	0.00064	0.03579	–	0.00158
140	0.00062	0.00061	0.03483	–	0.00142
160	0.00062	0.00056	0.03339	–	0.00120
170	0.00071	0.00054	0.03288	–	0.00111
250	0.00071	0.00046	0.03030	–	0.00085
260	0.00083	0.00045	0.03006	–	0.00079
620	0.00083	0.00041	0.02878	0.02554	0.00009
630	0.00083	0.00041	0.02877	0.02552	0.00009

sweep cut is not. We can see many iterations with $\mu - r < 0$, and few iterations with $\mu - r > 0$ and $\phi < \psi$ so one can check $\mu - r > 0$ as a preliminary stopping criterion that is faster to evaluate than ϕ. This can reduce the number of times conductance must be evaluated.

This improvement can be summarized with the distribution of the reduction in iterations and the increase in conductance. The goal of choosing a good stopping criteria for an optimization routine is to keep the number of iterations low and the quality of the results high. Figures 14 and 15 show that for most graphs the number

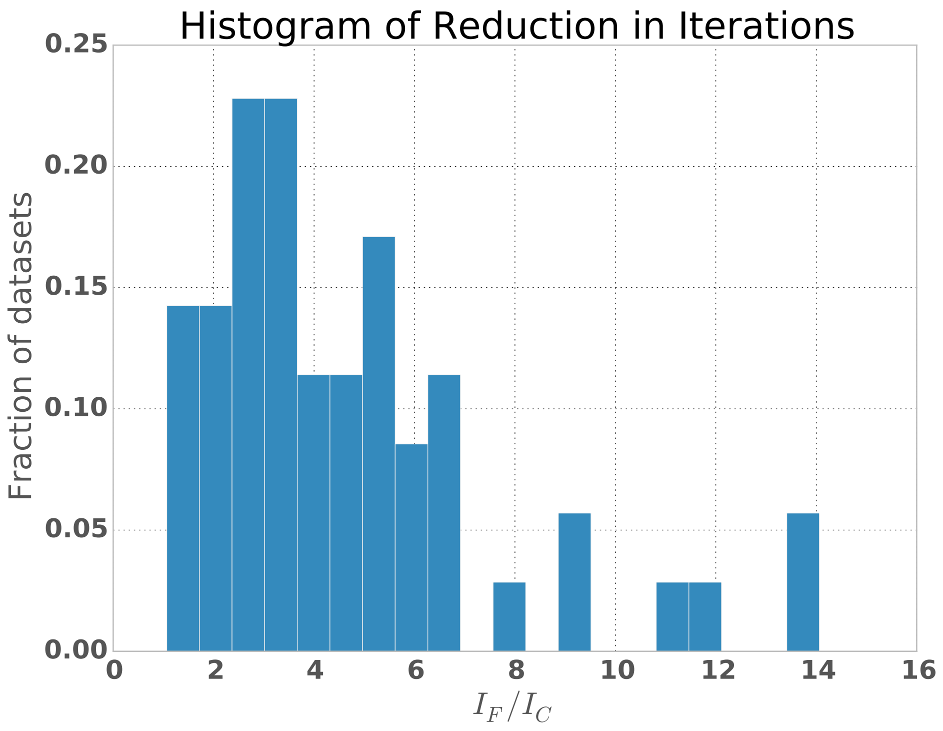


Figure 14: The distribution of iteration reduction over the experiments.

of iterations decreases by a large factor, more than 2X, and the conductance does not increase by a large factor. The median increase in conductance is less than 2X. From this empirical evaluation, we conclude that the effects of using this stopping criterion are a dramatic improvement in performance with a slight degradation of quality. This trade-off will not satisfy all applications, but for the community detection application

Table 5: Improvement shown for each graph averaged over 10 runs. The columns are the graph name, iterations until $\phi < \psi$, iterations until $r < tol$, ratio of $\phi\left(\mathbf{x}^{I_C}\right)$ to $\phi\left(\mathbf{x}^{I_F}\right)$, the conductance found when $r < tol$, and the residual upon early termination. This table shows the graphs where our convergence criterion was reached within the maximum number of iterations.

Graph	I_C	I_F	$\frac{\Phi_C}{\Phi_F}$	Φ_F	Residual
N/adjnoun	10.0	42.0	1.0515	0.4615	0.0407
N/as-22july06	17.0	132.0	1.7490	0.0298	0.0363
N/astro-ph	54.5	315.5	1.2273	0.0046	0.0031
N/celegansneural	10.0	24.5	1.1272	0.2258	0.0345
N/cond-mat	43.0	487.5	0.9184	0.0152	0.0072
N/cond-mat-2003	67.0	225.5	1.0948	0.0064	0.0041
N/cond-mat-2005	59.0	329.5	1.1051	0.0064	0.0041
N/dolphins	14.5	20.0	1.0000	0.0682	0.0011
N/football	10.0	29.5	1.1107	0.1207	0.0254
N/hep-th	46.0	187.0	1.0331	0.0250	0.0067
N/karate	10.0	15.0	1.0000	0.1515	0.0049
N/lesmis	11.5	31.0	0.9916	0.1526	0.0187
N/netscience	46.5	108.5	1.1571	0.0048	0.0022
N/polblogs	12.0	30.0	0.7371	0.1250	0.0220
N/polbooks	14.5	20.0	0.9973	0.0476	0.0027
N/power	236.5	785.5	0.9240	0.0025	0.0003
S/Oregon-1	14.5	83.5	1.3502	0.0685	0.0372
S/Oregon-2	14.5	87.5	1.3659	0.0489	0.0379
S/amazon0302	233.5	493.0	1.0324	0.0008	0.0003
S/amazon0312	103.0	800.0	1.2125	0.0018	0.0017
S/amazon0505	147.0	800.0	1.0702	0.0011	0.0007
S/amazon0601	229.5	641.5	1.0724	0.0006	0.0004
S/as-735	11.5	95.0	1.3674	0.0651	0.0531
S/as-caida	30.5	115.5	1.0527	0.0302	0.0121
S/ca-AstroPh	37.5	124.5	1.1896	0.0102	0.0069
S/ca-CondMat	41.0	155.0	1.1845	0.0109	0.0063
S/ca-GrQc	60.0	199.0	1.4333	0.0025	0.0028
S/ca-HepPh	62.0	138.5	1.2651	0.0024	0.0019
S/ca-HepTh	58.5	168.5	1.7388	0.0036	0.0029
S/cit-HepPh	21.5	89.0	1.3945	0.0357	0.0250
S/cit-HepTh	18.5	143.0	1.6112	0.0312	0.0267
S/email-Enron	58.0	188.0	1.0037	0.0045	0.0029
S/email-EuAll	158.0	259.0	1.0000	0.0001	0.0001
S/p2p-Gnutella04	17.5	33.0	1.0000	0.0455	0.0124
S/p2p-Gnutella05	10.0	50.5	1.2796	0.1500	0.0857
S/p2p-Gnutella06	10.0	49.0	1.0991	0.2222	0.0947
S/p2p-Gnutella08	12.5	46.5	1.2493	0.0500	0.0486
S/p2p-Gnutella09	15.0	39.0	1.0000	0.0429	0.0205
S/p2p-Gnutella24	10.0	55.5	1.7662	0.1000	0.0875
S/p2p-Gnutella25	13.5	61.0	1.5241	0.0435	0.0511
S/p2p-Gnutella30	10.0	70.0	1.3673	0.1000	0.0762
S/p2p-Gnutella31	10.0	142.0	1.0657	0.1111	0.0878
S/soc-Epinions1	49.5	122.0	1.2661	0.0061	0.0050
S/soc-Slashdot0811	18.5	143.5	2.2980	0.0135	0.0305
S/soc-Slashdot0902	12.5	155.5	4.0594	0.0119	0.0477
S/soc-sign-Slashdot081106	21.5	131.0	1.8164	0.0238	0.0295
S/soc-sign-Slashdot090216	25.0	124.5	1.5721	0.0192	0.0178
S/soc-sign-Slashdot090221	25.5	121.0	1.2030	0.0192	0.0157
S/soc-sign-epinions	32.0	155.0	1.5445	0.0138	0.0132
S/web-Google	278.0	800.0	0.9728	0.0007	0.0004
S/wiki-Talk	13.5	156.5	2.0337	0.0321	0.0650
S/wiki-Vote	10.0	30.5	1.0000	0.1250	0.0580
Average	49.1	189.4	1.2432	0.0557	0.0246

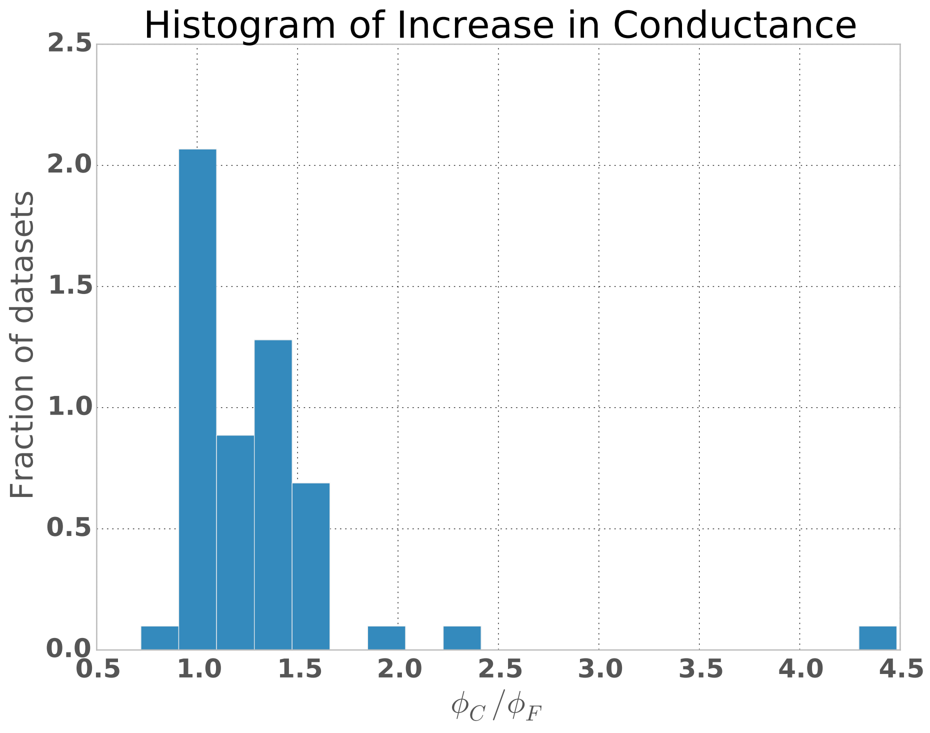


Figure 15: The distribution of conductance increases over the experiments.

on large social networks the improvement in speed is worth making this trade-off, especially when the community detection is part of the streaming framework described in Chapter 4.

5.6.3 Effect of Eigengap

It is not sufficient to check that a method works on some fixed set of inputs. In order to know that an improvement will generalize one must understand the factors that affect its efficacy. Regression analysis on the experimental results delivers understanding into the problem factors that affect the effectiveness of this new stopping criterion.

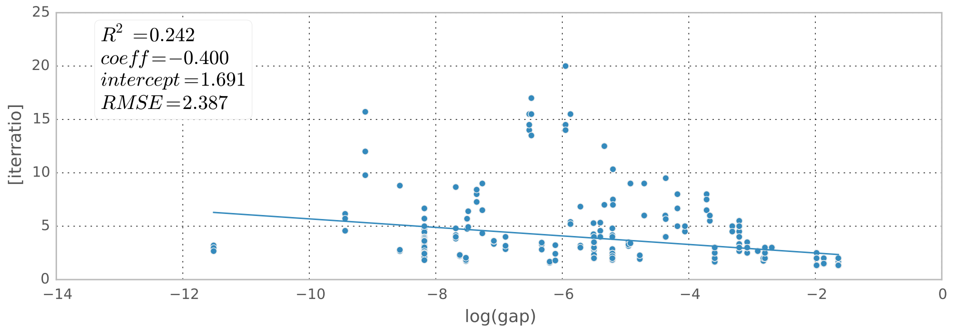
Standard error analysis says that that $e < \sqrt{2}\frac{r}{\delta}$ [73] as can be seen in Section 6.2. This error bound says that the difficulty of an eigenproblem is controlled by the eigengap δ. For a fixed error the residual necessary to guarantee that error is linear in the eigengap $\delta = |\lambda_2 - \lambda_3|$. Thus we focus on the dependence of our method on the eigengap δ. When predicting the iteration ratio $\frac{I_C}{I_F}$ as a function of the rows, nnz, gap,

I_f, and λ_2 only the coefficients of gap, and λ_2 are significant with $R^2 = 0.3173$. These coefficient indicate that as the problem gets harder, the we save more iterations. This is taken over 851 samples and is significant at $p < 0.001$. When predicting $\frac{\phi_F}{\phi_C}$ as a function of the same variables the $R^2 < 0.1$ indicating that there is no good linear fit. None of the coefficients are significant given the same thresholds as above. This indicates that there is no reason to believe that this method breaks down on harder problems. Regression analysis of iterratio, $\frac{I_F}{I_C}$, and phiratio, $\frac{\phi_C}{\phi_F}$, as a function of gap, δ is shown in Figure 16. One can see that there is a linear relationship between $\log \delta$ and $\frac{I_F}{I_C}$ showing that as the problem gets harder our stopping criterion saves more iterations. Based on the lack of a linear relationship between $\log \delta$ and $\frac{\phi_C}{\phi_F}$, there is no evidence to suggest that this method loses approximation quality on harder problems.

5.6.4 Alternative Stopping Criteria

A stopping criterion based on the General Cheeger Inequality can be derived by substituting an estimate of λ_2 into the inequality. Choosing μ as the estimate yields the criterion $\phi(\mathbf{x}) < \sqrt{2\mu}$. This test does not consider the accuracy of the estimate. We can see from Table 4 that this criterion is satisfied too early to ensure a close approximation to the value of ϕ given by a close approximation to the true eigenvector. Once one considers a rigorous lower bound on λ_2 instead of an arbitrary approximation, one derives a stopping criterion that works.

An alternative approach to stopping criteria comes from examining changes in the objective function. A common approach in data mining is to stop when the objective function does not decrease from one iteration to the next. This comes from an idea that changes in the objective function are monotonically decreasing in magnitude. This strategy works for some solvers and thus appears to be a good convergence criterion. However in this case $\phi(\mathbf{x}^i)$ can stall. More precisely, for some



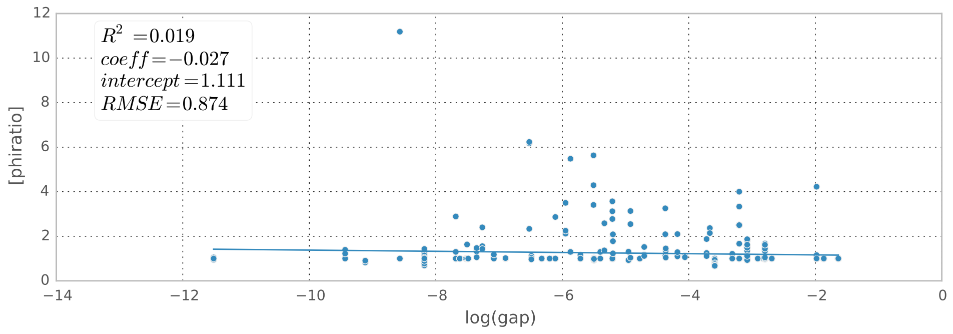


Figure 16: Results for individual runs on each graph, not averaged, are plotted. The iteration savings $\frac{I_F}{I_C}$, shown on top, increase as the gap decreases and the loss in conductance $\frac{\phi_C}{\phi_F}$, shown on bottom, does not increase as the gap decreases.

iteration i, $\phi(\mathbf{x}^i) - \phi(\mathbf{x}^{i+1}) = 0$ while there exists a j much larger than i such that $\phi(\mathbf{x}^j) < \phi(\mathbf{x}^i)$. Therefore, stopping when the conductance stops improving yields large loss in conductance. This occurs for ill conditioned problems such as the Web-Stanford graph as is illustrated in Figure 17.

5.6.5 Balance

The theorems used in this paper do not guarantee the existence of balanced cuts. Balanced cuts are favored in applications. One way to get a multi-way partition is to recursively construct 2-way partitions until the parts are sufficiently small. If the smaller part of the 2-way partition contains more than pn vertices for large enough $p < 1/2$, this creates a balanced, low depth recursion tree. In order to understand

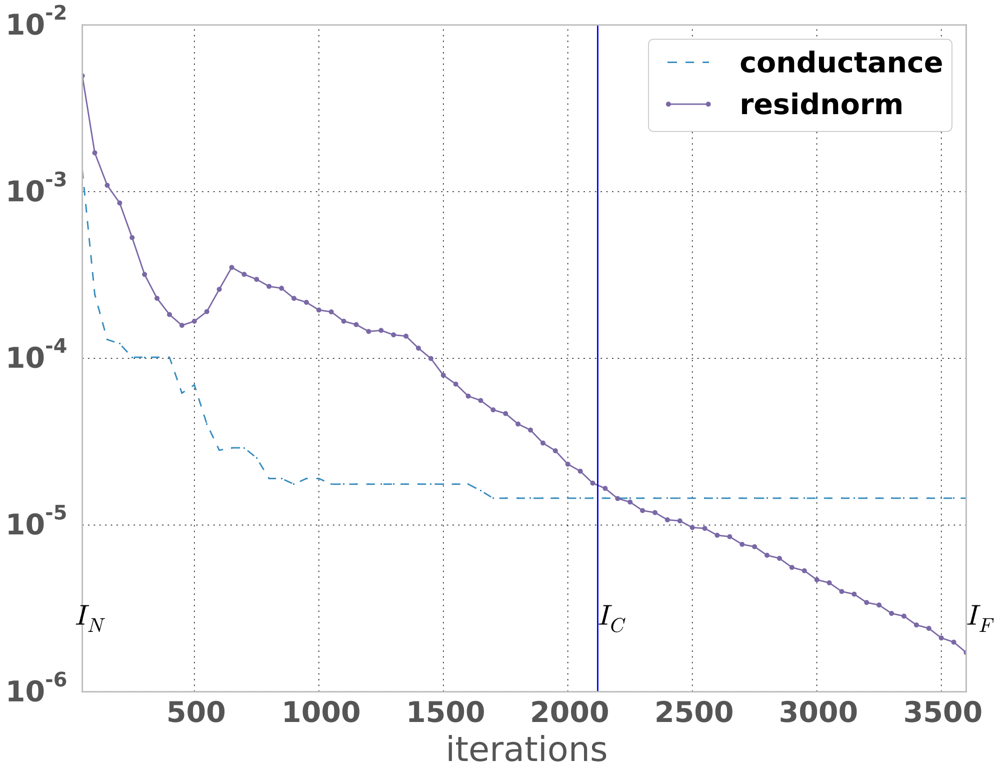


Figure 17: The decrease in conductance and residual for SNAP/web-Notre-Dame is shown. The flat regions of the conductance curve indicate where one would stop based on $\phi\left(\mathbf{x}^i\right) - \phi\left(\mathbf{x}^{i+1}\right) = 0$. This shows that neither stopping at I_N iterations nor when the objective function stops decreasing achieves low conductance partitions for this problem, even though both satisfy $\phi\left(\mathbf{x}\right) < \sqrt{2\lambda_2}$. Using our criterion and stopping at I_C iterations does achieve a low conductance cut.

the effects of our stopping criterion on choosing a balanced cut we make a similar comparison as Section 5.6.2 while restricting to balanced cuts. We introduce the notation ϕ^p as the best sweep cut which contains at least p percent of the vertices on each side. Table 6 shows the minimum, maximum, and mean taken over all graphs, of the best sweep cut of the final iterate ϕ_F^0, the best balanced sweep cut of the final iterate ϕ_F^{10}, and the ratio between the conductance of the best balanced sweep cut of the I_C-th iterate ϕ_C^{10}. The value of ϕ_C^0 and ϕ_F^0 for each graph can be found as ϕ_C and ϕ_F in Table 5. For balanced cuts, the increase in conductance due to our stopping criterion is on average less than a factor of 1.3 and for over half the graphs the same conductance is found, which suggests that our methods works well in practice even with a balance condition. In contrast, for half the graphs, the ratio of conductance between the best cut and the best balanced cut for the final iteration is at least 4.5. In other words, the cost of restricting to a balanced cut exceeds the cost due to the novel stopping criterion.

Table 6: The effect of restricting to balanced cuts is shown. The max, mean, and min are taken over all graphs. ϕ_F^{10} represents the conductance of the best cut of the final computed eigenvector with at least $\frac{n}{10}$ vertices on each side. ϕ_C^{10} represents the conductance of the best balanced cut achieved with our stopping criterion.

	ϕ_F^0	ϕ_F^{10}	$\frac{\phi_C^{10}}{\phi_F^{10}}$	$\frac{\phi_F^{10}}{\phi_F^0}$
min	0.0001	0.0017	0.4111	1.0000
median	0.0455	0.2080	1.0000	4.5926
mean	0.0736	0.2732	1.2557	36.829
max	0.6133	0.8245	5.0514	1001

5.7 Conclusions

This work fits into a larger context of understanding the connection between numerical accuracy of solvers and data mining quality. We measure the solver accuracy using the norm of the residual vector and the data mining quality as the conductance of the resulting partition. We show that by careful analysis of both measures, one can rigorously derive improved parameter free stopping criterion. This stopping criterion is empirically validated on real world networks. The result of our careful analysis is a large reduction in the number of iterations used to solve this data mining problem with iterative methods. This leads to faster methods for large problems. Understanding the relationship between the numerical method and the data mining method leads to an algorithmic improvement. This improvement is shown on real world networks and is realized in a state of the art solver. We also demonstrate empirically that simpler convergence criteria based on intuition do not achieve factor of two approximations to the prior work.

This paper draws the following quantitative conclusions.

1. Our stopping criterion for spectral partitioning leads to a 4.15 fold decrease in iteration with only a 1.24 fold increase in resulting conductance.

2. A practical choice for the residual tolerance parameter of eigensolvers for low conductance partitioning is 10^{-4}.

3. Alternative stopping criteria are too weak to ensure high quality solutions, as is shown on the S/web-Google graph.

4. When imposing a balance condition of 10% on the cuts, stopping using our criterion increases the conductance by a factor of 1.4 on average compared to using the high fidelity eigenvectors.

5. Analyzing the performance of our method as a function of spectral graph indicates that our method reduces cost more on harder problems.

By analyzing the numerical accuracy of iterative methods along with the data mining objective function, we are able to gain new insights. This opens new questions about the relationship to preconditioning, multiway cuts, and higher dimensional techniques. The conclusions in this work should be robust to improvements in preconditioning. The guarantees of our stopping criteria are valid and still decrease the number of iterations. When developing preconditioners for these problems, one must be aware of the effect on data mining accuracy. The effect of this method on hierarchical spectral partitioning is an open question.

In order to apply the proof of $\psi \leq \sqrt{2\lambda}$, one must ensure $|\mu - \lambda_1| = \min_i |\mu - \lambda_i|$. While we show empirically that stopping when $\phi < \psi$ provides low conductance cuts, the proof used in the guarantee does not apply without that hypothesis. For some difficult problems this condition is not ensured by the IRAM solver. Further study of approximating arbitrary linear combinations of the low energy eigenvectors is necessary to improve these techniques.

CHAPTER VI

SPECTRAL PARTITIONING WITH BLENDS OF EIGENVECTORS

In order to understand the numerical aspects of spectral partitioning, we examine the convergence of iterative solvers to invariant subspaces rather than single eigenvectors. Experiments with the stochastic block model from Section 5.2 show that linear combinations of the leading eigenvectors lead to good partitions. Section 6.2 proves error bounds for invariant subspaces, and Section 6.3 analyzes a model problem to show that when the graph has the right structure, we can find large residual tolerances that lead to correct partitions.

Spectral partitioning covers a broad class of methods that use eigenvectors and eigenvalues to partition graphs. Convergence analysis of eigensolvers uses the distribution of energy in the Fourier basis of the matrix in order to quantify the convergence rate and conditioning of the problem. In the case of ill conditioned problems, the energy is spread among several eigenvalues close together. Blends of eigenvectors (linear combinations of eigenvectors with energy concentrated on only a subset of eigenvectors) model the output of eigensolvers. By treating blends of eigenvectors as the target of the computation and not a side effect of the approximation, we show better error bounds for their computation. For a model graph partitioning problem, we are able to prove the utility of computing spectral blends. These ideas are derived from an analysis of the ring of cliques. This analysis leads to evidence that as the number of parts increases, the partitioning problem becomes harder, and as the size of the parts increases, the problem is easier, although the run-times still increase in problem size. Localization in eigenvectors is used to show that a blend of top eigenvectors partition the ring of cliques graphs.

We provide new results connecting data analysis error to numerical accuracy in the context of spectral graph partitioning. We provide pointwise convergence guarantees so that spectral blends (linear combinations of eigenvectors) can be employed to solve data analysis problems with confidence in their accuracy. We apply this theory to an accessible model problem, the ring of cliques, by deriving the relevant eigenpairs and finding necessary and sufficient solver tolerances. Analysis of the ring of cliques provides an upper bound on eigensolver tolerances for graph partitioning problems. These results bridge the gap between linear algebra based data analysis methods and the convergence theory of iterative approximation methods. These results explain how the combinatorial structure of a problem can be recovered much faster than numerically accurate solutions to the associated linear algebra problem.

6.1 Introduction

Spectral methods are a valuable tool for finding cluster structure in data. While all spectral methods rely on approximating the eigenvectors of a matrix, the impact of numerical accuracy on the quality of the partitions is not fully understood. Spectral partitioning methods proceed in two steps, first one or more vectors approximating eigenvectors of a graph matrix are computed, and then a partitioning scheme is applied to those vectors. While many theoretical results quantify the relationship between the exact solution to the numerical problem and the solution to the original data mining problem, few address data analysis errors introduced by error in the numerical solution. For instance, [48] studies the runtime and quality (in terms of conductance) of partitioning algorithms including spectral methods. Often the eigenvector computation is used as a primitive operation without accounting for the trade-off between run time and numerical accuracy. Guattery and Miller [37] studies various methods of applying exact eigenvectors to partition graphs by producing examples where each variation does not find the optimal cut. This chapter addresses the effect of numerical error in the eigenvector computation on the quality of sweep cuts which reveal graph structure.

In order to understand the impact of numerical error on spectral partitioning, we study both general matrices and a specific family of graphs. Finding error and residual tolerances for general graphs is a difficult problem. Section 6.2 provides tools for deriving a residual tolerance for arbitrary graphs. Section 6.3 analyzes a model problem with clear cluster structure, where linear combinations of eigenvectors represent a space of multiple good partitions, and applies Section 6.2 results to derive a residual tolerance sufficient for solving this model problem. This use of a model problem is well established in the linear algebra literature where the Laplace equation on a regular grid is common in papers and software regarding the solution of systems of equations. Analysis of this model problem allows us to derive a solver tolerance for correctly recovering the clusters with a sweep cut scheme. This analysis illustrates the difference between accurately solving the equation and correctly recovering the combinatorial structure.

This approach to approximate eigenvectors can be applied to other applications where a numerical method solves a data mining problem, such as solving personalized Pagerank as a linear system [19] to rank vertices in a graph, or evaluating commute times [20] to produce a metric distance on the vertices. These methods also apply numerical solvers to infer a combinatorial or data analysis structure from the graph. A similar treatment, in terms of a model problem, of these methods would benefit our understanding of the relationship between numerical accuracy and data analysis accuracy.

Here we introduce the necessary concepts of data analysis quality and eigensolver accuracy. For this work we focus on partitioning graphs to minimize conductance as defined in Equation 3. For any vector $\mathbf{x}$, represent the sweep cut of $\mathbf{x}$ at t as in Equation (21).

$$S = S_{\mathbf{x}}(t) = \{i \mid x_i > t\} \tag{21}$$

We denote by $\phi(\mathbf{x})$ the minimal conductance of a sweep cut of $\mathbf{x}$, that is $\min_t \phi(S_x(t))$. This chapter treats deals with applications that can accept any partition with conductance less than a prescribed value ψ.

The accuracy of a solution to the eigenvector problem can be measured in three quantities: Rayleigh quotient, error, and residual. Spectral methods for finding low-conductance partitions rely on computing vectors $\mathbf{x}$ and corresponding scalars λ that solve the equations $\hat{L}\mathbf{x} = \lambda\mathbf{x}$ for some graph-associated matrix $\hat{L}$. The Rayleigh quotient, $\mu = \mathbf{x}^T\hat{L}\mathbf{x}$ is an approximation to the eigenvalue λ. The error $\|\mathbf{v} - \mathbf{x}\|$, where $\mathbf{v}$ is the closest exact solution, is not accessible to a solver in general. The solver can use the norm of the eigenresidual, $\left\|\hat{L}\mathbf{x} - \mu\mathbf{x}\right\|$, to determine when to stop iterations. For practical application of an eigensolver one must choose a residual tolerance $\left\|\hat{L}\mathbf{x} - \mu\mathbf{x}\right\| < \epsilon$ small enough to ensure that the computed eigenvector is accurate enough to solve the application problem. This chapter provides concrete residual tolerances for a specific model problem and provides tools for finding such tolerances for more general graphs.

This chapter uses the normalized adjacency matrix $D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = \hat{A}$ as defined in Table 1. We apply the identity $\lambda_k(\hat{L}) = 1 - \lambda_{n-k}(\hat{A})$ to replace computations involving small eigenvalues of the normalized Laplacian matrix with computations involving large eigenvalues of the adjacency matrix.

Conductance is an appropriate measure of partition quality for spectral partitioning because of Cheeger's inequality Theorem 3 which bounds the conductance of the graph in terms of the eigenvalues of the Laplacian matrix. A more general form of Cheeger's inequality is proven in [67].

Theorem 3. *General Cheeger Inequality [67] If* $\mathbf{x}$ *is a unit vector orthogonal to* $D^{\frac{1}{2}}\mathbf{1}$

such that $\mathbf{x}^T \hat{L} \mathbf{x} = \mu$ *then* $D^{-\frac{1}{2}}\mathbf{x}$ *has a sweep cut* S *such that* $\phi(S) = \phi(\mathbf{x}) \leq \sqrt{2\mu}$*.*

When $\mathbf{x}$ satisfies $\hat{L}\mathbf{x} = \lambda_2 \mathbf{x}$, $\phi_G \leq \phi\left(D^{-\frac{1}{2}}\mathbf{x}\right) \leq \sqrt{2\lambda_2}$. This general form of Cheeger's inequality indicates that finding low-energy Laplacian eigenvectors is sufficient for constructing low-conductance partitions of the graph.

In graph partitioning, the goal is to compute a partition of the graph that optimizes the chosen objective. When applying spectral methods to graph partitioning, our goal is not to compute very accurate eigenpairs, but instead to partition the vertex set of a graph correctly. Our results on the model problem indicate that approximate eigenvectors are sufficient to solve the data analysis problem and are much faster to compute if the graph has the right structure.

6.1.1 A Model Problem

We use a simple model (the ring of cliques) to study the capabilities of spectral partitioning algorithms, form theory to characterize performance, and potentially enhance these algorithms. Such use of model problems is well-established in the numerical analysis literature regarding iterative solutions to discretized partial differential equations (PDEs). The Dirichlet Laplacian on a unit square discretized on a Cartesian lattice is a simple problem with known eigenpairs and is used to study the properties of various eigensolvers. These simple model problems do not demonstrate the algorithms perform well on real-world problems, but are incredibly important tools for algorithm development and theoretical analysis. For spectral partitioning, the ring of cliques is one candidate model problem for which we can derive complete knowledge of the eigenpairs. In PDEs, the order of discretization error (difference between continuous solution and discrete solution) provides the solver with a stopping criterion. In spectral graph partitioning, we do not have this luxury, and we must develop theory to understand how perturbations in a spectral embedding impact partitioning quality. Another reason to develop a collection of model problems is to enable careful study of this impact in well-understood situations.

In order to provide a striking example of our improved analysis, Section 6.3 studies our model problem in detail. The goal is to understand when approximate eigenvectors have sweep cuts that correctly identify graph structures. The ring of cliques has been studied as "the most modular network" in order to demonstrate a resolution limit in the modularity maximization procedure for community detection [30]. For this family of highly structured graphs, the correct partition is unambiguous. We use the ring of cliques to investigate how spectral embeddings for partitioning are affected by numerical error. Because of the high degree of symmetry, the ring of cliques allows for a thorough closed form analysis producing formulas for the eigenvectors and eigenvalues. A sweep cut using exact eigenvectors partitions the graph with small conductance and successful recovery of all the clusters. We quantify the effects of approximation error on sweep cut partitions of this graph. Our findings demonstrate that despite a small spectral gap, which implies slow convergence of non-preconditioned eigensolvers, the ring of cliques is well partitioned by low accuracy approximations to the eigenvectors.

Studying the ring of cliques provides guidance for practitioners on useful tolerances for eigensolvers. We are able to construct the smallest perturbation that induces a mistake in the sweep cut partition. This perturbation shows that when looking for clusters of size b in a general graph the eigensolver tolerance must be smaller than $\mathcal{O}\left(b^{-\frac{1}{2}}\right)$. Analysis of the ring of cliques provides an upper bound on the eigensolver accuracy that is sufficient to recover community structure.

6.1.2 Contributions

This chapter provides the following contributions. Section 6.2 extends a known error bound on the computation of eigenvectors to the computation of linear combinations of eigenvectors. By extending this classical error bound to linear combinations of eigenvectors, we find a condition on the spectrum of where numerically accurate blends are easy to achieve. Theorem 6 provides a general condition under which approximate eigenvectors preserve sweep cuts. Section 6.3 analyzes a model problem and derives necessary and sufficient error tolerances for solving the model problem, which are essentially tight for some parameter regime. We show for the model problem where the number of clusters is polynomial in the size of the clusters, the power method takes $\mathcal{O}(1)$ iterations to identify the clusters.

6.1.3 Related Work

Iterative methods [58, 79] have been shown to provide fast approximate solutions for a wide range of problems. Many iterative eigensolvers can be represented as $\mathbf{y} = p(M)\mathbf{x}$ where p is a polynomial applied to the matrix M times a vector. The degree of p depends on the number of iterations of the method, which is controlled by the eigenresidual tolerance $\|M\mathbf{y} - \mu\mathbf{y}\| < \epsilon$. The simplest such method is the power method (Algorithm 1), which is easy to analyze because $p(M)$ is always M^k where k is the number of iterations. More sophisticated methods choose $p(M)$ adaptively and typically converge more quickly. A practical implementation of the Arnoldi method can be found in [58], which is commonly used in practice.

Localized eigenvectors are essential to analysis of the ring of cliques. Cucuringu and Mahoney examine the network analysis implications of localized interior eigenvectors in the spectrum of the co-voting network of US Senators [14]. The graph is defined with connections between members of the same session of Congress who vote together on the same bills and connections between individuals who are reelected to consecutive sessions. The first 41 eigenvectors are oscillatory across the congressional sessions with little variation between the vertices in the same session, but the next eigenvectors are small in magnitude on most sessions, but take large positive values on members of one party and large negative values on members of the other party within a few sessions. Thus blends of the dominant eigenvectors indicate the sessions of Congress. The ring of cliques also exhibits globally periodic extremal eigenvectors and localized interior eigenvectors due to its Kronecker product structure. We show that the ring of cliques has a basis for an interior eigenspace with the nonzero entries of each vector completely restricted to an individual clique. This localization allows us to show that approximate eigenvectors recover the interesting combinatorial structure.

Other work focuses on the impact of errors in measurement on the behavior of data analysis algorithms. In the context of Gram (kernel) matrices, Huang et al. [44], studies the effect of perturbing the original data points on the spectral partitioning method. A similar line of investigation is pursued in [98], where data points are quantized to reduce bandwidth in a distributed system. This work connects approximation with performance. If one can demonstrate that data analysis accuracy is not affected too much, then one can use an algorithm which sacrifices accuracy to improve performance. Our paper treats the data as correctly observed and handles error in the iterative solver.

The impact of approximate numerical computing has been shown useful for several applications. In [7], eigenvectors of a kernel matrix are approximated with the power method and then k-means is applied to these approximations. The k-means objective function is well approximated when using approximate eigenvectors. The bounds given in [7] depend on using the k eigenvectors to partition into k parts and depend on the kth spectral gap to control accuracy of approximation. Experiments also show that k-means on the approximate eigenvectors is faster and sometimes more accurate in terms of Normalized Mutual Information (NMI) compared to using exact eigenvectors. Our paper focuses on partitioning into two clusters based on sweep cuts of a single approximate eigenvector and makes a rigorous analysis of a model problem in order to understand how the numerical accuracy interacts with combinatorial structure of the data clusters. Pothen et al. [75], which used spectral partitioning for distributed memory sparse matrix computation, recognized the value of low-accuracy solutions. Approximate spectral coordinates are used to reorder matrices before conducting high accuracy linear solves. Our paper contributes to the understanding of how numerical approximation accuracy contributes to data analysis accuracy.

6.2 Blends of Eigenvectors

In order to understand the relationship between eigensolver error and graph partitioning, we study error bounds and the effect of pointwise error on the sweep cut procedure. Theorems 4 and 5 bound the error to a subspace in terms of the residual and quantities derived from the eigenvalues of the matrix. This control over the error is then used in Theorem 6 to relate eigenresidual to the conductance of a sweep cut of the graph. These results apply to general matrices. Although a small spectral gap implies poor control on the error to a single eigenvector, we derive a condition where low accuracy approximations effectively partition the graph. Section 6.3 applies these theorems to a special family of graphs to show that blends are faster to compute and provide nearly optimal partitions.

6.2.1 Converging to a Single Eigenspace

Let $\hat{L} \in \mathbb{R}^{n\times n}$, $\hat{L} = \hat{L}^T$ be a general symmetric matrix. Consider the solutions to the equation $\hat{L}\mathbf{v} = \lambda\mathbf{v}$. Because $\hat{L}$ is symmetric, there are n eigenvalues in $\mathbb{R}$ (counting multiplicities). The set of all eigenvalues is the spectrum $\lambda(\hat{L})$, which we order decreasingly as $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. For $k = 1, \ldots, n$, let $\mathbf{v}_k$ be an eigenvector associated with λ_k, $\hat{L}\mathbf{v}_k = \lambda_k\mathbf{v}_k$, such that $\mathbf{v}_k^T\mathbf{v}_l = 0$ whenever $l \neq k$. Define the eigenspace associated with λ_k as the invariant subspace associated with λ_k, that is

$$\mathcal{X}_k := \left\{ \mathbf{x} \in \mathbb{R}^n \,:\, \hat{L}\mathbf{x} = \lambda_k\mathbf{x} \right\}.$$

These definitions imply $\dim(\mathcal{X}_k) = \operatorname{mult}(\lambda_k)$ and $\mathcal{X}_k = \mathcal{X}_l$ when $\lambda_k = \lambda_l$.

Remark 1. *The results in this section are stated and proved in terms of generic symmetric matrix $\hat{L}$ because they apply beyond spectral graph theory. Spectral partitioning methods use the eigenvectors of $\hat{L} = I - \hat{A}$. Counting eigenvalues from largest to smallest starting with 1, we see $\lambda_2(\hat{A}) = \lambda_{n-1}(\hat{L})$ with the same eigenvectors. Letting $\mathbf{k} = D^{1/2}\mathbf{1}\left\|D^{1/2}\mathbf{1}\right\|^{-1}$, the normalized eigenvector of $\hat{A}$ associated with $\lambda_1(\hat{A}) = 1$, one can use $\hat{L} = \hat{A} - \mathbf{k}\mathbf{k}^t$ in the results of this section. Subtracting $\mathbf{k}\mathbf{k}^T$ moves the eigenvalue* 1 *to* 0*, or $\mathbf{k}$ is an eigenvector associated with $0 \in \lambda(\hat{A} - \mathbf{k}\mathbf{k}^t)$. Thus, for this $\hat{L}$ and for k where $\lambda_k(\hat{L}) > 0$, we have*

$$\lambda_k(\hat{L}) = \lambda_{k+1}(\hat{A}).$$

In particular, for the Fiedler eigenvalue, $\lambda_1(\hat{L}) = \lambda_2(\hat{A}) = 1 - \lambda_{n-1}(\hat{L})$, and the Fiedler vectors in the associated eigenspace correspond to extremal eigenvalue $\lambda_1(\hat{L})$. Computationally, implementations of iterative methods approximating the eigenvectors of $\lambda_2(\hat{A})$ perform better with a routine applying the operator $\hat{A} - \mathbf{k}\mathbf{k}^t$.

Let $(\mathbf{x}, \mu)$, $\mathbf{x} \in \mathbb{R}^n$, $\mu \in \mathbb{R}$, be an approximate eigenpair of $\hat{L}$ with $\|\mathbf{x}\| = 1$ and $\mu = \mathbf{x}^T\hat{L}\mathbf{x}$, the *Rayleigh quotient*, which minimizes the function $\left\|\hat{L}\mathbf{x} - \theta\mathbf{x}\right\|$ over all real values θ. Define the two-norm of the eigenresidual as $\epsilon = \left\|\hat{L}\mathbf{x} - \mu\mathbf{x}\right\|$. As in [73, 50], we have a simple eigenvalue bound. By decomposing $\mathbf{x}$ in the eigenbasis $\mathbf{x} = \sum_{k=1}^n \alpha_k \mathbf{v}_k$, we see

$$\epsilon^2 = \left\|\hat{L}\mathbf{x} - \mu\mathbf{x}\right\|^2 = \sum_{k=1}^n \alpha_k^2(\lambda_k - \mu)^2 \geq \left(\sum_{k=1}^n \alpha_k^2\right)\left(\min_{1\leq k\leq n}(\lambda - \mu)^2\right)$$

meaning there exists an eigenvalue λ_k within ϵ of μ,

$$\min_{1\leq k\leq n}|\lambda_k - \mu| \leq \epsilon.$$

Also in [73], we have bounds estimating convergence to an eigenspace in angle. Define the eigengap for λ_k as $\delta_k = \min_{\lambda\in\lambda(\hat{L})\setminus\lambda_k}|\lambda_k - \lambda|$. Moreover, if ϵ is small compared to δ_k there exists a normalized eigenvector $\mathbf{v} \in \mathcal{X}_k$ with which $\mathbf{x}$ has a small angle,

$$\min_{\mathbf{v}\in\mathcal{X}_k} \sqrt{1 - \langle\, \mathbf{x},\, \mathbf{v}\,\rangle^2} \le \frac{\epsilon}{\delta_k}.$$

Instead of presenting a proof of this well-known result, we derive a similar bound for ℓ_2 and point-wise approximation to an eigenspace associated with an extremal eigenvector.

Theorem 4. *Consider approximate eigenpair* $(\mathbf{x}, \mu)$ *of symmetric* $\hat{L} \in \mathbb{R}^{n\times n}$ *with* $\|\mathbf{x}\| = 1$ *and* $\mu = \mathbf{x}^T \hat{L}\mathbf{x}$*. Assume*

$$|\mu - \lambda_1| < \min_{\lambda\in\lambda(\hat{L})\backslash\lambda_1} |\mu - \lambda|.$$

Given eigenresidual ϵ *and eigengap* δ_1*, there exists an eigenvector* $\mathbf{v} \in \mathcal{X}_1$*, with* $\|\mathbf{v}\| = 1$*, and error bound*

$$\|\mathbf{x} - \mathbf{v}\| \le \frac{\sqrt{8}\epsilon}{\delta_1}.$$

Proof. Let $\alpha\mathbf{v}$ be the closest vector in $\mathcal{X}_1$ to $\mathbf{x}$. Decompose $\mathbf{x}$ into its eigenvector components within $\mathcal{X}_1$ and perpendicular to $\mathcal{X}_1$, $\mathbf{x} = \alpha\mathbf{v} + \sum_{\mathbf{v}_k\perp\mathcal{X}_1} \alpha_k \mathbf{v}_k$. Because μ is closer to λ_1 than any other eigenvalue, we have

$$\epsilon^2 \;=\; |\lambda_1 - \mu|^2\alpha^2 + \sum_{\mathbf{v}_k\perp\mathcal{X}_k} |\lambda_k - \mu|^2\alpha_k^2 \;\ge\; \sum_{\mathbf{v}_k\perp\mathcal{X}_k} |\lambda_k - \mu|^2\alpha_k^2 \ge \frac{\delta_1^2}{4} \sum_{\mathbf{v}_k\perp\mathcal{X}_k} \alpha_k^2.$$

Rearranging gives

$$\frac{4\epsilon^2}{\delta_1^2} \;\ge\; \sum_{\mathbf{v}_k\perp\mathcal{X}_k} \alpha_k^2 = 1 - \alpha^2.$$

$$\|\mathbf{x} - \mathbf{v}\|^2 \;=\; \|\mathbf{x} - \alpha\mathbf{v}\|^2 + \|\mathbf{v} - \alpha\mathbf{v}\|^2 \le \|\mathbf{x} - \alpha\mathbf{v}\|^2 + (1-\alpha)^2 \le 2(1-\alpha^2) \le \frac{8\epsilon^2}{\delta_1^2},$$

□

This result implies,

$$\frac{1}{n}\sum_{i\in\mathcal{V}}|x_i - v_i|^2 \le \|\mathbf{x}-\mathbf{v}\|^2 \le \frac{8\epsilon^2}{n\delta_1^2}$$

so if ϵ^2/n is small compared to the δ_1^2, then the average average error squared is also small. Moreover we have a point-wise error bound,

$$\max_{i\in\mathcal{V}}|x_i - v_i| \le \|\mathbf{x}-\mathbf{v}\| \le \frac{\sqrt{8}\epsilon}{\delta_1}$$

For a large graph, it is typical that Fiedler eigenvalue is so close to the next-to-largest eigenvalue, that the error bounds demand an extremely small eigenresidual for convergence. Note that this error analysis is independent of the algorithm used to compute the solution. Choosing $\mathbf{x} = \mathbf{v}_3$ shows that the condition $|\mu - \lambda_1| < \min_{\lambda\in\lambda(\hat{L})\setminus\lambda_1}|\mu - \lambda|$ is necessary. Thus, for matrices $\hat{L}$ with small δ, we need to remove this condition. When the δ is small, a reasonable number of iterations of an eigensolver may produce a vector $\mathbf{x}$ with $\mathbf{x}^T\hat{L}\mathbf{x}$ close to λ_1, which may not be very close to the true extremal eigenspace. In this case we examine convergence to a linear combination of eigenvectors associated with a range of eigenvalues.

In some situations, a small spectral gap does not imply that the graph is difficult to partition. Section 6.3 describes a class of graphs that are easy to partition with an approximate eigenvector, but for which calculating an eigenvector with close point-wise approximation is extremely costly. Let $\mathcal{G}_0$ be the ring of cliques in Section 6.3.1 and form $\mathcal{G}$ by weighting one internal edge (i, j) with $1 - \eta$. If the edge is chosen so that i and j are both interior vertices (no connections outside their blocks) then the spectral gap for $\mathcal{G}$ gets arbitrarily small as $\eta \to 0$ by We omit this algebra, but note it yields

$$\delta \le \frac{c\eta^2}{(b-1)^2},$$

where $c < 1$ is a positive constant independent of η.

6.2.2 Converging to a Subspace Spanned by Multiple Eigenspaces

This section generalizes the previous error bound to the distance between $\mathbf{x}$ and a subspace spanned by the eigenvectors associated with a range of eigenvalues. Assume that linear combinations of eigenvectors associated with a range of eigenvalues $[\lambda_q, \lambda_p]$ are satisfactory for subsequent data analysis algorithms. If the Rayleigh quotient is within $[\lambda_q, \lambda_p]$ and the eigenresidual is smaller than the distance between μ and any eigenvalue outside $[\lambda_q, \lambda_p]$, then the following theorem holds. The following theorem is a consequence of the Davis–Kahan Theorem [18]. It is presented here in this form for clarity..

Theorem 5. *Consider approximate eigenpair* $(\mathbf{x}, \mu)$ *of symmetric* $\hat{L} \in \mathbb{R}^{n\times n}$ *with* $\|\mathbf{x}\| = 1$ *and* $\mu = \mathbf{x}^T\hat{L}\mathbf{x} \in [\lambda_q, \lambda_p]$. *Define*

$$\delta_{p,q}(\mu) = \min_{\{k<p\}\cup\{k>q\}} |\lambda_k - \mu| \qquad \textit{and} \qquad \mathcal{X}_p^q := \bigotimes_{k=p}^{q} \mathcal{X}_k.$$

Given eigenresidual $\epsilon = \left\|\hat{L}\mathbf{x} - \mu\mathbf{x}\right\| \le \min(\mu - \lambda_{q-1}, \lambda_{p+1} - \mu)$ *there exists a vector* $\mathbf{v} \in \mathcal{X}_p^q$, *with* $\|\mathbf{v}\| = 1$, ℓ_2 *error bound,*

$$\|\mathbf{x} - \mathbf{v}\| \le \frac{\sqrt{2}\epsilon}{\delta_{p,q}(\mu)},$$

Proof. Let $\mathbf{x} = \sum_{k=1}^n \alpha_k \mathbf{v}_k$ and $\Pi\mathbf{x} = \sum_{k=p}^q \alpha_k \mathbf{v}_k$, the ℓ_2-orthogonal projection onto $\mathcal{X}_p^q$. Note $\sum_{k=1}^n \alpha_k^2 = 1$. In the case where $\|\Pi\mathbf{x}\| = 1$, we can let $\mathbf{v} = \mathbf{x}$ and see the bound is clearly satisfied. For $\|\Pi\mathbf{x}\| < 1$, we first demonstrate that $\|\mathbf{x} - \Pi\mathbf{x}\|$ is controlled by ϵ,

$$\begin{aligned}
\epsilon^2 &= \left\|\hat{L}\mathbf{x} - \mu\mathbf{x}\right\|^2 \\
\epsilon^2 &= \sum_{k=1}^{n} \alpha_k^2 |\lambda_k - \mu|^2 \\
\epsilon^2 &= \sum_{\{k<p\}\cup\{k>q\}} \alpha_k^2 |\lambda_k - \mu|^2 + \sum_{k=p}^{q} \alpha_k^2 |\lambda_k - \mu|^2 \\
\epsilon^2 &\geq \delta_{p,q}(\mu)^2 \sum_{\{k<p\}\cup\{k>q\}} \alpha_k^2 + \sum_{k=p}^{q} \alpha_k^2 |\lambda_k - \mu|^2 \\
\epsilon^2 &\geq \delta_{p,q}(\mu)^2 \sum_{\{k<p\}\cup\{k>q\}} \alpha_k^2 \\
\epsilon^2 &\geq \delta_{p,q}(\mu)^2 \left(1 - \sum_{k=p}^{q} \alpha_k^2\right) \\
\frac{\epsilon}{\delta_{p,q}(\mu)} &\geq \|\mathbf{x} - \Pi\mathbf{x}\|.
\end{aligned}$$

There is a unit vector in $\mathcal{X}_p^q$ that is also within some factor of ϵ to $\mathbf{x}$. Let $\mathbf{v} = \Pi\mathbf{x}/\|\Pi\mathbf{x}\|$, then $\|\mathbf{v} - \Pi\mathbf{x}\| = 1 - \|\Pi\mathbf{x}\|$. We have

$$\begin{aligned}
\|\Pi\mathbf{x}\|^2 + \|(I - \Pi)\mathbf{x}\|^2 &= 1 \\
\|\Pi\mathbf{x}\|^2 &= 1 - \|(I - \Pi)\mathbf{x}\|^2 \\
\|\Pi\mathbf{x}\|^2 &\geq 1 - \frac{\epsilon^2}{\delta_{p,q}(\mu)^2} \\
\|\Pi\mathbf{x}\| &\geq \sqrt{1 - \frac{\epsilon^2}{\delta_{p,q}(\mu)^2}}.
\end{aligned}$$

Using the inequality $a \leq \sqrt{a}$ for $a \in (0, 1)$, we see

$$\|\mathbf{v} - \Pi\mathbf{x}\| = 1 - \|\Pi\mathbf{x}\| \leq 1 - \sqrt{1 - \frac{\epsilon^2}{\delta_{p,q}(\mu)^2}} \leq \frac{\epsilon}{\delta_{p,q}(\mu)}.$$

Then, because $(\mathbf{x} - \Pi\mathbf{x})^T(\Pi\mathbf{x} - \mathbf{v}) = 0$, we have

$$\|\mathbf{x} - \mathbf{v}\|^2 = \|\mathbf{x} - \Pi\mathbf{x}\|^2 + \|\Pi\mathbf{x} - \mathbf{v}\|^2 \leq \frac{2\epsilon^2}{\delta_{p,q}(\mu)^2}.$$

□

Remark 2. *Note that the size of the* blend gap, $\delta_{p,q}(\mu)$, *is dependent on i) the size of* $|\lambda_p - \lambda_q|$, *ii) how internal Rayleigh quotient* μ *is within* $[\lambda_q, \lambda_p]$, *iii) and how far the exterior eigenvalues are,* $|\lambda_q - \lambda_{q+1}|$ *and* $|\lambda_{p-1} - \lambda_p|$. *For a problem where the spectrum is not known a priori, it is difficult to state an acceptable interval* $[\lambda_q, \lambda_p]$ *for accomplishing a given data mining task. Section 6.3.2 provides an example where one can choose* $[\lambda_q, \lambda_p]$ *a priori. The Congress graph has* $\lambda_{41} - \lambda_{42} \geq 0.4$ *and the first* 41 *eigenvectors indicate the natural clustering of the graph into sessions [14]. This analysis thus applies to this real world network.*

For our application, $p = 1$ and $\delta_{pq} = \mu - \lambda_{q+1}\left(\hat{A} - \mathbf{k}\mathbf{k}^t\right) \geq \lambda_q - \lambda_{q+1}$, which can be much larger than the spectral gap $\lambda_1(\hat{A} - \mathbf{k}\mathbf{k}^t) - \lambda_2(\hat{A} - \mathbf{k}\mathbf{k}^t)$. In Section 6.2.1 the goal of computation is a single eigenvector and the output of the approximation is a blend of eigenvectors, the coefficients of the output in the eigenbasis of the matrix describes the error introduced by approximation. The goal of computation in Section 6.2.2 is a blend of eigenvectors, and we improve the error bound when the spectral gap is small.

In order to relate numerical accuracy to conductance for general graphs we examine the impact of pointwise error on sweep cuts. For any prescribed conductance value ψ, we derive a condition on vectors $\mathbf{v}$ such that we can guarantee that small perturbations of $\mathbf{v}$ have conductance less than or equal to ψ. Let $S_{\mathbf{v}}(t)$ represent the sweep cut of $\mathbf{v}$ at t as in Equation (21).

Lemma 3. *For any graph G, vector* $\mathbf{v} \in \mathbb{R}^n$ *and scalar* $\psi > 0$, *define* $T_\psi(\mathbf{v}) = \{t \mid \phi(S_{\mathbf{v}}(t)) \leq \psi\}$. *Let* $g_{\mathbf{v}}(t) = \min_i |v_i - t|$ *and* $g_{\mathbf{v}} = \max_{t \in T_\psi(\mathbf{v})} g_{\mathbf{v}}(t)$. *If* $\|\mathbf{z}\|_\infty < g_{\mathbf{v}}$, *then* $\phi(\mathbf{v} + \mathbf{z}) \leq \psi$.

Proof. If $S_{\mathbf{v}}(t) = S_{\mathbf{v}+\mathbf{z}}(t)$ we can apply $\phi(\mathbf{v} + \mathbf{z}) < \phi(S_{\mathbf{v}+\mathbf{z}}(t)) = \phi(S_{\mathbf{v}}(t)) \leq \psi$. $S_{\mathbf{v}}(t) = S_{\mathbf{v}+\mathbf{z}}(t)$ if and only if $sign(v_i + z_i - t) = sign(v_i - t)$ for all i. By checking cases, one can see that $\|\mathbf{z}\|_\infty < \min_i |v_i - t|$ is sufficient to guarantee that $v_i + z_i - t$ has the same sign as $v_i - t$. $\square$

Note that Lemma 3 is not a necessary condition — $\mathbf{z} = \frac{1}{2}(\mathbf{v} - t\mathbf{1})$ is a much larger perturbation of $\mathbf{v}$ such that $\phi(\mathbf{v} + \mathbf{z}) \leq \psi$. Lemma 3 defines $g_{\mathbf{v}}$ as a measure of sensitivity of a single vector with respect to preserving sweep cuts of conductance less than or equal to ψ. For vectors $\mathbf{v}$ with small $g_{\mathbf{v}}$, a small perturbation can disrupt the sweep cuts which achieve conductance less than ψ. By defining the sensitivity of an invariant subspace appropriately, Theorem 6 provides a path to deriving a residual tolerance for arbitrary graphs. The minimum and maximum degree of G are denoted by d_{min}, d_{max}.

Theorem 6. *Let G be a graph and $\psi > 0$. Define $V = span\{D^{-\frac{1}{2}}\mathbf{v}_1 \ldots D^{-\frac{1}{2}}\mathbf{v}_q\}$, where for $j \in \{1 \ldots q\}$ $\hat{A}\mathbf{v}_j = \lambda_q \mathbf{v}_j$ and $\mathbf{v}_j$ are orthogonal. For any vector $\mathbf{x}$, let $\mu = \mathbf{x}^T \hat{A}\mathbf{x}$ and $\delta_{p,q}(\mu) = \min_{\{k<p\}\cup\{k>q\}} |\lambda_k - \mu|$. For any $\mathbf{q} \in V$, let $g_{\mathbf{v}}$ be defined as in Lemma 3. Define $g = \min_{\mathbf{v}\in V, \|\mathbf{v}\|_2 = 1} g_{\mathbf{v}}$. If $\left\|\hat{A}\mathbf{x} - \mu\mathbf{x}\right\| < \frac{1}{\sqrt{2}}\sqrt{\frac{d_{min}}{d_{max}}}\delta_{p,q}(\mu) g$, then $\phi\left(D^{-\frac{1}{2}}\mathbf{x}\right) \leq \psi$.*

Proof. By Theorem 5 applied to $\mathbf{x}$, there is a unit vector $\mathbf{q} \in Span\{\mathbf{v}_1 \ldots \mathbf{v}_q\}$ such that $\|\mathbf{x} - \mathbf{q}\|_\infty \leq \|\mathbf{x} - \mathbf{q}\|_2 < \sqrt{\frac{d_{min}}{d_{max}}} g$. Define $\mathbf{z} = (D^{-\frac{1}{2}}\mathbf{x} - \mathbf{v}) \|\mathbf{v}\|^{-1}$, where $\mathbf{v} = D^{-\frac{1}{2}}\mathbf{q} \in V$. By scaling and normalizing we see

$$\|\mathbf{z}\|_2 = \frac{\left\|D^{-\frac{1}{2}}\mathbf{x} - D^{-\frac{1}{2}}\mathbf{q}\right\|_2}{\left\|D^{-\frac{1}{2}}\mathbf{q}\right\|_2} < \sqrt{\frac{d_{max}}{d_{min}}} \|\mathbf{x} - \mathbf{q}\| < g$$

Since

$$\|\mathbf{z}\|_\infty < g = \min_{\mathbf{v}\in V, \|\mathbf{v}\|_2 = 1} \max_{t \in T_{\mathbf{v}}} \min_{i \in \{1 \ldots n\}} |v_i - t| < g_{\mathbf{v}},$$

Lemma 3 implies $\phi\left(D^{-\frac{1}{2}}\mathbf{x}\right) \leq \psi$. □

If one can bound the value of g from below, then Theorem 6 gives a residual tolerance for the eigenvector approximation when using sweep cuts to partition the graph. Section 6.3.3 applies this theorem to the ring of cliques family of graphs.

This section connects the eigenresidual to the error when computing blends of eigenvectors, and quantifies the impact of error on the sweep cut procedure. If the eigengap is small enough, then one cannot guarantee that the Rayleigh quotient is closest to the Fiedler value, thus one cannot guarantee that the computed eigenvector is close to the desired eigenvector. In this small gap setting, a small eigenresidual indicates that the computed vector is close to the desired invariant subspace. Theorem 6 shows that vectors with small eigenresidual preserve low conductance sweep cuts for general graphs. Theorem 6 illustrates how the residual tolerance depends on both the blend gap $\delta_{p,q}(\mu)$ and the sensitivity g of the eigenvectors. The following section applies this theory to the ring of cliques in order to derive solver tolerances for graph partitioning.

6.3 The Ring of Cliques

To demonstrate the theory developed in Section 6.2, we employ a previously studied model problem, the ring of cliques [30]. Theorems 7 and 8 derive explicit formulas for all eigenvalues and eigenvectors. These formulas determine the relevant residual tolerance. Moreover, complete spectral knowledge gives a strong understanding of the convergence properties of simple iterative methods.

The ring of cliques has several attractive properties for analysis of spectral partitioning. The community structure is as extreme as possible for a connected graph, so the solution is well-defined. Also, theorems about block circulant matrices [85] produce closed form solutions to the eigenvector problem. This graph serves as a canonical example of when solving an eigenproblem accurately is unnecessarily expensive to achieve data analysis success. This example shows that it is possible for the combinatorial structure of the data to be revealed faster than the algebraic structure of the associated matrices. The graph is simple to partition accurately as there are many cuts relatively close to the minimum, any robust partitioning algorithm will correctly recover the cliques in this graph. However, a Fiedler eigenvector is difficult to calculate with guarantees of point-wise accuracy when using non-preconditioned iterative methods. An algorithm that computes a highly accurate eigenpair will be inefficient on large problem instances. Sections 6.3.3 and 6.3.4 apply the tools from Section 6.2 in order to derive a residual tolerance sufficient for solving the ring of cliques. Section 6.3.5 bounds the number of power method iterations necessary to recover the ring of cliques, and Section 6.3.6 validates and illustrates these observations with an experiment.

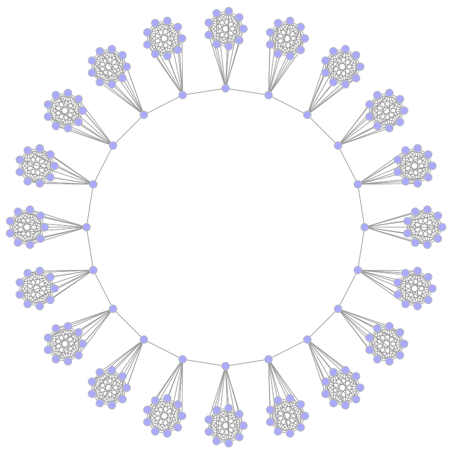
6.3.1 Definition

A q-ring of b-cliques, $\mathcal{R}_{b,q}$, is parameterized by a block size b and a number of blocks q. Each block represents a clique of size b and all possible internal connections exist within each individual set of b vertices. For each block, there is a single vertex called the *corner* connected to the corners of the adjacent cliques. These q corners form a ring. Each block also has $(b-1)$ *internal* vertices that have no edges leaving the block. The adjacency matrix associated with $\mathcal{R}_{b,q}$ is a sum of tensor products of simple matrices (identity, cycle, and rank-one matrices). We have

$$A = I_q \otimes (\mathbf{1}_b \mathbf{1}_b^T - I_b) + C_q \otimes (\mathbf{e}_1 \mathbf{e}_1^T),$$

where the I_k are identity matrices of dimension k, $\mathbf{1}_b$ is a constant vector with dimension b, $\mathbf{e}_1 \in \mathbb{R}^b$ is the cardinal vector with a one in its first entry and zero elsewhere, and C_q is the adjacency matrix of a cycle graph on q vertices. The matrix A and other matrices associated with this graph are *block-circulant*, which implies the eigenvectors are the Kronecker product of a periodic vector and the eigenvectors of a small eigenproblem defined by structure in the blocks. Figure 18a shows the structure of the graph, and Figure 18b shows the block structure of the adjacency matrix.

Any partition that breaks a clique cuts at least $b-2$ edges while any partition that does not break any cliques cuts at most q edges. The minimum conductance partition is break the ring into two contiguous halves by cutting exactly two edges. There are $q/2$ partitions that achieve this minimal cut for even q. We will consider



$$\begin{bmatrix} J_b & \mathbf{e}_1\mathbf{e}_1^T & 0 & \cdots & & \mathbf{e}_1\mathbf{e}_1^T \\ \mathbf{e}_1\mathbf{e}_1^T & J_b & \mathbf{e}_1\mathbf{e}_1^T & 0 & & \\ 0 & \mathbf{e}_1\mathbf{e}_1^T & J_b & \mathbf{e}_1\mathbf{e}_1^T & 0 & \\ \vdots & & \ddots & \ddots & \ddots & \\ & & 0 & \mathbf{e}_1\mathbf{e}_1^T & J_b & \mathbf{e}_1\mathbf{e}_1^T \\ \mathbf{e}_1\mathbf{e}_1^T & & & 0 & \mathbf{e}_1\mathbf{e}_1^T & J_b \end{bmatrix}$$

$$\text{where} \quad J_b = \mathbf{1}_b\mathbf{1}_b^T - I_b.$$

(a) A drawing of $\mathcal{R}_{b,q}$ laid out to show structure.

(b) The adjacency matrix of $\mathcal{R}_{b,q}$ has block circulant structure.

any of these equivalent. Any partition that breaks fewer than $b-2$ edges will be regarded as a good, but not optimal cut. The fact that many partitions are close to optimal and the vast majority of partitions are very far from optimal is a feature of this model problem.

6.3.2 Eigenpairs of ROC Normalized Adjacency Matrix

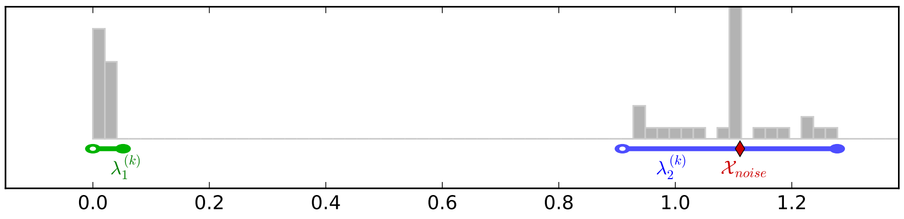


Figure 19: Distribution of eigenvalues of $\hat{L}$ for $\mathcal{R}_{b=8,q=32}$. The gray bars represent a histogram of the eigenvalues, including multiplicities. The red diamond represents a large multiplicity of $(n-2q)$ at $-(b-1)^{-1}$ corresponding to $\mathcal{X}_{noise}$ (see Theorem 7). There is a green interval near 1 containing the portion of the spectrum given by $\lambda_1^{(k)}$ and a blue interval near 0 containing $\lambda_1^{(k)}$ for $k = 0, 1, \ldots \lfloor q/2 \rfloor$ (see Theorem 8). The open left endpoint of each interval signifies that the eigenvalue corresponding to $\lambda_{\lceil q/2 \rceil}^{(k)}$ is not present when q is odd. Small eigenvalues of the Laplacian matrix correspond to large eigenvalues of the Adjacency matrix.

Due to the block-circulant structure of the ring of cliques, we can compute the eigenvalues and eigenvectors in a closed form. Let $\mathbf{c}^{k,q}, \mathbf{s}^{k,q}$ be the periodic vectors as defined in Equation (22).

$$\mathbf{c}_j^{k,q} = \cos\left(\frac{2\pi k}{q} j\right) \quad \mathbf{s}_j^{k,q} = \sin\left(\frac{2\pi k}{q} j\right) \quad \text{for } j = \{1 \dots q\} \tag{22}$$

These vectors are periodic functions with k cycles through the interval $[-1, 1]$ sampled at the qth roots of unity. We employ results from [41] and [85] to derive the full spectrum of $\hat{A}$ and a full eigenbasis. In summary, there is an eigenvalue $-(b-1)^{-1}$ with a large multiplicity, $n - 2q = (b-2)q$. Furthermore, the eigenspace associated with $-(b-1)^{-1}$ can be represented in a basis that contains variation internal to each clique, that is with eigenvectors of the form $\mathbf{h} \otimes \mathbf{e}_i$ where $\mathbf{e}_i$ is the ith standard basis vector for each $i \in \{1 \dots q\}$. For this reason, we call $\lambda_{noise} = -(b-1)^{-1}$ a noise eigenvalue. The positive eigenvalues are called signal eigenvalues. The signal eigenvectors have the form $\mathbf{p}^{k,q} \otimes (\xi \mathbf{e}_1 + \mathbf{g}_1)$, where $\mathbf{p}^{k,q}$ is either $\mathbf{s}^{k,q}$ or $\mathbf{c}^{k,q}$, $\mathbf{e}_1$ is one in its first entry, $\mathbf{g}_1$ is zero in its first entry and one elsewhere, and ξ is a scalar. All of the internal members of the cliques take the same value in any eigenvector associated with $\lambda_k(\hat{A}) \neq (b-1)^{-1}$. The slowly varying eigenvectors (associated with $\lambda_k(\hat{A}) \approx 1$) give nearly optimal partitions of the graph. Linear combinations of these slowly varying signal eigenvectors also give low conductance partitions. There are $q-1$ non-localized eigenvectors with small positive or negative eigenvalues. These eigenvectors have the internal clique members and their corner with different sign which causes them to misclassify some of the corners. The distribution of the eigenvalues of $\mathcal{R}_{b,q}$ is illustrated in Figure 19. The rest of Section 6.3.2 contains formulas for the eigenpairs and the details of their derivations.

Theorem 7. ($\mathcal{R}_{b,q}$ Noise Eigenpairs) *There is an eigenspace $\mathcal{X}_{noise}$ of multiplicity $(n-2q)$ associated with eigenvalue*

$$\lambda_{noise} = \frac{-1}{b-1}.$$

Any vector that is zero-valued on all corner vertices and sums to zero on each individual set of internal vertices is in $\mathcal{X}_{noise}$.

Proof. This is a specific case of locally supported eigenvectors [41] (LSEVs) brought on by a high level of local symmetry in $\mathcal{R}_{b,q}$. For each clique $\mathcal{K}$, let $\mathbf{x} = \mathbf{e}_i - \mathbf{e}_j$ for vertices i and j that are internal vertices of $\mathcal{K}$. Both $\mathbf{x}$ and $\hat{A}\mathbf{x}$ are zero valued outside of $\mathcal{K}$. Internally, due to $D^{-1/2}(\mathbf{e}_i - \mathbf{e}_j) = (b-1)^{-1/2}(\mathbf{e}_i - \mathbf{e}_j)$ and the orthogonality $\mathbf{1}_b^T(\mathbf{e}_i - \mathbf{e}_j) = 0$, we see

$$\hat{A}(\mathbf{e}_i - \mathbf{e}_j) = \frac{1}{b-1}(\mathbf{1}_b\mathbf{1}_b^T - I)(\mathbf{e}_i - \mathbf{e}_j) = \frac{-1}{b-1}(\mathbf{e}_i - \mathbf{e}_j).$$

Thus, $\mathbf{x}$ is an eigenvector of $\hat{A}$ associated with $-1/(b-1)$. There are $(b-2)$ such linearly independent eigenvectors for $\mathcal{K}$, and the same is true for all q cliques. Thus, we have a multiplicity of $q(b-2) = n - 2q$ for eigenvalue $\lambda_{noise} = (b-1)^{-1}$.

□

These vectors are in the interior of the spectrum and thus are very well attenuated by the power-method[1]. The remaining eigenvectors must be constant on the internal nodes of the blocks because of orthogonality to the LSEVs which are spanned by $\mathbf{e}_i - \mathbf{e}_j$. In any vector $\mathbf{v}$ the projection of $\mathbf{v}$ onto the global eigenvectors defines a mean value for the elements of the blocks. Since all of the eigenvectors of interest are orthogonal to the constant vector, their entries must sum to zero. So the LSEVs cannot change the mean of a block. The remaining eigenvectors are given in Theorem 8.

[1]In a single iteration the shifted power method, $\mathbf{x}_{k+1} = (\hat{A} + (b-1)^{-1}I)\mathbf{x}_k$, perfectly eliminates all of the energy in the $(n-2q)$-dimensional eigenspace associated with $\lambda = -(b-1)^{-1}$. If the graph is perturbed with the addition and removal of a few edges, the eigenvectors become slightly less localized and the associated eigenvalues spread out to a short range of values and are not perfectly eliminated in a single iteration. However, the power method or a Krylov method will rapidly attenuate the energy in the associated eigenspaces.

Theorem 8. **($\mathcal{R}_{b,q}$ Signal Eigenpairs)** *For $k = 0, \ldots \lceil \frac{q}{2} \rceil - 1$, define*

$$\begin{aligned}
\alpha_k &= 2\cos\left(\frac{2\pi k}{q}\right) \\
\beta_k &= \frac{1}{2}\left(\alpha_k\sqrt{\frac{b-1}{b+1}} - \sqrt{b^2-1} + \sqrt{\frac{b+1}{b-1}}\right) \\
\xi_1^{(k)} &= \beta_k + \sqrt{\beta_k^2 + (b-1)} \\
\xi_2^{(k)} &= \beta_k - \sqrt{\beta_k^2 + (b-1)} \\
\lambda_1^{(k)} &= \frac{\xi_1^{(k)}}{\sqrt{b^2-1}} + 1 - \frac{1}{b-1} \\
\lambda_2^{(k)} &= \frac{\xi_2^{(k)}}{\sqrt{b^2-1}} + 1 - \frac{1}{b-1}
\end{aligned}$$

Let $\mathbf{1}_b$ and $\mathbf{1}_q$ be the vectors of all ones in $\mathbb{R}^b$ and $\mathbb{R}^q$, respectively. Also let $\mathbf{e}_1 \in \mathbb{R}^b$ have a one in its first entry, zero elsewhere and $\mathbf{g}_1 = \mathbf{1}_b - \mathbf{e}_1$. We have the following eigenpairs.

(i) For $k = 0$, we have 2 eigenvalues of $\hat{A}$, $\lambda_1^{(0)}$ and $\lambda_2^{(0)}$, each with multiplicity 1. The associated (unnormalized) eigenvectors are

$$\mathbf{v}_1^{(0)} = \sqrt{b+1}(\mathbf{1}_q \otimes \mathbf{e}_1) + \sqrt{b-1}(\mathbf{1}_q \otimes \mathbf{g}_1)$$

and

$$\mathbf{v}_2^{(0)} = (b-1)^{3/2}(\mathbf{1}_q \otimes \mathbf{e}_1) - \sqrt{b+1}(\mathbf{1}_q \otimes \mathbf{g}_1)$$

respectively.

(ii) For each $k = 1, ..., \lceil \frac{q}{2} \rceil - 1$, we have 2 eigenvalues of $\hat{A}$, $\lambda_1^{(k)}$ and $\lambda_2^{(k)}$, each with multiplicity 2. Two independent (unnormalized) eigenvectors associated with $\lambda_1^{(k)}$ are

$$\mathbf{v}_{1,1}^{(k)} = \mathbf{c}^{k,q} \otimes \left(\xi_1^{(k)}\mathbf{e}_1 + \mathbf{g}_1\right) \quad \textit{and} \quad \mathbf{v}_{1,2}^{(k)} = \mathbf{s}^{k,q} \otimes \left(\xi_1^{(k)}\mathbf{e}_1 + \mathbf{g}_1\right).$$

Two independent (unnormalized) eigenvectors associated with $\lambda_2^{(k)}$ are

$$\mathbf{v}_{2,1}^{(k)} = \mathbf{c}^{k,q} \otimes \left(\xi_2^{(k)}\mathbf{e}_1 + \mathbf{g}_1\right) \quad \textit{and} \quad \mathbf{v}_{2,2}^{(k)} = \mathbf{s}^{k,q} \otimes \left(\xi_2^{(k)}\mathbf{e}_1 + \mathbf{g}_1\right).$$

(iii) If q is even, then for $k = \frac{q}{2}$, we have 2 eigenvalues of $\hat{A}$, $\lambda_1^{(q/2)}$ and $\lambda_2^{(q/2)}$, each with multiplicity 1. The associated (unnormalized) eigenvectors are

$$\mathbf{v}_1^{(q/2)} = \mathbf{c}^{q/2,q} \otimes \left(\xi_1^{(q/2)}\mathbf{e}_1 + \mathbf{g}_1\right)$$

and

$$\mathbf{v}_2^{(q/2)} = \mathbf{s}^{q/2,q} \otimes \left(\xi_2^{(q/2)}\mathbf{e}_1 + \mathbf{g}_1\right)$$

respectively.

Note if values of $\lambda_p^{(k)}$ and $\lambda_q^{(l)}$ coincide for $(p,k) \neq (q,l)$ the eigenvalue multiplicities add up.

Proof. Let $D_b = \operatorname{diag}((b+1)\mathbf{1} - 2\mathbf{g}_1)$. Matrix $\hat{A}$ can be organized into blocks that all co-commute with each other implying the eigenvectors are tensor products involving the eigenvectors of the blocks. We decompose $\hat{A} = (I_q \otimes B_1) + (C_q \otimes B_2)$, where I_q is the identity in $\mathbb{R}^q$, C_q is the adjacency matrix of a q- cycle, $(C_q)_{ij} = 1 \iff |i-j| = 1 \mod q$, $B_1 = D_b^{-1/2}(\mathbf{1}_b\mathbf{1}_b^T - I)D_b^{-1/2}$, and $B_2 = \frac{1}{b+1}\mathbf{e}_1\mathbf{e}_1^T$. Because any eigenvector $\mathbf{y}$ of C_q is also an eigenvector of I_q, eigenvectors of $\hat{A}$ have the form $\mathbf{y} \otimes \mathbf{z}$. Vectors $\mathbf{z}$ are derived by plugging various eigenvectors of C_q into $\mathbf{y}$ and solving for a set of constraints that $\mathbf{z}$ must satisfy for $(\mathbf{y} \otimes \mathbf{z})$ to be an eigenvector associated with $\hat{A}$.

We describe the eigendecomposition of C_q. For $k = 0, \ldots, \lceil \frac{q}{2} \rceil - 1$, $\alpha_k = 2\cos(2\pi/k)$ is an eigenvalue of C_q. For $k = 0$, α_0 is simple, and $\mathbf{1}_b$ is the associated eigenvector. For $k = 1, \ldots \lceil \frac{q}{2} \rceil$-1, α_k has multiplicity 2 and the associated 2-dimensional eigenspace is span$(\{\mathbf{c}^{k,q}, \mathbf{s}^{k,q}\})$, as defined in (22). If q is even, then $\alpha_{q/2}$ is a simple eigenvalue as well and the associated eigenvector is $\mathbf{c}^{q/2,q}$. Letting $\mathbf{y}$ be an eigenvector associated with α_k and using the properties of Kronecker products, we see

$$\begin{aligned}
\hat{A}(\mathbf{y} \otimes \mathbf{z}) &= \left[(I_q \otimes B_1) + (C_q \otimes B_2)\right](\mathbf{y} \otimes \mathbf{z}) = \left[(I_q\mathbf{y} \otimes B_1\mathbf{z}) + (C_q\mathbf{y} \otimes B_2\mathbf{z})\right] \\
&= \left[(\mathbf{y} \otimes B_1\mathbf{z}) + (\alpha_k\mathbf{y} \otimes B_2\mathbf{z})\right] = \left[(\mathbf{y} \otimes B_1\mathbf{z}) + (\mathbf{y} \otimes \alpha_k B_2\mathbf{z})\right] \\
&= \mathbf{y} \otimes (B_1\mathbf{z} + \alpha_k B_2\mathbf{z}) = \mathbf{y} \otimes \left[(B_1 + \alpha_k B_2)\mathbf{z}\right].
\end{aligned}$$

Here we see that if $\mathbf{z}$ is an eigenvector of $H_k := B_1 + \alpha_k B_2$, then $(\mathbf{y} \otimes \mathbf{z})$ is an eigenvector associated with $\hat{A}$. Observe that $H_k = D_b^{-1/2}(\mathbf{1}_b\mathbf{1}_b^T + \alpha_k\mathbf{e}_1\mathbf{e}_1^T - I)D_b^{-1/2}$ is a scaling of a rank-2 shift from the identity matrix, where we would expect 3 eigenvalues: two simple and one of multiplicity $(b-2)$.

We can easily verify that there is a $(b-2)$-dimensional eigenspace of H_k associated with $-1/(b-1)$. The tensor products of these vectors are an alternative basis associated with the locally supported eigenvectors from Theorem 7. The associated eigenspace of H_k is orthogonal to span$(\{\mathbf{e}_1, \mathbf{g}_1\})$. Due to eigenvector orthogonality, the last two eigenvectors must be in the range of span$(\{\mathbf{e}_1, \mathbf{g}_1\})$. Note that $D_b^p\mathbf{e}_1 = (b+1)^p\mathbf{e}_1$ and $D_b^p\mathbf{g}_1 = (b-1)^p\mathbf{g}_1$. We use this to solve for these eigenvectors and their associated eigenvalues in terms of α_k and b,

$$\begin{aligned}
H_k(\xi\mathbf{e}_1 + \mathbf{g}_1) &= \lambda(\xi\mathbf{e}_1 + \mathbf{g}_1) \\
(\mathbf{1}_b\mathbf{1}_b^T + \alpha_k\mathbf{e}_1\mathbf{e}_1^T - I_b)D_b^{-1/2}(\xi\mathbf{e}_1 + \mathbf{g}_1) &= \lambda D_b^{1/2}(\xi\mathbf{e}_1 + \mathbf{g}_1)
\end{aligned}$$

The right-hand side expands to

$$\left(\lambda\xi\sqrt{b+1}\right)\mathbf{e}_1 + \left(\lambda\sqrt{b-1}\right)\mathbf{g}_1,$$

The left-hand side expands and simplifies to

$$\begin{aligned}
\left(\frac{\xi}{\sqrt{b+1}} + \sqrt{b-1}\right)(\mathbf{e}_1 + \mathbf{g}_1) + \left(\frac{\xi\alpha_k}{\sqrt{b+1}} - \frac{\xi}{\sqrt{b+1}}\right)\mathbf{e}_1 + \left(\frac{-1}{\sqrt{b-1}}\right)\mathbf{g}_1 &= \\
\left(\frac{\xi\alpha_k}{\sqrt{b+1}} + \sqrt{b-1}\right)\mathbf{e}_1 + \left(\frac{\xi}{\sqrt{b+1}} + \sqrt{b-1} - \frac{1}{\sqrt{b-1}}\right)\mathbf{g}_1 &=
\end{aligned}$$

The coefficients of $\mathbf{e}_1$ and $\mathbf{g}_1$ must be equal, individually, because they are not linearly dependent. Equating the left-hand and right-hand sides and simplifying gives two nonlinear equations in ξ and λ,

$$\begin{aligned} \frac{\xi\alpha_k}{b+1} + \sqrt{\frac{b-1}{b+1}} &= \xi\lambda \qquad (23) \\ \frac{\xi}{\sqrt{b^2-1}} + 1 - \frac{1}{b-1} &= \lambda. \qquad (24) \end{aligned}$$

Multiplying the second equation by ξ, setting the left-hand sides of both equations equal to eliminate λ, then simplifying, yields the following quadratic equation in ξ,

$$\xi^2 - \left(\alpha_k\sqrt{\frac{b-1}{b+1}} - \sqrt{b^2-1} + \sqrt{\frac{b+1}{b-1}}\right)\xi - (b-1) = 0,$$

which is easily solved. Define

$$\beta_k = \frac{1}{2}\left(\alpha_k\sqrt{\frac{b-1}{b+1}} - \sqrt{b^2-1} + \sqrt{\frac{b+1}{b-1}}\right) \qquad \text{and} \qquad \gamma_k = b-1.$$

Given ξ, λ is determined by the second equation in (24). The solution set to nonlinear equations (23)-(24) is then

$$\begin{aligned} \xi_1^{(k)} &= \beta_k + \sqrt{\beta_k^2 + \gamma_k}, \qquad \lambda_1^{(k)} = \frac{\xi_1^{(k)}}{\sqrt{b^2-1}} + 1 - \frac{1}{b-1}, \qquad \text{and} \\ \xi_2^{(k)} &= \beta_k - \sqrt{\beta_k^2 + \gamma_k}, \qquad \lambda_2^{(k)} = \frac{\xi_2^{(k)}}{\sqrt{b^2-1}} + 1 - \frac{1}{b-1}. \end{aligned}$$

Thus we have local eigenpairs of H_k, $((\xi_1^{(k)}\mathbf{e}_1 + \mathbf{g}_1), \lambda_1^{(k)})$ and $((\xi_2^{(k)}\mathbf{e}_1 + \mathbf{g}_1), \lambda_2^{(k)})$. The local eigenpairs $((\xi_1^{(k)}\mathbf{e}_1 + \mathbf{g}_1), \lambda_1^{(k)})$ yield global eigenpairs of $\hat{A}$ of the form $((\mathbf{c}_k \otimes (\xi_1^{(k)}\mathbf{e}_1 + \mathbf{g}_1)), \lambda_1^{(k)})$ and $((\mathbf{s}_k \otimes (\xi_1^{(k)}\mathbf{e}_1 + \mathbf{g}_1)), \lambda_1^{(k)})$. Similarly, $((\xi_1^{(k)}\mathbf{e}_1 + \mathbf{g}_1), \lambda_2^{(k)})$ yield global eigenvectors of $\hat{A}$ associated with $\lambda_2^{(k)}$. This accounts for the last $2q$ eigenpairs of $\hat{A}$. $\square$

In order to illustrate these formulas, Figure 20 shows the computed eigenvectors for the graph $\mathcal{R}_{16,10}$ along with the eigenvalues. Eigenvectors associated with eigenvalues close to 1 have low conductance sweep cuts.

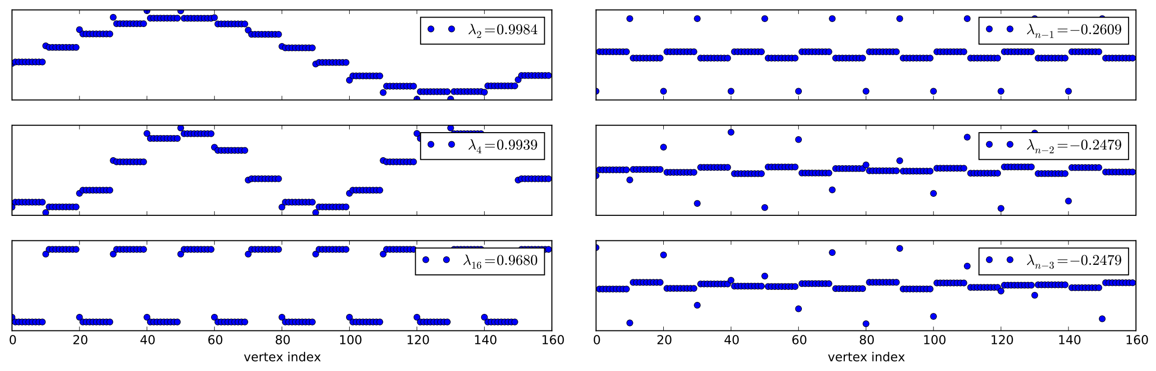


Figure 20: The eigenvectors of $\hat{A}$ are shown for $\mathcal{R}_{16,10}$. The eigenvectors with eigenvalues close to 1 (left) indicate the block structure with differing frequencies. The eigenvectors close to -1 (right) assign opposite signs to the internal vertices and corner vertices of each block.

Corollary 1. *The asymptotic expansions of the eigenvalues are as follows.*

(i) For the signal eigenpairs, we see

$$\lambda_1^{(k)} = 1 - \frac{4-\alpha_k}{2(b^2-1)} + \frac{1}{4(b-1)^2} + \frac{\alpha_k^2}{4(b+1)^2} + \mathcal{O}(b^{-3})$$

(ii) For the non-signal (and non-noise) eigenpairs, we see

$$\lambda_2^{(k)} = \frac{\alpha_k - 1}{b+1} - \frac{\alpha_k^2}{4(b+1)^2} - \frac{\alpha_k}{2(b^2-1)} - \frac{1}{4(b-1)^2} + \mathcal{O}(b^{-3})$$

(iii) In particular, if q is even we have

$$\begin{aligned}
\lambda_2 = \lambda_1^{(1)} &\approx 1 - \left[2 - \cos\left(\frac{2\pi}{q}\right)\right]\frac{1}{b^2-1} + \frac{1}{4(b-1)^2} + \cos^2\left(\frac{2\pi}{q}\right)\frac{1}{(b+1)^2},\\
\lambda_q = \lambda_1^{(q/2)} &\approx 1 - \frac{3}{b^2-1} + \frac{1}{4(b-1)^2} + \frac{1}{(b+1)^2},\\
\lambda_{q+1} = \lambda_2^{(0)} &\approx \frac{1}{b+1} - \frac{1}{(b+1)^2} - \frac{1}{b^2-1} - \frac{1}{(b-1)^2},\\
\lambda_n = \lambda_2^{(q/2)} &\approx \frac{-2}{b+1} - \frac{1}{(b+1)^2} + \frac{1}{b^2-1} - \frac{1}{(b-1)^2}.
\end{aligned}$$

In other words,

$$1-\frac{C_2}{b^2}<\lambda_2<1, \qquad 1-\frac{C_q}{b^2}<\lambda_q<1,$$

$$0<\lambda_{q+1}<\frac{C_{q+1}}{b}, \qquad -\frac{C_n}{b}<\lambda_n<0,$$

where C_2, C_q, C_{q+1}, and C_n are positive quantities of order 1.

Proof. These are seen through the formulas in Theorem 8 and first-order Taylor expansion of $\sqrt{\beta_k^2+(b-1)}$ about leading asymptotic term $\frac{1}{4}((b^2-1)^2)$ and simplification. Let $\theta=\sqrt{b-1}$ and $\eta=\sqrt{b+1}$. Then

$$\beta_k^2=\frac{1}{4}\left(\frac{\alpha_k\theta}{\eta}-\theta\eta+\frac{\eta}{\theta}\right)^2=\frac{\alpha_k^2\theta^2}{4\eta^2}+\frac{\theta^2\eta^2}{4}+\frac{\eta^2}{4\theta^2}-\frac{\alpha_k\theta^2}{2}+\frac{\alpha_k}{2}-\frac{\eta^2}{2}.$$

The first-order Taylor expansion we employ is $\sqrt{a^2+x}=a+\frac{1}{2}a^{-1}x+\mathcal{O}(a^{-3}x^2)$, which yields

$$\begin{aligned}\sqrt{\beta_k^2+\theta^2} &= \sqrt{\left[\frac{\theta\eta}{2}\right]^2+\left[\left(1-\frac{\alpha_k}{2}\right)\theta^2-\frac{\eta^2}{2}+\frac{\alpha_k^2\theta^2}{4\eta^2}+\frac{\alpha_k}{2}+\frac{\eta^2}{4\theta^2}\right]}\\ &= \frac{\theta\eta}{2}+\frac{1}{\theta\eta}\left[\left(1-\frac{\alpha_k}{2}\right)\theta^2-\frac{\eta^2}{2}+\frac{\alpha_k^2\theta^2}{4\eta^2}+\frac{\alpha_k}{2}+\frac{\eta^2}{4\theta^2}\right]+\mathcal{O}(b^{-2})\\ &= \frac{\theta\eta}{2}+\left(1-\frac{\alpha_k}{2}\right)\frac{\theta}{\eta}-\frac{\eta}{2\theta}+\frac{\alpha_k^2\theta}{4\eta^3}+\frac{\alpha_k}{2\theta\eta}+\frac{\eta}{4\theta^3}+\mathcal{O}(b^{-2}).\end{aligned}$$

Now, we see

$$\begin{aligned}\xi_1^{(k)} &= \beta_k+\sqrt{\beta_k^2+\theta^2}=\frac{1}{2}\left(\frac{\alpha_k\theta}{\eta}-\theta\eta+\frac{\eta}{\theta}\right)+\sqrt{\beta_k+\theta^2}\\ &= \frac{\theta}{\eta}+\frac{\alpha_k^2\theta}{4\eta^3}+\frac{\alpha_k}{2\theta\eta}+\frac{\eta}{4\theta^3}+\mathcal{O}(b^{-2}), \qquad \text{and}\\ \xi_2^{(k)} &= \beta_k-\sqrt{\beta_k^2+\theta^2}=\frac{1}{2}\left(\frac{\alpha_k\theta}{\eta}-\theta\eta+\frac{\eta}{\theta}\right)-\sqrt{\beta_k+\theta^2}\\ &= -\theta\eta+(\alpha_k-1)\frac{\theta}{\eta}+\frac{\eta}{\theta}-\frac{\alpha_k^2\theta}{4\eta^3}-\frac{\alpha_k}{2\theta\eta}-\frac{\eta}{4\theta^3}+\mathcal{O}(b^{-2}).\end{aligned}$$

Then, noting $\eta^{-1} - \theta^{-1} = -2\eta^{-1}\theta^{-1}$, we see

$$\begin{aligned}
\lambda_1^{(k)} &= 1 - \frac{1}{\theta^2} + \frac{\xi_1^{(k)}}{\theta\eta} \\
&= 1 - \frac{1}{\theta^2} + \frac{1}{\eta^2} + \frac{\alpha_k}{2\theta^2\eta^2} + \frac{1}{4\theta^4} + \frac{\alpha_k^2}{4\eta^4} \\
&= 1 - \frac{4-\alpha_k}{2\theta^2\eta^2} + \frac{1}{4\theta^4} + \frac{\alpha_k^2}{4\eta^4} + \mathcal{O}(b^{-3}).
\end{aligned}$$

Substituting in for θ and η gives the result presented in (i). Similarly, we see (ii) via

$$\lambda_2^{(k)} = 1 - \frac{1}{\theta^2} + \frac{\xi_2^{(k)}}{\theta\eta} = \frac{\alpha_k - 1}{\eta^2} - \frac{\alpha_k^2}{4\eta^4} - \frac{\alpha_k}{2\theta^2\eta^2} - \frac{1}{4\theta^4} + \mathcal{O}(b^{-3}).$$

Lastly, (iii) is seen by plugging in for specific values of k and $\alpha_k = 2\cos\left(\frac{2\pi k}{q}\right)$.

$\square$

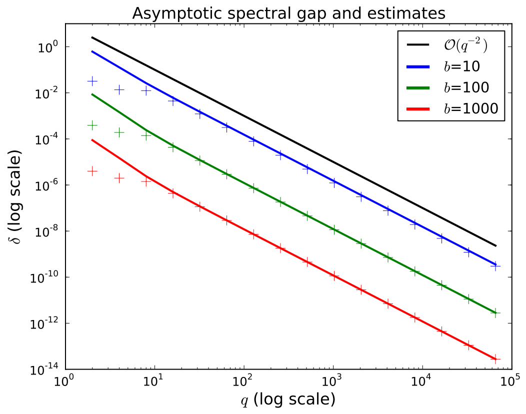
Remark 3. *We observe several facts:*

- *The vector $D^{-1/2}\mathbf{v}_1^{(0)}$ is the constant vector. It causes no errors, but does not help to separate any of the cliques.*
- *The vectors $D^{-1/2}\mathbf{v}_{1,1}^{(k)}$ and $D^{-1/2}\mathbf{v}_{1,2}^{(k)}$ for $k = \{1 \dots \lceil q/2 \rceil - 1\}$ assign the same sign to the corners as the internal members of each block and are associated with positive eigenvalues. Note that we can consider all these eigenvectors as signal eigenvectors,*

$$\mathcal{X}_{signal} = span\left\{\mathbf{v}_1^{(1)}, \mathbf{v}_1^{(2)}, ..., \mathbf{v}_1^{(\lceil q \rceil - 1)}\right\}.$$

  Because $\mathcal{X}_{signal} \perp \mathcal{X}_{noise}$, all sweep cuts of vectors in $\mathcal{X}_{signal}$ keep internal vertices of each clique together.
- *If q is even, the vector $D^{-1/2}\mathbf{v}_2^{(q/2)}$ has values at the corners of opposite sign to the values of the internal vertices and the sign of each corner oscillates around the ring. This is the most oscillatory global eigenvector.*

- *The vectors $D^{-1/2}\mathbf{v}_{2,1}^{(k)}$ and $D^{-1/2}\mathbf{v}_{2,2}^{(k)}$ for $k = \{1 \dots \lceil q/2 \rceil - 1\}$ assign opposite signs to the corners and the internal members of each block. If vectors make large contributions to the blend we compute, then there is potential to misclassify several of the corner vertices.*



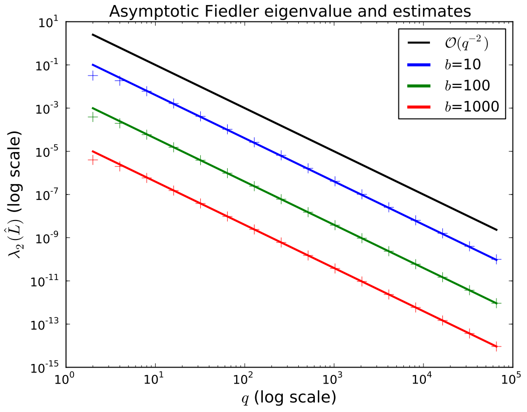


Figure 21: Asymptotic estimates of spectral gaps (left) and Fiedler eigenvalues (right) for rings of cliques with parameters $b = 10, 100, 1000$ and $q = 2, 4, ..., 8096$. Lines represent leading-order terms derived in Theorems 9 and 10 and '+' represent actual eigenvalues as given by the formulas in Theorem 8.

We use the previous result to derive asymptotic estimates of the gaps between eigenvalues near 1 and the size of the Fiedler eigenvalue. These asymptotic estimates are compact formulas in terms of b and q. Figure 21 verifies these estimates empirically.

Theorem 9. (ROC Asymptotic Eigengap) *For the graph $\mathcal{R}_{b,q}$ the spectral gap relevant for computing an eigenvector associated with $\lambda_2(\hat{L})$ is asymptotically $\mathcal{O}(b^{-2}q^{-2})$ for large b and q.*

Proof. Because $\lambda_2(\hat{A}) = \lambda_3(\hat{A})$, the eigenvalues of interest are $\lambda_2(\hat{A}) = \lambda_1^{(1)}$ and $\lambda_4(\hat{A}) = \lambda_1^{(2)}$. We will take Taylor expansions and collect the leading-order terms to understand the asymptotic behavior. Define the scalar function $f(x) = \sqrt{x + a}$, for a constant a that we define below. Using the formulas in the previous result, we see that $\lambda_1^{(1)} - \lambda_1^{(2)} =$

$$\frac{\xi_1^{(1)} - \xi_1^{(2)}}{\sqrt{b^2-1}} = \frac{\beta_1 - \beta_2 + \sqrt{\beta_1^2 + (b-1)} - \sqrt{\beta_2^2 + (b-1)}}{\sqrt{b^2-1}} = \frac{\beta_1 - \beta_2 + f(x_1) - f(x_2)}{\sqrt{b^2-1}} \tag{25}$$

with

$$x_k = \left(\frac{1}{4} - \frac{1}{2(b+1)}\right)\alpha_k^2 + \left(1 - \frac{b}{2}\right)\alpha_k \qquad \text{and} \qquad a = \frac{b^2}{4} + \frac{b}{2} - \frac{3}{2} + \frac{1}{2(b-1)}.$$

We expand the two differences in the numerator of (25) separately, concentrating on the $f(x_1) - f(x_2)$ term first. A first-order Taylor expansion of $f(x)$ at x_1 yields

$$f(x_2) = f(x_1) + f'(x_1)(x_2 - x_1) + \frac{f''(y)}{2}(x_2 - x_1)^2.$$

where $y \in (\min(x_1, x_2), \max(x_1, x_2))$. Rearranging and plugging in, we see

$$f(x_1) - f(x_2) = f'(x_1)(x_1 - x_2) - \frac{f''(y)}{2}(x_1 - x_2)^2 = \frac{x_1 - x_2}{2\sqrt{x_1 + a}} - \frac{(x_1 - x_2)^2}{4(y + a)^{3/2}}.$$

Further, assume $q >> 4\pi k$ and use a third-order Taylor expansion at 0 to see $\alpha_k = 2 - 4(\pi k/q)^2 + \mathcal{O}(q^{-4})$. Similarly, $\alpha_k^2 = 4\cos^2(2\pi k/q) = 2(1 + \cos(4\pi k/q)) = 4 - 16(\pi k/q)^2 + \mathcal{O}(q^{-4})$. Thus,

$$\begin{aligned}(x_1 - x_2) &= \left(\frac{1}{4} - \frac{1}{2(b+1)}\right)\left(\frac{48\pi^2}{q^2}\right) + \left(1 - \frac{b}{2}\right)\left(\frac{12\pi^2}{q^2}\right) + \mathcal{O}\left(bq^{-4}\right) \\ &= \frac{-6\pi^2 b}{q^2} + \frac{24\pi^2}{q^2} + \mathcal{O}\left(bq^{-4}\right).\end{aligned}$$

Expansion of $\beta_1 - \beta_2$ using the cosine Taylor expansions shows

$$\beta_1 - \beta_2 = \frac{\alpha_1 - \alpha_2}{2}\sqrt{\frac{b-1}{b+1}} = \frac{6\pi^2}{q^2}\sqrt{\frac{b-1}{b+1}} + \mathcal{O}(q^{-4}).$$

Lastly, $(y + a)^{3/2}$ is $\mathcal{O}(b^3)$ and $(x_1 + a)^{1/2}$ is $\mathcal{O}(b)$, so

$$\lambda_1^{(1)} - \lambda_1^{(2)} = \frac{6\pi^2}{q^2(b+1)} + \frac{-3\pi^2 b}{q^2\sqrt{x_1 + a}\sqrt{b^2-1}} + \frac{12\pi^2}{q^2\sqrt{x_1 + a}\sqrt{b^2-1}} + \mathcal{O}(b^{-1}q^{-4}).$$

As $b \to \infty$ we see $2b^{-1}\sqrt{x_1 + a} \to 1$, so the first two terms cancel asymptotically and the third term is $\mathcal{O}\left(q^{-2}b^{-2}\right)$. □

Theorem 10. (ROC Asymptotic Fiedler Eigenvalue) *Let b and q be large and the same order. For graph $\mathcal{R}_{b,q}$ the smallest nonzero eigenvalue of $\hat{L}$ is $\mathcal{O}(b^{-2}q^{-2})$.*

Proof. The eigenvalue of interest is $\lambda_2(\hat{L}) = 1 - \lambda_2(\hat{A}) = 1 - \lambda_1^{(1)}$. Define scalar function $g(z) = \sqrt{z + 1/4}$. Using the formulas in Theorem 8, we see

$$1 - \lambda_1^{(1)} = \frac{1}{b-1} - \frac{\xi_1^{(1)}}{\sqrt{b^2-1}} = \frac{1}{b-1} - \frac{\beta_1 + \sqrt{\beta_1^2 + (b-1)}}{\sqrt{b^2-1}} = \frac{1}{b-1} - \frac{\beta_1}{\sqrt{b^2-1}} - g(z) \tag{26}$$

with

$$z = \frac{(1-\alpha_1)b}{2(b^2-1)} + \frac{2\alpha_1 - 3}{2(b^2-1)} + \frac{1}{4(b-1)^2} + \frac{\alpha_1^2}{4(b+1)^2}.$$

We derive the result by demonstrating that the larger terms in (26) cancel. Expanding the second term in in the right-hand-side yields

$$\frac{\beta_1}{\sqrt{b^2-1}} = \frac{\alpha_1}{2(b+1)} - \frac{1}{2} + \frac{1}{2(b-1)} = -\frac{1}{2} + \frac{(1+\alpha_1)b}{2(b^2-1)} + \frac{1-\alpha_1}{2(b^2-1)}.$$

A second-order Taylor expansion of $g(z)$ at zero shows for each z, we see $g(z) = g(0) + g'(0)z + \frac{1}{2}g''(0)z^2 + \frac{1}{6}g'''(0)z^3 + \frac{1}{24}g^{(iv)}(y)z^4 = \frac{1}{2} + z - z^2 + 2z^3 + \mathcal{O}(z^4)$, where

$$g'(z) = \frac{1}{2\sqrt{z+1/4}}, \qquad g''(z) = -\frac{1}{4(z+1/4)^{3/2}}, \qquad \text{and} \qquad g'''(z) = \frac{3}{8(z+1/4)^{5/2}}.$$

Plugging in, we see the terms of $g(z)$ up to order b^{-2} are

$$\frac{1}{2} + \frac{(1-\alpha_1)b}{2(b^2-1)} + \frac{2\alpha_1-3}{2(b^2-1)} + \frac{1}{4(b-1)^2} + \frac{\alpha_1^2}{4(b+1)^2} - \frac{(1-\alpha_1)^2 b^2}{4(b^2-1)^2}.$$

The constant terms in $\beta_1/\sqrt{b^2-1}$ and $g(z)$ cancel. The order b^{-1} terms in $1-\lambda_1^{(1)}$ cancel to an order b^{-2} term,

$$\frac{1}{b-1} - \frac{(1+\alpha_1)b}{2(b^2-1)} - \frac{(1-\alpha_1)b}{2(b^2-1)} = \frac{1}{b-1} - \frac{b}{b^2-1} = \frac{1}{b^2-1}.$$

Combining fractions, the order b^{-2} terms in $1 - \lambda_1^{(1)}$ are reduced to

$$\begin{aligned} &\frac{1}{b^2-1} - \left[\frac{1-\alpha_1}{2(b^2-1)} + \frac{2\alpha_1-3}{2(b^2-1)} + \frac{1}{4(b-1)^2} + \frac{\alpha_1^2}{4(b+1)^2} - \frac{(1-\alpha_1)^2 b^2}{4(b^2-1)^2}\right] \\ = \quad & \frac{(2-\alpha_1)b^2}{(b^2-1)^2} + \frac{(2\alpha_1^2-3)b}{2(b^2-1)^2} + \frac{-\alpha_1^2 + 2\alpha_1 - 9}{4(b^2-1)^2} \end{aligned}$$

Factoring in the other b^{-3} terms and the cosine expansion $\alpha_1 = 2 - 4(\pi/q)^2 + \mathcal{O}(q^{-4})$, we see

$$\frac{4\pi^2 b^2}{q^2(b^2-1)^2} + \mathcal{O}\left(b^{-4} + b^{-2}q^{-4}\right).$$

□

Theorem 11. *For any vector* $\mathbf{x} \in \mathcal{X}_{signal}$ *of* $\mathcal{R}_{b,q}$*,* $\phi(\mathbf{y}) \leq q\,\phi(\mathbf{v}_2)$*.*

Proof. For any $\mathbf{x} \in \mathcal{X}_2$, let $S, \bar{S}$ be the partition given by the optimal sweep cut of $D^{-\frac{1}{2}}\mathbf{x}$. Fiedler's nodal domain theorem implies at least one of $S, \bar{S}$ is a connected component. Because the eigenvectors are block constant, all vertices of each clique are assigned to the same side of the partition. These imply that $E(S, \bar{S}) = 2$. The optimal conductance partition is found when $\mathrm{vol}(S) = \mathrm{vol}(\bar{S}) = q\binom{b}{2}$. Thus $\phi_G = \phi(\mathbf{v}) = 2(q\binom{b}{2})^{-1}$.

For any $\mathbf{x} \in \mathcal{X}_{signal}$, the optimal sweep cut of $D^{-\frac{1}{2}}\mathbf{x}$ will partition the graph into two pieces one containing k blocks and the other containing $n-k$ blocks for some $k \leq \frac{n}{2}$. That is, $\min(\mathrm{vol}(S), \mathrm{vol}(\bar{S})) = k\binom{b}{2}$. Since only edges between cliques are cut, $E(S, \bar{S}) \leq 2k$. Thus $\phi(\mathbf{y}) \leq 2\binom{b}{2}^{-1}$. By assigning adjacent blocks to alternating sides of the partition, we see that the bound is tight.

□

Notice that the smallest eigenvalue of $\hat{L}$ scales as $\mathcal{O}(b^{-2}q^{-2})$, but the optimal conductance scales as $\mathcal{O}(b^{-2}q^{-1})$, and that the worst case sweep cut partition of a blend has conductance $2\binom{b}{2}^{-1}$ independent of q. The remainder of this section shows that by accepting this factor of q in conductance, one gains tremendously in computational efficiency.

6.3.3 A residual tolerance for ring of cliques

In order to derive a residual tolerance for the ring of cliques, we show that for any vector in $Span\{\mathbf{v}_1 \ldots \mathbf{v}_q\}$ at least one block is sufficiently far from the other blocks in spectral coordinates.

Lemma 4. *Define $\mathcal{B}_i = \{ib+2 \dots ib+q\}$ as the vertex blocks of $\mathcal{R}_{b,q}$. For any vector $\mathbf{x}$, let $\mu_j = |\mathcal{B}_i|^{-1} \sum_{i \in \mathcal{B}_j} x_i$. Let $\mathcal{W}$ be the span of $\{\mathbf{e}_{ib+1} \mid i \in 0 \dots q-1\} \cup \{\mathbf{e}_{\mathcal{B}_i} \mid i \in 0 \dots q-1\}$. For any vector $\mathbf{x} \in \mathcal{W}$, $\|\boldsymbol{\mu}\|_\infty > n^{-1/2} \|\mathbf{x}\|_2$.*

Proof. By construction of $\mathbf{x}$, $\mu_i = x_j$ for all $j \in \mathcal{B}_i$. Thus $\|\mathbf{x}\|_\infty = \|\boldsymbol{\mu}\|_\infty$. Equivalence of norms implies $\|\mathbf{x}\|_2 < \sqrt{n}\mu_q$. □

We are able to apply Theorem 6 and derive a residual tolerance for recovering the ring of cliques.

Corollary 2. *If $\mathbf{x}$ is an approximate eigenvector of $\mathcal{R}_{b,q}$ with eigenresidual less than $\frac{C}{q\sqrt{n}}$ for some constant C and $\mathbf{x}^T\mathbf{1} = 0$, then $\phi\left(D^{-\frac{1}{2}}\mathbf{x}\right) \leq 2\binom{b}{2}^{-1}$*

Proof. In the setting of Theorem 6, choose $G = \mathcal{R}_{b,q}$, $\psi = 2\binom{b}{2}^{-1}$. In the notation of Lemma 4 applied to sorted $\mathbf{v}$, for all $\mathbf{v} \in V$, $g_v = \max_i(\mu_{i+1} - \mu_i) \geq q^{-1} \|\boldsymbol{\mu}\|_\infty \geq (q\sqrt{n})^{-1}$. For some $C \in \mathcal{O}(1)$, $\delta_{p,q}(\mu)g > \frac{C}{q\sqrt{n}}$. So Theorem 6 implies computing $\mathbf{x}$ to a residual tolerance of $\frac{C}{q\sqrt{n}}$ is sufficient to guarantee $\phi(\mathbf{x}) \leq 2\binom{b}{2}^{-1}$. □

Corollary 2 gives a sufficient condition on approximate eigenvectors of $\mathcal{R}_{b,q}$ such that $\mathbf{x}$ partitions the graph at least as well as any partition that recovers the cliques. Theorem 12 and Theorem 13 using analysis specialized for $\mathcal{R}_{b,q}$ in Section 6.3.4 to construct the minimal perturbation that causes the sweep cut proceedure to fail.

6.3.4 Minimal Perturbation

We want to find the minimal error at which a vector can make a mistake. The effects of the corner vertices only enter into the constants of the following results, and for clarity of exposition we omit handling of the corner vertices. Theorem 12 shows that no perturbation with norm less than $(1+2qn)^{-\frac{1}{2}}$ can induce a mistake in the sweep cut. Theorem 13 constructs a perturbation that induces a mistake in the sweep cut and has norm less than $b^{-\frac{1}{2}}$. For the parameter regime $q \in \mathcal{O}(1)$, the bounds in Corollary 2, Theorem 12, and Theorem 13 are all equivalent up to constant factors.

Using the same notation as Lemma 4, we say that a vector $\mathbf{y}$ recovers all the cliques of $\mathcal{R}_{b,q}$ if there is a threshold $t \in (\min_i y_i, \max_i y_i)$ such that for all $\mathcal{B}_i$ with $j, k \in \mathcal{B}_i$, $y_j < t$ if and only if $y_k < t$.

Theorem 12. *Let $\mathcal{W}$ be defined as in Lemma 4, and let $\mathbf{P}_{\mathcal{W}}$ be the orthogonal projector onto $\mathcal{W}$. For any vector $\mathbf{y}$ orthogonal to $\mathbf{1}$, define $\mathbf{z} = (\mathbf{I} - \mathbf{P}_{\mathcal{W}})\mathbf{y}$. If $\|\mathbf{z}\|_2 \leq (1 + 2qn)^{-\frac{1}{2}} \|\mathbf{y}\|_2$, then $\mathbf{y}$ recovers all the cliques of $\mathcal{R}_{b,q}$.*

Proof. Define $\mathbf{x} = \mathbf{P}_{\mathcal{W}}\mathbf{y}$. Without loss of generality, relabel the vertices and blocks such that $\mu_i \leq \mu_{i+1}$. Let $\alpha_i = \max_i z_i$ and $\beta_i = \min_i z_i$ for each $\mathcal{B}_i$. Note that $\alpha_i > 0$ and $\beta_i < 0$ since $\mathbf{z} \perp \mathbf{e}_{\mathcal{B}_i}$. The vector $\mathbf{y}$ recovers all the cliques if and only if there is a $\mathcal{B}_i$ where $\alpha_i - \beta_{i+1} \leq \mu_{i+1} - \mu_i$. In this case, a threshold can be chosen in $(\mu_i + \alpha_j, \mu_{i+1} + \beta_k)$. Suppose that $\mathbf{y}$ does not recover all the cliques, then for all $\mathcal{B}_i$ $\alpha_i - \beta_{i+1} > \mu_{i+1} - \mu_i$. This implies $\sum_{i=0}^{q-1} \alpha_i - \beta_{i+1} > \sum_{i=0}^{q-1} \mu_{i+1} - \mu_i$. Thus we can bound the 1-norm error as follows:

$$\|\mathbf{z}\|_1 \geq \sum_i (\alpha_i - \beta_{i+1}) \geq \mu_q - \mu_1 \geq n^{-\frac{1}{2}} \|\mathbf{x}\|_2 .$$

Since $\mathbf{z}$ must have at least q nonzero entries $\|\mathbf{z}\|_2 > (2qn)^{-\frac{1}{2}} \|\mathbf{x}\|_2$. Applying $\|\mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{z}\|^2$, we see that $(1 + 2qn)^{-\frac{1}{2}} \|\mathbf{y}\| < \|\mathbf{z}\|_2$.

□

The proof of Theorem 12 yields a construction for the minimal perturbation of $\mathbf{x}$ that does not recover all the cliques.

Theorem 13. *For any unit vector $\mathbf{x} \in \mathcal{W}$ orthogonal to $\mathbf{1}$, there exists a perturbation $\mathbf{z}$ where $\|\mathbf{z}\| < b^{-\frac{1}{2}}, \mathbf{P}_{\mathcal{W}}\mathbf{z} = 0$ such that $\mathbf{y} = \mathbf{x} + \mathbf{z}$ does not recover all the cliques.*

Proof. For any $\mathbf{x} \in \mathcal{W}$, set $\alpha_0 = 0, \beta_{q-1} = 0, \alpha_i = -\beta_{i+1} = \frac{\mu_{i+1} - \mu_i}{2}$.

$$\|\mathbf{z}\|_2^2 = \sum_{i=0}^{q-1} \alpha_i^2 + \beta_{i+1}^2 = \frac{1}{2} \sum_{i=0}^{q-1} (\mu_{i+1} - \mu_i)^2 = b^{-1} \|\mathbf{x}\|_2^2 - 2 \sum_{i=0}^{q-1} \mu_{i+1}\mu_i < b^{-1} \|\mathbf{x}\|_2^2$$

□

Theorem 12 implies that $(1+2qn)^{-\frac{1}{2}}$ is a sufficient accuracy to ensure recovery of all the cliques, and Theorem 13 implies that for some elements of the top invariant subspace accuracy less than $b^{-\frac{1}{2}}$ is necessary to ensure recovery of all the cliques from that vector.

Figure 22 lends validation to the formulas in Theorem 12. The experiment shown is to take a random (Gaussian unit norm) linear combination of $\mathcal{X}_{signal}$, and then construct the minimal perturbation that makes a mistake. Figure 22 shows the minimum over all samples as a function of n. This experiment is conducted for three different parameter regimes, $q = 25$, $b = 25$, and $b = q = \sqrt{n}$. One can see that the lower bound from Theorem 12 is below the empirical observation, and that this lower bound is within a constant factor of the observed size of the minimal perturbation.

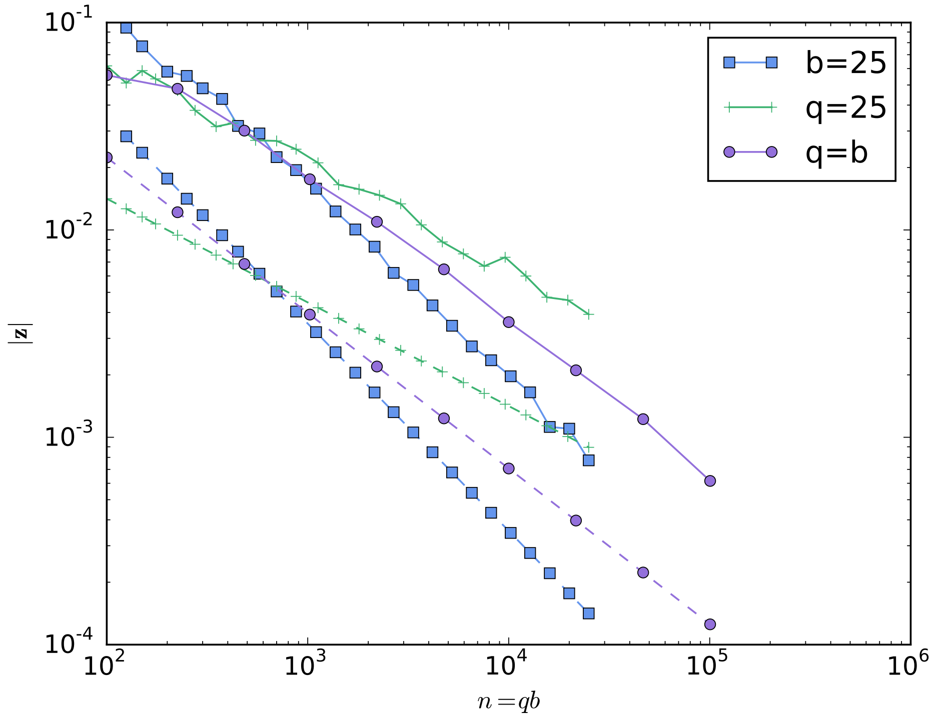


Figure 22: Empirical measurements of minimal error perturbations on a log-log scale. Lower bounds are shown in the same color with dashed lines.

We can now apply Theorem 4 and Theorem 5 to determine the residual tolerance for an eigensolver for graph partitioning. The residual tolerance can be no larger than that for the the ring of cliques, but some graphs may require smaller tolerances.

We can specialize the above analysis for the Fiedler vector in order to derive error and residual tolerances for recovering the optimal partition. A Fiedler vector of $\mathcal{R}_{b,q}$ is a shifted version of $\mathbf{v}_2 = \mathbf{c}_q \otimes \mathbf{1}_b$. The block means for the fiedler vector are proportional to $\mu_i = \cos\frac{2\pi i}{q}$, or a phase shifted version of this vector. Once Fiedler vector is sorted the block means are $\mu_i = \cos\left(\frac{2\pi}{q}\lceil\frac{i}{2}\rceil\right)$ where i ranges over $\{q, q+1\ldots 2q-1\}$.

Theorem 14. *For $\mathbf{x} = \mathbf{v}_2/|0\mathbf{v}_2|0$, which is a unit norm Fiedler vector of $\mathcal{R}_{b,q}$, there exists a $\mathbf{z}$ satisfying Equation (27) such that any sweep cut of $\mathbf{z}+\mathbf{v}_2$ makes a mistake.*

$$\|\mathbf{z}\|^2 = 2b^{-1}\sin^2\left(\frac{\pi}{q}\right) \tag{27}$$

Proof. We examine

$$\mu_{i+1} - \mu_i = \left(\cos\frac{2\pi(i+1)}{q} - \cos\frac{2\pi i}{q}\right)^2$$

and the sum

$$\sum_{i=q}^{2q-1}(\mu_{i+1}-\mu_i)^2 = \left(\cos\frac{2\pi(i+1)}{q} - \cos\frac{2\pi i}{q}\right)^2$$

The fact that $\lceil\frac{i}{2}\rceil = \lceil\frac{i+1}{2}\rceil$ implies $\mu_{i+1} = \mu_i$ allows us to reparameterize the above sum as

$$\sum_{i=\lceil q/2\rceil}^{q-1} = \left(\cos\frac{2\pi\lceil i+1/2\rceil}{q} - \cos\frac{2\pi i}{q}\right)^2 = \sum_{\lceil q/2\rceil}^{q-1}\left(\cos\frac{2\pi(i+1)}{q} - \cos\frac{2\pi i}{q}\right)^2 = q\sin^2\frac{\pi}{q}$$

Thus there exists a $\mathbf{z}$, $\|\mathbf{z}\|^2 = q\sin^2\left(\frac{\pi}{q}\right)$ such that any sweep cut of $\mathbf{z}+\mathbf{v}_2$ makes a mistake. Normalization of $\mathbf{x}$ yields the theorem. □

The vector in Theorem 14 is minimal in 1-norm by construction. Similar but more involved algebra could be used to find the minimal 2 norm perturbation. Table 7 verifies these equalities on a range of problems using the formulas in Theorem 8. This leads to relative errors on the order of 10^{-16}. Table 8 compares the error threshold computed by Equation (27) to the error bound found by numerical simulation. One can solve the problem using `ARPACK`, and then compute the necessary error to perturb the provided solution. The relative errors of Table 8 are on the order of 10^{-3}, which is sufficiently small considering the value is computed by taking the sum of squared differences in an eigenvector computed to an accuracy of $10^{-15} \times n^2$.

An error tolerance quaranteeing the optimal partitioning of $\mathcal{R}_{b,q}$ is $b^{-1/2}\sin(\frac{\pi}{q})$. The residual error bound $\epsilon < \sqrt{8}b^{-1/2}\sin(\frac{\pi}{q})\delta_2 = \mathcal{O}\left(n^{-5/2}q^{-1/2}\right)$ is a residual tolerance guaranting optimal partitioning of $\mathcal{R}_{b,q}$ using spectral sweep cuts. Even with a solver requiring $\mathcal{O}\left(\log \epsilon^{-1}\right)$ iterations, this is $\mathcal{O}\left(n\right)$ iterations. In contrast the residual tolerance for recovering the cliques is $\mathcal{O}\left(q^{-\frac{1}{2}}n^{-\frac{1}{2}}\right)$. The next section shows how the larger residual tolerance required by partitioning $\mathcal{R}_{b,q}$ with spectral blends leads to fewer iterations of an iterative solver.

6.3.5 The Power Method

From the eigenvalues and error tolerances above, one can determine an upper bound on the number of iterations required by the power method to recover all the cliques in $\mathcal{R}_{b,q}$.

Theorem 15. *Let $\mathbf{x}_0$ be sampled from $\mathcal{N}_n(0,1)$. Let $\mathbf{x}_k$ be the k-th iteration of the power method, $\mathbf{x}_{k+1} \longleftarrow \hat{A}\mathbf{x}_k / \left\|\mathbf{x}_k\right\|$ for $\mathcal{R}_{b,q}$. Let $\zeta = \left(\frac{e^{7/8}}{8}\right)^{q/2} + \left(\frac{2}{e}\right)^{(n-q)/2}$. There is a k^* of $\mathcal{O}\left(\log_b q\right)$ such that for $k \geq k^*$ a sweep cut based on $\mathbf{x}_k$ makes no errors with probability at least $1-\zeta$.*

Proof. First, we bound $\left\|(\mathbf{I} - \mathbf{P}_{\mathcal{W}})\mathbf{x}_0\right\|^2$ and $\left\|\mathbf{P}_{\mathcal{W}}\mathbf{x}_0\right\|^2$ probabilistically. Each entry in $\mathbf{x}_0$ is independently sampled from $\mathcal{N}(0,1)$. For any orthonormal basis of $\mathbb{R}^n$, $\{\mathbf{v}_k\}_{k=1}^n$, the distribution of each $\mathbf{v}_k^T\mathbf{x}_0$ is also $\mathcal{N}(0,1)$. Therefore, the distribution of $\left\|\mathbf{P}_{\mathcal{W}}\mathbf{x}_0\right\|^2$ is a χ^2-distribution of order q, which has expected value q and cumulative distribution

Table 7: Validation of Theorem 14 by comparing to formulas in Theorem 8.

n	q	b	$\epsilon_{observed}$	$\epsilon_{formula}$	relerror
21	3	7	4.6291e-01	4.6291e-01	-2.2204e-16
60	3	20	2.7386e-01	2.7386e-01	-4.4409e-16
80	4	20	2.2361e-01	2.2361e-01	0.0000e+00
120	6	20	1.5811e-01	1.5811e-01	-2.2204e-16
126	9	14	1.2927e-01	1.2927e-01	-6.6613e-16
160	8	20	1.2102e-01	1.2102e-01	-4.4409e-16
160	8	20	1.2102e-01	1.2102e-01	-4.4409e-16
240	12	20	8.1846e-02	8.1846e-02	-4.4409e-16
320	16	20	6.1693e-02	6.1693e-02	-6.6613e-16
360	18	20	5.4912e-02	5.4912e-02	-1.3323e-15
384	24	16	4.6148e-02	4.6148e-02	-1.1102e-15
520	26	20	3.8117e-02	3.8117e-02	-2.2204e-15
567	27	21	3.5827e-02	3.5827e-02	-1.9984e-15
576	24	24	3.7680e-02	3.7680e-02	-8.8818e-16
640	32	20	3.0996e-02	3.0996e-02	-1.7764e-15
768	24	32	3.2632e-02	3.2632e-02	-1.3323e-15
780	39	20	2.5446e-02	2.5446e-02	-2.6645e-15
960	24	40	2.9187e-02	2.9187e-02	-1.3323e-15
1152	24	48	2.6644e-02	2.6644e-02	-1.1102e-15
1160	58	20	1.7120e-02	1.7120e-02	-3.5527e-15
1280	64	20	1.5517e-02	1.5517e-02	-3.5527e-15
1344	24	56	2.4667e-02	2.4667e-02	-1.1102e-15
1536	24	64	2.3074e-02	2.3074e-02	-1.1102e-15
1728	24	72	2.1754e-02	2.1754e-02	-8.8818e-16
1920	24	80	2.0638e-02	2.0638e-02	-1.3323e-15
2112	24	88	1.9678e-02	1.9678e-02	-1.3323e-15
2268	81	28	1.0363e-02	1.0363e-02	-6.2172e-15
2304	24	96	1.8840e-02	1.8840e-02	-8.8818e-16
2560	128	20	7.7606e-03	7.7606e-03	-6.6613e-15
5120	256	20	3.8806e-03	3.8806e-03	-1.2657e-14
10240	512	20	1.9403e-03	1.9403e-03	-2.5757e-14
20480	1024	20	9.7017e-04	9.7017e-04	-4.9516e-14
40960	2048	20	4.8509e-04	4.8509e-04	-9.9032e-14
81920	4096	20	2.4254e-04	2.4254e-04	-1.9362e-13
163840	8192	20	1.2127e-04	1.2127e-04	-3.8591e-13
327680	16384	20	6.0636e-05	6.0636e-05	-7.6650e-13

Table 8: Validation of Theorem 14 by comparing to vectors provided by `ARPACK`.

n	q	b	$\epsilon_{observed}$	$\epsilon_{formula}$	relerror
60	3	20	2.4104e-01	2.7386e-01	1.1983e-01
80	4	20	2.0519e-01	2.2361e-01	8.2368e-02
120	6	20	1.4880e-01	1.5811e-01	5.8876e-02
160	8	20	1.1670e-01	1.2102e-01	3.5656e-02
240	12	20	8.0070e-02	8.1846e-02	2.1702e-02
300	30	10	4.6069e-02	4.6747e-02	1.4487e-02
360	18	20	5.4157e-02	5.4912e-02	1.3747e-02
450	30	15	3.7788e-02	3.8168e-02	9.9635e-03
520	26	20	3.7777e-02	3.8117e-02	8.9165e-03
592	37	16	2.9762e-02	2.9983e-02	7.3936e-03
600	30	20	3.2804e-02	3.3055e-02	7.5749e-03
750	30	25	2.9385e-02	2.9565e-02	6.1059e-03
780	39	20	2.5302e-02	2.5446e-02	5.6318e-03
888	37	24	2.4359e-02	2.4481e-02	4.9988e-03
900	30	30	2.6851e-02	2.6989e-02	5.1127e-03
1050	30	35	2.4877e-02	2.4987e-02	4.3968e-03
1160	58	20	1.7057e-02	1.7120e-02	3.6767e-03
1184	37	32	2.1121e-02	2.1201e-02	3.7728e-03
1200	30	40	2.3283e-02	2.3373e-02	3.8565e-03
1350	30	45	2.1961e-02	2.2037e-02	3.4343e-03
1480	37	40	1.8906e-02	1.8963e-02	3.0290e-03
1500	30	50	2.0841e-02	2.0906e-02	3.0953e-03
1650	30	55	1.9877e-02	1.9933e-02	2.8172e-03
1776	37	48	1.7267e-02	1.7311e-02	2.5299e-03
2072	37	56	1.5992e-02	1.6027e-02	2.1719e-03
2368	37	64	1.4963e-02	1.4992e-02	1.9026e-03
2664	37	72	1.4110e-02	1.4134e-02	1.6927e-03
2960	37	80	1.3389e-02	1.3409e-02	1.5245e-03
3256	37	88	1.2767e-02	1.2785e-02	1.3867e-03
3552	37	96	1.2225e-02	1.2241e-02	1.2717e-03

function $\gamma(q/2, z/2)/\Gamma(q/2)$, where $\Gamma(\cdot)$ is the gamma function and $\gamma(\cdot,\cdot)$ is the lower incomplete gamma function. Let $c_0 \in (0,1)$, using Chernoff bounds we have

$$p_0 := \text{Prob}\left[\|\mathbf{P}_{\mathcal{W}}\mathbf{x}_0\|^2 > c_0 q\right] = 1 - \frac{\gamma\left(\frac{q}{2}, \frac{c_0 q}{2}\right)}{\Gamma\left(\frac{q}{2}\right)} \geq 1 - \left(c_0 e^{1-c_0}\right)^{q/2}$$

Similarly, $\|(\mathbf{I} - \mathbf{P}_{\mathcal{W}})\mathbf{x}_0\|^2$ is from a χ^2-distribution of order order $n-q$, with expected value $n-q$ and known cumulative distribution function. Let $c_1 \in (1, \infty)$, we have

$$p_1 := \text{Prob}\left[\|\mathbf{P}_{\mathcal{W}}\mathbf{x}_0\|^2 < c_1(n-q)\right] = \frac{\gamma\left(\frac{n-q}{2}, \frac{c_1(n-q)}{2}\right)}{\Gamma\left(\frac{n-q}{2}\right)} \geq 1 - \left(c_1 e^{1-c_1}\right)^{(n-q)/2}$$

The union of events $\left[\|\mathbf{P}_{\mathcal{W}}\mathbf{x}_0\|^2 > c_0 q\right]$ and $\left[\|\mathbf{P}_{\mathcal{W}}\mathbf{x}_0\|^2 < c_1(n-q)\right]$ is a subset of all possibilities for which $\left[\frac{\|(\mathbf{I}-\mathbf{P}_{\mathcal{W}})\mathbf{x}_0\|^2}{\|\mathbf{P}_{\mathcal{W}}\mathbf{x}_0\|^2} < \frac{c_1(n-q)}{c_0 q}\right]$ holds. Therefore, setting $c_0 = 1/8$ and $c_1 = 2$, we see

$$\text{Prob}\left[\frac{\|(\mathbf{I}-\mathbf{P}_{\mathcal{W}})\mathbf{x}_0\|^2}{\|\mathbf{P}_{\mathcal{W}}\mathbf{x}_0\|^2} < \frac{c_1(n-q)}{c_0 q}\right] > p_0 p_1 > 1 - \zeta$$

where $\zeta := \left(\frac{e^{7/8}}{8}\right)^{q/2} + \left(\frac{2}{e}\right)^{(n-q)/2}$ is a small positive constant when $q, b > 4$. Because a sweep cut does not depend on the norm of a vector, we consider the iteration, $\mathbf{x}_k \longleftarrow \frac{1}{\lambda_q}\hat{A}\mathbf{x}_{k-1}$ which is equivalent to the power method. Letting $\lambda^* = \max(|\lambda_{q+1}|, |\lambda_n|)$, this iteration accentuates vector components in the range of $\mathbf{P}_{\mathcal{W}}$ by a factor greater than 1 and attenuates those orthogonal to this space by factors less than λ^*/λ_q. If $\|\mathbf{P}_{\mathcal{W}}\mathbf{x}_0\|^2 > c_0 q$, then

$$\|\mathbf{x}_k\|^2 \geq \|\mathbf{P}_{\mathcal{W}}\mathbf{x}_k\|^2 \geq \|\mathbf{P}_{\mathcal{W}}\mathbf{x}_0\|^2 \geq c_0 q.$$

Also, if $\|(\mathbf{I}-\mathbf{P}_{\mathcal{W}})\mathbf{x}_0\|^2 \leq c_1(n-q)$, then

$$\|(\mathbf{I}-\mathbf{P}_{\mathcal{W}})\mathbf{x}_k\|^2 \leq \left(\frac{\lambda^*}{\lambda_q}\right)^{2k} \|(\mathbf{I}-\mathbf{P}_{\mathcal{W}})\mathbf{x}_0\|^2 \leq \left(\frac{\lambda^*}{\lambda_q}\right)^{2k} c_1(n-q).$$

Therefore, under the assumptions on $\mathbf{x}_0$, the k-th iteration satisfies

$$\frac{\|(\mathbf{I}-\mathbf{P}_{\mathcal{W}})\mathbf{x}_k\|}{\|\mathbf{x}_k\|} \leq \left(\frac{\lambda^*}{\lambda_q}\right)^k \sqrt{\frac{c_1}{c_0}(b-1)} = 4\left(\frac{\lambda^*}{\lambda_q}\right)^k \sqrt{b-1}.$$

By Theorem 12, if this ratio is less than $(1+2qn)^{-1/2}$, then $\mathbf{x}_k$ makes no errors. We see this is ensured by

$$k \geq k^* := \left\lceil \frac{\log 4 + \log(b-1) + \log(1+2qn)}{2\log\left(\lambda_q/\lambda^*\right)} \right\rceil.$$

Revisiting Corollary 1 we that $\lambda_q > 1 - C_q/b^2$ and $\lambda^* < \max(C_{q+1}, C_n)/b$ so $\lambda_q/\lambda^* = C^*b$, where C^* is an order 1 constant. Plugging this in we see that k^* is in $\mathcal{O}\left(\log_b q\right)$.

□

6.3.6 Experiment

Here we show the results of a numerical experiment in order to lend intuition and validation to the theorems. Take $\mathcal{R}_{b,q}$ and a random seed vector $\mathbf{x}^{(0)}$. Then apply the power iteration $\mathbf{x}^{(i)} = (\hat{A} - \mathbf{k}\mathbf{k}^t)\mathbf{x}^{(i-1)}$. Far $b = 20$ and $q = 30$ the relevant measures of convergence are shown in Table 9. Figure 23 illustrates the convergence behavior in terms of the conductance of all sweep cuts, and the reordered adjacency matrix represented in sparsity plots. Table 9 shows that the convergence to the Fiedler vector stalls after iteration 3, but convergence to the space orthogonal to $\mathcal{X}_{noise}$ continues unabated. Letting Π be the projection onto $\mathcal{X}_{noise}^{\perp}$, we measure $\left\|\Pi\mathbf{x}^{(i)}\right\|$ for each iteration. Applying Theorem 15, we calculate that $k^* = 5$ iterations will perfectly resolve the clique structure with probability at least $1 - \zeta = 0.99999998575$. After one iteration the sweep cut did not split any cliques, but only a single clique is shaved off. After 3 iterations a nearly optimal partition is found.

6.4 Conclusions

When partitioning graphs where the spectral gap is small, computation of an accurate approximation to a Fiedler vector is difficult. In order to satisfy the needs of spectral partitioning without computing eigenvectors to high accuracy, we introduce spectral

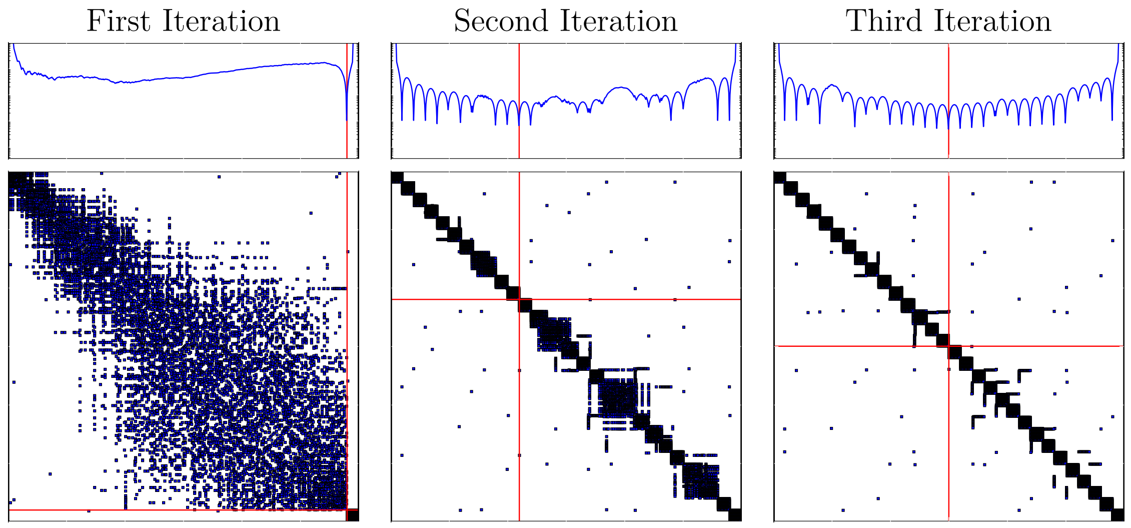


Figure 23: First several iterations of the power method applied to $\mathcal{R}_{b=20,q=30}$. Above: sweep conductance of A reordered by sorting the 1^{st} (top-left), 2^{nd} (top middle), and 3^{rd} iterations (top-right). Horizontal axis represents which vertex to split at under the induced ordering; vertical axis is the conductance for each split on a log scale. Below: matrix sparsity plots of A reordered by sorting the 1^{st} (bottom-left), 2^{nd} (bottom-middle), and 3^{rd} iterations (bottom-right). Red lines demonstrate which edges are cut for the optimal cut in each ordering.

Table 9: Table corresponding to Figure 23 with 10 iterations. Convergence to the Fiedler eigenpair is slow, yet convergence to the orthogonal complement of $\mathcal{X}_{noise}$ is rapid (column 2).

i	$\lVert\mathbf{x}^{(i)} - \mathbf{P}_2\mathbf{x}^{(i)}\rVert$	$\lVert\Pi\mathbf{x}^{(i)}\rVert$	μ	$\phi(\mathbf{x})$	$\sqrt{2\mu}$
0	1.24806e+01	2.25433e+01	1.52966e+00	2.03094e+00	1.74909e+00
1	9.20534e-01	5.02679e-02	8.37276e-01	1.05263e-02	1.29404e+00
2	1.02629e-01	1.45977e-02	5.67751e-02	6.68577e-03	3.36972e-01
3	1.08242e-02	8.10095e-04	1.01284e-02	4.89853e-03	1.42326e-01
4	1.01568e-02	4.28591e-05	9.87761e-03	4.89853e-03	1.40553e-01
5	1.01262e-02	2.26698e-06	9.85062e-03	4.89853e-03	1.40361e-01
6	1.01022e-02	1.19907e-07	9.82549e-03	4.89853e-03	1.40182e-01
7	1.00782e-02	6.34217e-09	9.80038e-03	4.89853e-03	1.40003e-01
8	1.00543e-02	3.35448e-10	9.77526e-03	4.89853e-03	1.39823e-01
9	1.00303e-02	1.77422e-11	9.75013e-03	4.89853e-03	1.39643e-01
10	1.00063e-02	9.38388e-13	9.72498e-03	4.89853e-03	1.39463e-01
...	...	...	...	...	...
∞	0.00000e+00	0.00000e+00	1.14176e-04	3.50018e-04	1.51113e-02
			$\lambda_2(\hat{L})$	$\phi(\mathbf{v}_2) = \phi_{\mathcal{G}}$	$\sqrt{2\lambda_2(\hat{L})}$

blends and in particular the *blend gap*. Section 6.2.2 controls the distance between an approximate eigenvector and an invariant subspace in terms of the eigenresidual and blend gap thereby showing that accurate approximation to a spectral blend is easier to compute than an accurate approximation of a single eigenvector. We provide a general tool for deriving residual tolerances based on the structure of the graph spectrum. In order to illustrate the utility of spectral blends, Section 6.3 studies a model problem and uses the theory of block cyclic matrices and locally supported eigenvectors to present a closed form for the eigenvalues and vectors. We show that any blend of large eigenvalue eigenvectors for the ring of cliques recovers a correct clustering. This indicates that for problems where there are multiple good partitions of the graph, spectral blends can be used to partition accurately. The eigendecomposition of the model problem provides error and residual tolerances for solving this problem with sweep cuts. Theorem 12 allows us to give guidance for error tolerances for spectral partitioning. One should solve the eigenproblem to a residual tolerance no greater than $\mathcal{O}\left(n^{-1}\right)$ for graphs of size n. In contrast, recovering the optimal spectral partition requires a residual of $\mathcal{O}\left(n^{-7/2}\right)$, which is infeasible for large datasets. Theorem 15 shows that for the ring of cliques where the number of clusters is polynomial in the sizes of the clusters, the number of power method steps taken to recover the clusters is $\mathcal{O}\left(1\right)$. Further research will be able to expand these results to more general graphs which have multiple good partitions.

CHAPTER VII

CONCLUSION

This thesis explores the application of numerical, statistical, and streaming data analysis techniques to graphs. Chapter 4 advances a novel framework for graph analysis based on streaming graph features combined with statistical analysis of those features. Vertex feature construction using `STINGER` leads to statistical analysis and machine learning applications for dynamic behavior analysis, outlier detection, and behavioral clustering. The behavioral cluster structure of dynamic complex networks can be found with a two phase process. The first phase extracts graph topology based features representing each vertex, which incorporate information about the changes in the graph, and the second phase applies well understood statistical and machine learning methods solve the analysis problem such as revealing the temporal cluster structure of the vertices. This framework allows one to apply the best streaming graph algorithms with the best statistical methods to understand complex data sets without reinventing statistical methods in the graph theoretic context.

Chapter 5 and Chapter 6 study the impact of approximation error when numerical methods are applied to graph analysis. Novel analysis of these numerical approaches gives rise to convergence criteria and deeper understanding of the interaction between numerical methods and graph analysis. By studying the relationship between numerical accuracy and data mining quality, a novel stopping criterion for spectral partitioning is derived and experimentally validated. This stopping criterion works better for ill conditioned graphs when spectral partitioning takes the longest. New theorems are proven to understand the convergence behavior of iterative methods for matrices with clustered spectra. The key parameter introduced in the analysis is the blend gap, which characterizes the separation of the invariant subspaces as reflected in a test vector. The impact of numerical error on sweep cuts can be understood by examining the sensitivity of the invariant subspace associate with the low energy Laplacian eigenvalues. Applying this technique to a model problem yields concrete bounds on the necessary tolerances for such graphs. These bounds can be used to predict the necessary and sufficient solver tolerances for general graphs. This analysis connects the continuous values of the Laplacian eigenvectors to the discrete sweep cut algorithm.

These techniques improve the understanding of dynamic graphs and our understanding of how to compute under the constraint of rapid changes to the input data. The link between numerical accuracy and data analysis accuracy is difficult to characterize in general and this dissertation contributes to a specific aspect of this link. For the spectral partitioning problem, we show that there is a large gap between the accuracy required to recover the Fiedler vector partitions and the accuracy required to recover the cluster structure in the data. This gap can be exploited to solve problems with fewer iterations and to ensure algorithms are robust to numerical errors.

REFERENCES

[1] Ahn, K. J., Guha, S., and McGregor, A., “Graph sketches: sparsification, spanners, and subgraphs,” in *Proceedings of the 31st symposium on Principles of Database Systems*, pp. 5–14, ACM, 2012.

[2] Akoglu, L., McGlohon, M., and Faloutsos, C., “Oddball: Spotting anomalies in weighted graphs,” in *Advances in Knowledge Discovery and Data Mining (KDD2010)*, pp. 410–421, Springer, 2010.

[3] Bader, D. A., Kintali, S., Madduri, K., and Mihail, M., “Approximating Betweenness Centrality,” in *Proc.\ 5th Workshop on Algorithms and Models for the Web-Graph (WAW2007)*, vol. 4863 of *Lecture Notes in Computer Science*, (San Diego, CA), pp. 134–137, Springer-Verlag, Dec. 2007.

[4] Bakshy, E., Hofman, J. M., Mason, W. A., and Watts, D. J., “Everyone’s an influencer: quantifying influence on Twitter,” in *Proceedings of the fourth ACM international conference on Web search and data mining (WSDM2011)*, pp. 65–74, ACM, 2011.

[5] Bishop, C. M., *Pattern Recognition and Machine Learning (Information Science and Statistics).* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[6] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E., “Fast unfolding of communities in large networks,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, Oct. 2008.

[7] Boutsidis, C., Gittens, A., and Kambandur, A., “Spectral clustering via the power method âĂŞ provably,” 2015.

[8] Buluç, A., Meyerhenke, H., Safro, I., Sanders, P., and Schulz, C., “Recent advances in graph partitioning,” *CoRR*, vol. abs/1311.3144, 2013.

[9] Cha, M., Haddadi, H., Benevenuto, F., and Gummadi, P. K., “Measuring user influence in Twitter: The million follower fallacy.,” *International AAAI Conference on Web and Social Media (ICWSM2010)*, vol. 10, no. 10-17, p. 30, 2010.

[10] Chung, F. and Simpson, O., “Computing heat kernel pagerank and a local clustering algorithm,” *arXiv preprint arXiv:1503.03155*, 2015.

[11] Chung, F. R., *Spectral graph theory*, vol. 92. American Mathematical Soc., 1997.

[12] Cichocki, A. and Phan, A.-H., "Fast local algorithms for large scale nonnegative matrix and tensor factorizations," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E92-A, pp. 708–721, 2009.

[13] Cichocki, A., Zdunek, R., and Amari, S., "Hierarchical als algorithms for nonnegative matrix and 3D tensor factorization," *Lecture Notes in Computer Science*, vol. 4666, pp. 169–176, 2007.

[14] Cucuringu, M. and Mahoney, M. W., "Localization on low-order eigenvectors of data matrices," *CoRR*, vol. abs/1109.1355, 2011.

[15] Dainotti, A., King, A., Claffy, K., Papale, F., and PescapÃĺ, A., "Analysis of a "/0" stealth scan from a botnet," in *Internet Measurement Conference (IMC2012)*, pp. 1–14, Nov 2012.

[16] Dainotti, A., Squarcella, C., Aben, E., Claffy, K., Chiesa, M., Russo, M., and PescapÃĺ, A., "Analysis of country-wide internet outages caused by censorship," *IEEE/ACM Transactions on Networking*, 2014.

[17] Dasgupta, A., Hopcroft, J. E., and McSherry, F., "Spectral analysis of random graphs with skewed degree distributions," in *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pp. 602–610, IEEE, 2004.

[18] Davis, C. and Kahan, W. M., "Some new bounds on perturbation of subspaces," *Bulletin of the American Mathematical Society*, vol. 75, no. 4, pp. 863–868, 1969.

[19] Del Corso, G. M., Gullí, A., and Romani, F., "Fast pagerank computation via a sparse linear system," *Internet Math.*, vol. 2, no. 3, pp. 251–273, 2005.

[20] Doyle, P. G. and Snell, J. L., *Random walks and electric networks*, vol. 22. Carus Mathematical Monographs, Mathematical Association of America, Washington, DC, 1984.

[21] Ediger, D., Appling, S., Briscoe, E., McColl, R., and Poovey, J., "Real-time streaming intelligence: Integrating graph and nlp analytics," in *Proc. High Performace Embedded Computing Workshop (HPEC2014)*, 2014.

[22] Ediger, D., Jiang, K., Riedy, J., and Bader, D. A., "Massive streaming data analytics: A case study with clustering coefficients," in *4th Workshop on Multithreaded Architectures and Applications (MTAAP)*, (Atlanta, Georgia), Apr. 2010.

[23] Ediger, D., Jiang, K., Riedy, J., Bader, D. A., Corley, C., Farber, R., and Reynolds, W. N., "Massive social network analysis: Mining twitter for social good," *Parallel Processing, International Conference on*, pp. 583–593, 2010.

[24] Ediger, D., Riedy, E. J., Bader, D. A., and Meyerhenke, H., "Tracking structure of streaming social networks," in *5th Workshop on Multithreaded Architectures and Applications (MTAAP)*, May 2011.

[25] Erdős, P. and Rényi, A., "On the strength of connectedness of a random graph," *Acta Mathematica Hungarica*, vol. 12, no. 1-2, pp. 261–267, 1961.

[26] Fairbanks, J. P., Kennan, R., Park, H., and A., B. D., "Behavioral clusters in dynamic graphs," *Parallel Computing*, 2015.

[27] Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., and Zhang, J., "On graph problems in a semi-streaming model," *Theoretical Computer Science*, vol. 348, no. 2-3, pp. 207–216, 2005.

[28] Fishkind, D., Sussman, D., Tang, M., Vogelstein, J., and Priebe, C., "Consistent adjacency-spectral partitioning for the stochastic block model when the model parameters are unknown," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 1, pp. 23–39, 2013.

[29] Flajolet, P. and Martin, G. N., "Probabilistic counting," in *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*, SFCS '83, (Washington, DC, USA), pp. 76–82, IEEE Computer Society, 1983.

[30] Fortunato, S. and Barthélemy, M., "Resolution limit in community detection," *Proceedings of the National Academy of Sciences*, vol. 104, no. 1, pp. 36–41, 2007.

[31] Freeman, L., "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35–41, 1977.

[32] Geisberger, R., Sanderst, P., and Schultest, D., "Better approximation of betweenness centrality," in *Proceedings of the Tenth Workshop on Algorithm Engineering and Experiments and the Fifth Workshop on Analytic Algorithmics and Combinatorics*, vol. 129, p. 90, Society for Industrial & Applied, 2008.

[33] Glasgow, K. and Fink, C., "Hashtag lifespan and social networks during the London riots," in *Social Computing, Behavioral-Cultural Modeling and Prediction* (Greenberg, A. M., Kennedy, W. G., and Bos, N. D., eds.), vol. 7812 of *Lecture Notes in Computer Science*, pp. 311–320, Springer Berlin Heidelberg, 2013.

[34] Golub, G. H. and Van Loan, C. F., *Matrix Computations.* The Johns Hopkins University Press, 3rd ed., 1996.

[35] Golub, G. H. and Van Loan, C. F., *Matrix Computations (4th Ed.).* Baltimore, MD, USA: Johns Hopkins University Press, 2013.

[36] Gonzales, E. F. and Zhang, Y., "Accelerating the Lee-Seung algorithm for non-negative matrix factorization," *Dept. Comput. & Appl. Math., Rice Univ., Houston, TX, Tech. Rep. TR-05-02*, 2005.

[37] Guattery, S. and Miller, G. L., "On the quality of spectral separators," *SIAM Journal on Matrix Analysis and Applications*, vol. 19, no. 3, pp. 701–719, 1998.

[38] Guimerkà, R. and Amaral, L., "Functional cartography of complex metabolic networks," *Nature*, vol. 433, no. 7028, pp. 895–900, 2005. cited By 1225.

[39] Henderson, K., Eliassi-Rad, T., Faloutsos, C., Akoglu, L., Li, L., Maruhashi, K., Prakash, B. A., and Tong, H., "Metric forensics: a multi-level approach for mining volatile graphs," in *Knowledge Discovery and Data Mining (KDD2010)*, pp. 163–172, 2010.

[40] Henderson, K., Gallagher, B., Eliassi-Rad, T., Tong, H., Basu, S., Akoglu, L., Koutra, D., Faloutsos, C., and Li, L., "Rolx: structural role extraction & mining in large graphs," in *Proceedings of the international conference on Knowledge discovery and data mining (KDD2012)*, pp. 1231–1239, ACM, 2012.

[41] Henson, V. E. and Sanders, G., "Locally supported eigenvectors of matrices associated with connected and unweighted power-law graphs," vol. 39, pp. 353–378, 2012.

[42] Ho, N.-D., Dooren, P. V., and Blondel, V. D., "Descent methods for nonnegative matrix factorization," *CoRR*, vol. abs/0801.3199, 2008.

[43] Holland, P. W., Laskey, K. B., and Leinhardt, S., "Stochastic blockmodels: first steps," *Social Networks*, vol. 5, no. 2, pp. 109–137, 1983.

[44] Huang, L., Yan, D., Taft, N., and Jordan, M. I., "Spectral clustering with perturbed data," in *Advances in Neural Information Processing Systems 21* (Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., eds.), pp. 705–712, Curran Associates, Inc., 2008.

[45] Jerrum, M., *Counting, sampling and integrating: algorithms and complexity.* Lectures in Mathematics, Basel: Birkhäuser Verlag, 2003.

[46] Kang, U., Meeder, B., Papalexakis, E. E., and Faloutsos, C., "Heigen: Spectral analysis for billion-scale graphs," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 26, no. 2, pp. 350–362, 2014.

[47] Kannan, R., Ishteva, M., and Park, H., "Bounded matrix low rank approximation," in *IEEE 12th International Conference on Data Mining (ICDM2012)*, pp. 319–328, IEEE, 2012.

[48] Kannan, R., Vempala, S., and Vetta, A., "On clusterings: Good, bad and spectral," *Journal of the ACM (JACM)*, vol. 51, no. 3, pp. 497–515, 2004.

[49] Karypis, G. and Kumar, V., "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, pp. 359–392, Dec. 1998.

[50] Kato, T., "On the upper and lower bounds of eigenvalues," *Journal of the Physical Society of Japan*, vol. 4, no. 4-6, pp. 334–339, 1949.

[51] Kim, J., He, Y., and Park, H., "Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework," *Journal of Global Optimization*, vol. 58, no. 2, pp. 285–319, 2014.

[52] Kim, J. and Park, H., "Fast nonnegative matrix factorization: An active-set-like method and comparisons," *SIAM J. Scientific Computing*, vol. 33, no. 6, pp. 3261–3281, 2011.

[53] Kloster, K. and Gleich, D. F., "Personalized PageRank Solution Paths," *ArXiv e-prints*, Mar. 2015.

[54] Kokiopoulou, E., Chen, J., and Saad, Y., "Trace optimization and eigenproblems in dimension reduction methods," *Numerical Linear Algebra with Applications*, vol. 18, no. 3, pp. 565–602, 2011.

[55] Kuang, D., Yun, S., and Park, H., "Symnmf: nonnegative low-rank approximation of a similarity matrix for graph clustering," *Journal of Global Optimization*, pp. 1–30, 2014.

[56] Kwak, H., Lee, C., Park, H., and Moon, S., "What is Twitter, a social network or a news media?," in *19th World-Wide Web (WWW2010) Conference*, (Raleigh, North Carolina), pp. 591–600, Apr. 2010.

[57] Lee, D. D. and Seung, H. S., "Learning the parts of objects by non-negative matrix factorization.," *Nature*, vol. 401, pp. 788–791, Oct. 1999.

[58] Lehoucq, R. and Sorensen, D., "Deflation techniques for an implicitly restarted arnoldi iteration," *SIAM Journal on Matrix Analysis and Applications*, vol. 17, no. 4, pp. 789–821, 1996.

[59] Leskovec, J., Kleinberg, J., and Faloutsos, C., "Graph evolution: Densification and shrinking diameters," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, p. 2, 2007.

[60] Leskovec, J. and Krevl, A., "SNAP Datasets: Stanford large network dataset collection." `http://snap.stanford.edu/data`, June 2014.

[61] Lin, C. J., "Projected gradient methods for nonnegative matrix factorization," *Neural Comput.*, vol. 19, pp. 2756–2779, Oct. 2007.

[62] Lusseau, D., "The emergent properties of a dolphin social network," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 270, no. Suppl 2, pp. S186–S188, 2003.

[63] Lusseau, D. and Newman, M. E. J., "Identifying the role that animals play in their social networks," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 271, no. Suppl 6, pp. S477–S481, 2004.

[64] Lyzinski, V., Sussman, D., Tang, M., Athreya, A., and Priebe, C. submitted 2013.

[65] Lyzinski, V., Sussman, D. L., Tang, M., Athreya, A., and Priebe, C. E., "Perfect clustering for stochastic blockmodel graphs via adjacency spectral embedding," *Electron. J. Statist.*, vol. 8, no. 2, pp. 2905–2922, 2014.

[66] Mendoza, M., Poblete, B., and Castillo, C., "Twitter under crisis: can we trust what we RT?," in *Proceedings of the First Workshop on Social Media Analytics (SOMA2010)*, (New York, NY, USA), pp. 71–79, ACM, 2010.

[67] Mihail, M., "Conductance and convergence of markov chains-a combinatorial treatment of expanders," in *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pp. 526–531, IEEE Comput. Soc. Press, 1989.

[68] Mihail, M. and Papadimitriou, C., *Randomization and Approximation Techniques in Computer Science: 6th International Workshop, RANDOM 2002 Cambridge, MA, USA, September 13–15, 2002 Proceedings*, ch. On the Eigenvalue Power Law, pp. 254–262. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002.

[69] Möhring, R. H., Schilling, H., Schütz, B., Wagner, D., and Willhalm, T., *Experimental and Efficient Algorithms: 4th International Workshop, WEA 2005, Santorini Island, Greece, May 10-13, 2005. Proceedings*, ch. Partitioning Graphs to Speed Up Dijkstra's Algorithm, pp. 189–202. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[70] Newman, M. E. J. and Girvan, M., "Finding and evaluating community structure in networks," *Phys. Rev. E*, vol. 69, p. 026113, Feb 2004.

[71] Newman, M., "Newman Datasets." `http://www-personal.umich.edu/~mejn/netdata/`, apr 2013.

[72] Paatero, P. and Tapper, U., "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values," *Environmetrics*, vol. 5, no. 2, pp. 111–126, 1994.

[73] Parlett, B., *The Symmetric Eigenvalue Problem.* Society for Industrial and Applied Mathematics, 1998.

[74] Persi Diaconis, D. S., "Geometric bounds for eigenvalues of markov chains," *The Annals of Applied Probability*, vol. 1, no. 1, pp. 36–61, 1991.

[75] Pothen, A., Simon, H. D., and Liou, K.-P., "Partitioning sparse matrices with eigenvectors of graphs," *SIAM Journal on Matrix Analysis and Applications*, vol. 11, no. 3, pp. 430–452, 1990.

[76] Psorakis, I., Roberts, S., Ebden, M., and Sheldon, B., "Overlapping community detection using Bayesian non-negative matrix factorization," *Physical Review E*, vol. 83, p. 066114, June 2011.

[77] Rossi, R. A., Gallagher, B., Neville, J., and Henderson, K., "Modeling dynamic behavior in large evolving graphs," in *WSDM*, pp. 667–676, 2013.

[78] Rousseeuw, P. J. and Driessen, K. V., "A fast algorithm for the minimum covariance determinant estimator," *Technometrics*, vol. 41, no. 3, pp. 212–223, 1999.

[79] Saad, Y., *Numerical Methods for Large Eigenvalue Problems: Revised Edition*, vol. 66. Siam, 2011.

[80] Sakaki, T., Okazaki, M., and Matsuo, Y., "Earthquake shakes Twitter users: real-time event detection by social sensors," in *Proceedings of the 19th international conference on World wide web (WWW2010)*, pp. 851–860, ACM, 2010.

[81] Schölkopf, B., Platt, J. C., Shawe-Taylor, J. C., Smola, A. J., and Williamson, R. C., "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, pp. 1443–1471, July 2001.

[82] Signorini, A., Segre, A. M., and Polgreen, P. M., "The use of Twitter to track levels of disease activity and public concern in the u.s. during the influenza a h1n1 pandemic," *PLoS ONE*, vol. 6, p. e19467, 05 2011.

[83] Sorensen, D. C., "Implicitly restarted arnoldi/lanczos methods for large scale eigenvalue calculations," (Hampton, VA), 1995.

[84] Spielmat, D. A. and Teng, S.-H., "Spectral partitioning works: Planar graphs and finite element meshes," in *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pp. 96–105, IEEE, 1996.

[85] Tee, G. J., "Eigenvectors of block circulant and alternating circulant matrices," 2005.

[86] The CAIDA UCSD, "passive traces - 2014," 2014.

[87] Tong, H. and Lin, C.-Y., "Non-negative residual matrix factorization: problem definition, fast solutions, and applications," *Statistical Analysis and Data Mining (SDM2012)*, vol. 5, pp. 3–15, Feb. 2012.

[88] Tong, H., Papadimitriou, S., Sun, J., Yu, P. S., and Faloutsos, C., "Colibri: Fast mining of large static and dynamic graphs," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, (New York, NY, USA), pp. 686–694, ACM, 2008.

[89] Trevisan, L., *Lecture Notes on Expansion, Sparsest Cut, and Spectral Graph Theory.* 2013.

[90] Trevisan, L., "Lecture notes on expansion, sparsest cut, and spectral graph theory," 2014.

[91] Twitter, "#Goal," April 2012. `http://blog.uk.twitter.com/2012/04/goal.html`.

[92] Twitter, "Celebrating #Twitter7," March 2013. `http://blog.twitter.com/2013/03/celebrating-twitter7.html`.

[93] Von Luxburg, U., "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[94] Wang, F., Li, T., Wang, X., Zhu, S., and Ding, C., "Community discovery using nonnegative matrix factorization," *Data Mining and Knowledge Discovery*, vol. 22, pp. 493–521, July 2010.

[95] Wang, H., Tang, M., Park, Y., and Priebe, C. E., "Locality statistics for anomaly detection in time series of graphs," *ArXiv e-print 1306.0267*, submitted 2013.

[96] Wasserman, S. and Faust, K., *Social Network Analysis: Methods and Applications.* Cambridge University Press, 1994.

[97] Wilkinson, J. H., Wilkinson, J. H., and Wilkinson, J. H., *The algebraic eigenvalue problem*, vol. 87. Clarendon Press Oxford, 1965.

[98] Yan, D., Huang, L., and Jordan, M. I., "Fast approximate spectral clustering," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, (New York, NY, USA), pp. 907–916, ACM, 2009.

[99] Yang, J. and Leskovec, J., "Overlapping community detection at scale : A nonnegative matrix factorization approach," in *Proceedings of the international conference on Web search and data mining (WSDM2013)*, pp. 587–596, 2013.