UNIVERSITY OF CALIFORNIA, SAN DIEGO

Diffusion and Clustering on Large Graphs

A dissertation submitted in partial satisfaction of the requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Alexander Tsiatas

Committee in charge:

Professor Fan Chung Graham, Chair
Professor Kamalika Chaudhuri
Professor Tara Javidi
Professor Todd Kemp
Professor Lawrence K. Saul

2012
The dissertation of Alexander Tsiatas is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

__________________________________________

__________________________________________

__________________________________________

__________________________________________

__________________________________________

Chair

University of California, San Diego

2012
DEDICATION

To a complex and inspiring world.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature Page</td>
<td>iii</td>
</tr>
<tr>
<td>Dedication</td>
<td>iv</td>
</tr>
<tr>
<td>Epigraph</td>
<td>v</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>vi</td>
</tr>
<tr>
<td>List of Figures</td>
<td>ix</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xi</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>xii</td>
</tr>
<tr>
<td>Vita</td>
<td>xiv</td>
</tr>
<tr>
<td>Abstract of the Dissertation</td>
<td>xv</td>
</tr>
<tr>
<td>Chapter 1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 History</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Results in this dissertation</td>
<td>5</td>
</tr>
<tr>
<td>Chapter 2 Spectral Graph Theory, Diffusion, and Graph Cuts</td>
<td>9</td>
</tr>
<tr>
<td>2.1 Random walks and personalized PageRank</td>
<td>10</td>
</tr>
<tr>
<td>2.2 The normalized graph Laplacian and its spectrum</td>
<td>15</td>
</tr>
<tr>
<td>2.3 Graph cuts and the Cheeger ratio</td>
<td>16</td>
</tr>
<tr>
<td>Chapter 3 Global Graph Clustering Using PageRank Optimization</td>
<td>20</td>
</tr>
<tr>
<td>3.1 A distance metric and PageRank variance measures for graph clustering</td>
<td>21</td>
</tr>
<tr>
<td>3.2 The PageRank-Clustering algorithms</td>
<td>24</td>
</tr>
<tr>
<td>3.3 Some PageRank diffusion lemmas</td>
<td>28</td>
</tr>
<tr>
<td>3.4 Analysis of PageRank-Clustering</td>
<td>34</td>
</tr>
<tr>
<td>3.5 Acknowledgement</td>
<td>38</td>
</tr>
<tr>
<td>Chapter 4 Drawing Clustered Graphs Using PageRank</td>
<td>39</td>
</tr>
<tr>
<td>4.1 The graph drawing algorithm PageRank-Display</td>
<td>40</td>
</tr>
<tr>
<td>4.2 Graph drawing illustrations</td>
<td>42</td>
</tr>
<tr>
<td>4.3 Acknowledgement</td>
<td>43</td>
</tr>
</tbody>
</table>
Chapter 5 Spectral Analysis of Communication Networks with Dirichlet Eigenvalues ........................................... 50
5.1 Convergence of the Dirichlet spectral gap in finite trees .... 52
5.2 Comparison of spectral gaps in Rocketfuel networks and the 2-D grid ......................................................... 56
5.3 Spectral clustering of Rocketfuel networks using Dirichlet eigenvectors ...................................................... 58
5.4 Acknowledgement ................................................................. 61

Chapter 6 Spectral Clustering of Graphs with General Degrees in the Extended Planted Partition Model .......... 63
6.1 Planted partition models and graph Laplacians ............ 67
6.2 The EPPCluster algorithm for recovering the hidden partition ................................................................. 69
6.3 Analysis of EPPCluster ............................................................ 72
6.4 A statistical lower bound for the EPP problem ........... 88
6.5 Acknowledgement ................................................................. 90

Chapter 7 Dirichlet PageRank and Ranking Algorithms Based on Trust and Distrust ............................................ 91
7.1 PageRank with Dirichlet boundary conditions ............ 96
7.2 Dirichlet PageRank with given boundary conditions ... 102
7.3 The ApproxDirichPR algorithm and analysis .......... 103
7.4 Applications of Dirichlet PageRank ......................... 107
7.5 Acknowledgement ................................................................. 114

Chapter 8 Distributing Antidote to Combat Network Epidemics Using PageRank ......................................................... 115
8.1 Infection model and inoculation scheme ................ 117
8.2 Probabilistic analysis of infection paths .................. 121
8.3 Proofs of the main theorems ........................................... 127
8.4 Some questions for further study ......................... 130
8.5 Acknowledgement ................................................................. 131

Chapter 9 Hypergraph Coloring Games and Voter Models .... 132
9.1 The interaction model on hypergraphs ................ 136
9.2 The voting game on a hypergraph as a random walk on the associated state graph .............................................. 139
9.3 Memoryless interactions and semigroup spectral graph theory ................................................................. 141
9.4 The cut-off time for voter interaction games .......... 143
9.5 Estimating the probability of a given event ........... 145
9.6 The interaction model with partially memoryless interactions ........................................ 147
9.7 The general interaction model ........................................ 149
9.8 Acknowledgement ........................................ 151

Bibliography ........................................ 154
LIST OF FIGURES

Figure 2.1: The random-walk stationary distribution for a network [99], proportional to the degree distribution .......................... 12
Figure 2.2: PageRank vectors for a network [99], showing the effects of changing $\alpha$ and $s$ ................................................. 14
Figure 2.3: Two different partitions of the same network [99] and their Cheeger ratios ......................................................... 17
Figure 2.4: A partition of a network [99], showing the core of one side . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
Figure 2.5: Comparison of PageRank vectors when the starting distribution starts from inside or outside the core of the partition given in Figure 2.4 .............................................................. 19
Figure 3.1: Graph variance metrics for the dumbbell graph $U$ ......................... 29
Figure 4.1: Results of PageRank-Display ($\alpha = 0.03$) on the dolphin social network [99] .......................................................... 43
Figure 4.2: Results of PageRank-Display ($\alpha = 0.3$) on the dolphin social network [99] .......................................................... 44
Figure 4.3: Results of PageRank-Display ($\alpha = 0.1$) on the football game network [66] .......................................................... 45
Figure 4.4: Results of PageRank-Display ($\alpha = 0.3$) on the football game network [66] .......................................................... 46
Figure 4.5: Results of PageRank-Display on Zachary’s karate network [135] 47
Figure 4.6: Results of PageRank-Display on a network of political books about the 2004 US presidential election [92]................. 48
Figure 4.7: Results of PageRank-Display on a network of US Air Force flying teams [113] ......................................................... 49
Figure 5.1: Dirichlet spectral gap for successively larger 3-regular trees, showing convergence to a nonzero value ....................... 54
Figure 5.2: Comparison of traditional and Dirichlet spectral gaps in Rock-eftuel data as well as the 2-dimensional Euclidean grid ........ 58
Figure 5.3: Comparison of Cheeger ratio $h$ and number of components $c$ for cuts for various datasets using Dirichlet ($D$) and traditional ($T$) spectral clustering ...................................................... 62
Figure 6.1: Comparison of spectral partitioning of a graph generated from an EPP model, with three different matrices ................. 72
Figure 7.1: Comparison of three vertex rankings on small graph ............... 99
Figure 7.2: Three vertex rankings on a network [99] with two known spammers100
Figure 7.3: Comparison of vertex rankings on a network [70, 117] with negative edges ......................................................... 109
Figure 9.1: Simulations of the consensus game and its partially memoryless approximation. 152
LIST OF TABLES

Table 5.1: Structural and spectral properties of Rocketfuel datasets . . . . 57
Table 5.2: Aggregate data comparing Dirichlet spectral clustering with traditional spectral clustering for several Rocketfuel datasets . . . . 60
ACKNOWLEDGEMENTS

This dissertation would not be possible without the wisdom and mentorship of my advisor, Fan Chung. Her mathematical curiosity, persistence, and wealth of experience were invaluable to my experiences. Thanks also to Kamalika Chaudhuri for her graduate-school, career, and talk-preparation advice. I am positive that I would not have learned nearly as much without the opportunity to work with either of these inspiring researchers and mentors.

My family was always available to lend their support from thousands of miles away. Their encouragement was essential to the completion of my graduate studies; for that, many thanks are deserved. Thanks also to Christina, whose support and encouragement guided me throughout my studies.

Finally, thanks to friends both old and new for helping me adjust to the cultural changes in moving to San Diego. I was lucky to have many people I could count on to talk me though unpredictable scenarios, offer support and encouragement, and even to constantly listen to my incessant ramblings.

Chapters 3 and 4 contain material that appeared in the following two published articles:


Chapter 5 contains material that appears in the following article, currently being prepared for submission:

Chapter 6 contains material that appears in the following article, accepted for publication:


Chapter 7 contains material from two articles; one has been published, and the other has been accepted for publication:


Chapter 8 contains material from the following published article:


Chapter 9 contains material from the following published article:

### VITA

<table>
<thead>
<tr>
<th>Year</th>
<th>Degree</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008</td>
<td>B. S. in Computer Science <em>magna cum laude</em></td>
<td>Cornell University</td>
</tr>
<tr>
<td>2010</td>
<td>M. S. in Computer Science</td>
<td>University of California, San Diego</td>
</tr>
<tr>
<td>2012</td>
<td>Ph. D. in Computer Science</td>
<td>University of California, San Diego</td>
</tr>
</tbody>
</table>
ABSTRACT OF THE DISSERTATION

Diffusion and Clustering on Large Graphs

by

Alexander Tsiatas

Doctor of Philosophy in Computer Science

University of California, San Diego, 2012

Professor Fan Chung Graham, Chair

This dissertation studies two important algorithmic problems on networks: graph diffusion and clustering. These problems are closely related: bottlenecks that diffusive processes find difficult to cross make good cluster boundaries and vice versa.

First, we give an efficient global partitioning algorithm specifically tailored for a graph-theoretic setting. It uses sampling with PageRank vectors as probability distributions, optimizing new PageRank-distance metrics that depend only on the jumping constant $\alpha$. Once $k$ centers are found, we give a graph drawing algorithm that uses a force-based layout, with spring constants determined by PageRank vectors, that highlights clustered structure.
We study graph clustering in two additional contexts. The first is with subsets of communication networks that have imposed boundaries when the topologies reach private subnetworks. We empirically show that spectral clustering using Dirichlet eigenvectors instead of the usual eigenvectors can be more suitable. Second, we study the extended planted partition model, a random-graph model that starts with a predetermined partitioning of the vertices with arbitrary expected degrees. We give a spectral clustering algorithm to recover the planted partition under certain conditions. Our algorithm uses a new matrix, the degree-corrected random-walk Laplacian, and unlike some prior work, does not assume knowledge of any graph generation parameters.

Next, we propose and study Dirichlet PageRank, or PageRank vectors with arbitrary boundary conditions, to address vertex ranking problems resulting from the propagation of trust and distrust in networks. Using Dirichlet PageRank, we can compute vertex rankings in the presence of known spammers and negatively-weighted edges.

We also study a diffusive network epidemic model: the contact process. We show that network epidemics can be stopped by finding a local cluster around the infection’s starting points and inoculating the vertices in that cluster with antidote proportional to a personalized PageRank vector. This is more efficient than prior work which required distributing antidote widely to the entire graph. Finally, we propose a new voter model for the evolution of electoral opinions in social networks. Unlike the traditional voter model, our model operates on hypergraphs, allowing for more complex interactions, and it yields a wider range of outcomes.
Chapter 1

Introduction

In recent years, the mathematical and algorithmic study of large graphs has risen in importance. Computers have advanced to the point to make massive data storage possible, and many important datasets consist of entities as well as their relationships. This naturally leads to a graph representation, and it is essential to develop tractable computational tools to analyze and effectively use the data held in large networks.

There are two important problems in particular that have been extensively studied: graph diffusion and graph clustering. Graph diffusion refers to problems involving spreading or propagation along the edges of a graph. There are many real-world situations that can be characterized this way: Web users navigating from site to site on the Internet by clicking hyperlinks in a browser, new products becoming popular based on word-of-mouth and interactions between people, and disease spreading among human populations based on direct contact. Computer scientists and mathematicians have developed tools and methods for analyzing these and other related problems. Using techniques such as random walks and PageRank, many precise statements can be made about spreading processes on networks.

Graph clustering, or graph partitioning, is another important and well-studied problem on graphs. With large datasets, it is often desirable to condense the data into more manageable segments. The clustering problem is to organize the vertices in a graph into groups, where each group is made up of vertices that
are close to each other but comparatively separate from the rest of the graph. The applications of graph clustering are far-reaching, including such problems such as image segmentation, exploratory data analysis, and physical network analysis to find bottlenecks.

It turns out that diffusion and clustering are inherently related: if there is a set of vertices in a graph where diffusive processes have difficulty escaping, that set is often a good cluster. This relationship has been made rigorous by statements such as the Cheeger inequality. This dissertation will build upon the large body of recent research in graph clustering and diffusion. We will propose and study new algorithms for graph partitioning, develop new techniques for studying diffusion with the added context of a graph boundary, study techniques for combating network epidemics, and introduce a new model for studying the evolution of voting opinions in a social-network context. To accomplish such tasks, we will build upon foundations such as mixing results for personalized PageRank vectors [8], and spectral graph theory [35].

1.1 History

The study of large networks is a fairly recent development, and there have been many phenomena shown to be ubiquitous in real-world datasets. One important property is the small-world phenomenon [132], the existence of short paths between any randomly-chosen pair of vertices. Many real-world networks exhibit small-world behavior, and they are characterized by having a low average shortest path length and many dense subgraphs. This results in faster mixing of diffusive processes.

Another hallmark of large graphs is the power-law degree distribution. In the late 1990s, several groups independently discovered that virtually every large network found in nature exhibits a power-law degree distribution [2, 3, 23, 59]. This is in direct contrast with many natural mathematical structures and phenomena that are built on the normal distribution, such as the traditional Erdős-Rényi model of random graphs [58]. Thus, naturally-occurring graphs have more highly-
connected vertices than one would expect from a normal distribution or a graph
generated by an Erdős-Rényi model, and can result in a more clustered structure.

One more recently studied phenomenon observed in real-world graphs is
the notion of *global network curvature* [110]. Negatively curved graphs have been
empirically shown to result in congestion and bottlenecks at the core of the network,
further motivating the continued study of graph diffusion and clustering.

### 1.1.1 Random walks and PageRank

Random walks have been studied for over a century ([4] is a more recent
reference) and form the basis of studying diffusion on graphs. Besides their well-
known behavior and stationary distributions, Lovász and Simonovits showed that
random walks can be used to efficiently find sparse cuts in graphs [97, 98]. Fan
Chung [34] tightened their results into a local Cheeger inequality, and they have
also been used in algorithms to find the volume of convex bodies [97] as well as
local partitions in graphs [123].

PageRank was originally proposed by Brin and Page [22] as a ranking sys-
tem for Web pages, and is part of the reason for the success of Google as a search
engine. It was built by modeling a “random surfer” on the Web; the surfer follows
Web links, and the connection to diffusion is apparent: following graph edges.
Since then, PageRank has been studied as a mathematical tool, and in the litera-
ture “PageRank” refers to a vector representing the stationary probability of the
random surfer being at each vertex. While traditional PageRank probabilistically
returns to a random Web page selected uniformly at random, Personalized Page-
Rank [72, 75] generalizes this random restart to customized starting distributions,
often just a single vertex. Personalized PageRank has been used both for variants
of vertex ranking as well as local partitioning [8].

The exact computation of PageRank vectors requires solving a linear sys-
tem, and this can be computationally intractable for large networks. There are
iterative methods based on matrix multiplication, but for graphs as vast as the
Web, this is still unfeasible. Instead, algorithms using PageRank often use fast
approximation algorithms for PageRank, first developed by Andersen, Chung, and
Lang [8] and tightened by Chung and Zhao [46]. These algorithms require only local computations around the neighborhood of the PageRank seed vector.

1.1.2 Graph clustering

Graph clustering is a well-studied problem with applications in many areas. The basic problem is to divide a graph into logical subsets, where each subset is well-connected internally and well-separated from the rest of the graph. Such a problem comes up in many contexts, including exploratory data analysis, image segmentation, community identification, and bioinformatics. There are many graph clustering algorithms in use today; a more complete history appears in a survey by Schaeffer [120], and a tutorial on spectral clustering by von Luxburg [100]. Here, we focus on two variants: global and local graph clustering.

Global clustering is the problem of partitioning an entire graph into logical subsets. Note that while this dissertation focuses on clustering in a graph-theoretic setting, data represented by points in Euclidean space are often clustered using a similarity graph, making global graph clustering widely applicable. The earliest clustering algorithms are based on the $k$-means algorithm [96, 101], which has proven to be effective in many contexts, but it is NP-complete to optimize, and heuristic algorithms often become trapped in local minima.

More recent global clustering algorithms are based on spectral methods, using the eigenvectors or singular vectors of graph-theoretic matrices such as the adjacency matrix or graph Laplacian. Some early work on spectral clustering includes algorithms by Shi and Malik [122] and Ng et al. [112], and spectral clustering has been used in a machine-learning context by McSherry [104] and Dasgupta et al. [51].

Local graph partitioning is the problem of finding a dense community, well-separated from the rest of the graph, surrounding a given vertex $v$. Since we are only interested in finding one small community, it is desirable for local graph partitioning algorithms to operate using only local computations, rather than requiring matrix computations over the entire graph. Spielman and Teng [123, 124] gave the first local partitioning algorithm, Nibble, based on the random-walk mixing
results of Lovász and Simonovits [97, 98]. Andersen, Chung, and Lang [8] followed up with PageRank-Nibble, a more efficient local partitioning algorithm based on similar mixing results for personalized PageRank, with later refinements [7] and applications to directed graphs [9]. These algorithms continue to be refined, including a more recent result due to Andersen and Peres [10].

It is important to note that local graph partitions can be combined to find a global graph partitioning. This has led to efficient global graph partitioning algorithms by Spielman and Teng [124] and Andersen, Chung, and Lang [8].

1.2 Results in this dissertation

Graph clustering. In Chapter 3, we give a global graph partitioning algorithm PageRank-ClusteringA, along with an approximation version PageRank-ClusteringB. Knowing that PageRank is well-suited for capturing relationships between vertices in a graph, we propose a PageRank distance metric that is useful for clustering graphs. Because graphs with the small-world property have small diameters and short paths, traditional metrics such as shortest-path distance are not very descriptive, and the PageRank distance is much more useful.

Armed with the PageRank distance, we would like to optimize an overall graph metric similar to $k$-means with Euclidean distance. It is true that optimizing such quantities is NP-hard; instead, we show that if we sample vertices according to PageRank distributions, the problem is reduced to optimizing quantities that depend only on the PageRank jumping constant $\alpha$. We use probabilistic algorithms to show the correctness of PageRank-ClusteringA, provided that the input graph $G$ has a clustered structure to begin with. The approximation algorithm PageRank-ClusteringB is a more efficient, effectively using approximate PageRank algorithms from [8, 46] and matrix sampling methods from [119]. The result is an algorithm that automatically selects $k$ centers of mass and uses them to derive $k$ clusters. Hence, we have a graph clustering algorithm that uses a graph diffusion model, personalized PageRank, as a key tool.

Once we have our clustering algorithm, we can also use PageRank to draw
graphs, highlighting clustered structure. In Chapter 4, we present an algorithm **PageRank-Display** that uses a force-based layout algorithm, with spring constants determined by PageRank vectors, to draw clustered graphs in a way that shows the clustered structure. We give several examples of real-world network drawings.

In Chapter 5, we explore several real-world datasets representing subsets of IP-layer communication networks. These networks are smaller subsets of much larger networks, and thus they have an artificially induced boundary. This can happen often, since graph algorithms are often tractable only for small subsets of vast networks. Leskovec et al. [95] show that traditional spectral clustering algorithms often cut off tendrils of a graph near the boundary rather than at the network core. We show empirically that using Dirichlet eigenvectors in spectral clustering often yields bottlenecks that are closer to the core of the network.

Chapter 6 takes graph clustering to a machine-learning setting: we study the so-called *Extended Planted Partition* model, a random-graph model that incorporates a hidden partition \( V \) and varying edge probabilities resulting in arbitrary expected degree distributions. We introduce a new matrix, the *degree-corrected random-walk Laplacian*, and we use it to effectively recover \( V \) from an instantiation of a random graph, under certain conditions, without knowledge of the parameters used to generate it. We show that the degree-corrected random-walk Laplacian has eigenspace concentration results that are sharper than the adjacency matrix or the usual normalized Laplacian, and unlike the work of Dasgupta et al. [51], we do not require partial knowledge of the specific parameters used to generate the graph. We also show a statistical lower bound on the parameters of the model that is necessary for effective partition recovery by any method.

**Graph diffusion and vertex ranking with boundary conditions.**

In Chapter 7, we study a variant of the usual personalized PageRank: *Dirichlet PageRank vectors*. Dirichlet PageRank allows us to study graph diffusion and vertex ranking with the context of a graph boundary, allowing us to adjust the vector for a wide range of problems. We give an algorithm **ApproxDirichPR** for efficiently computing approximate Dirichlet PageRank vectors with arbitrary
boundary conditions, and we outline several vertex ranking problems that can be addressed by using them, including computing a vertex ranking with negatively-weighted edges, known spammer vertices, and a more personal network of trusted agents. Many of these applications address problems related to the propagation of trust and distrust in networks.

As stated earlier, we also use Dirichlet boundary conditions to cluster communication networks in Chapter 5. Dirichlet PageRank will also play a role in the study of network epidemics in Chapter 8.

**Other graph diffusion problems.** In Chapter 8, we study a network epidemic model and give a method for efficiently distributing antidote to combat epidemics. We study epidemics in the context of the well-studied susceptible-infected-susceptible contact process, a standard diffusive process used to model infections. Borgs et al. [19] give an scheme that requires giving antidote to the entire graph to eliminate the epidemic in $O(\log n)$ time, and we show that if we can find a local cluster $S$ around the infection’s starting points, we only need to distribute antidote to that cluster $S$ to achieve the same goal, as long as the amount of antidote given to each vertex $v \in S$ is proportional to a personalized PageRank vector. We show a probabilistic tradeoff between the Cheeger constant of the graph, the amount of antidote required, and the infection rate $\beta$ of the given model.

In Chapter 9, we study a voter model, a mathematical model of the flow of electoral opinions in a social network. Unlike the traditional voter model given in [47, 74], this interaction model does not generally result in unanimous consensus among the voters, better reflecting real-world scenarios. The interaction model also involves a hypergraph, allowing for interactions more complex than pairwise. We study many aspects of this model, including its spectrum and stationary distribution. To do this, we study a random walk on the state graph of the model, which is exponentially large, and therefore in the most general case, it is difficult to determine or even approximate components of the stationary distribution. However, for a special case where the interactions are memoryless, we can use clean results from semigroup spectral graph theory to determine the spectrum of the
state-graph random walk. We use these results to reason about estimating the probability of a given event occurring, and we use this memoryless version as a springboard to study a \textit{partially memoryless} process and finally to draw empirical inferences about the most general, state-dependent model.
Chapter 2

Spectral Graph Theory, Diffusion, and Graph Cuts

This dissertation concerns the study of large networks, and thus, basic terminology from graph theory is used throughout. Unless otherwise noted, we consider general undirected, unweighted graphs $G = (V, E)$ with vertex set $V$ and edge set $E$. By convention, $|V| = n$ and $|E| = m$.

For a vertex $v \in V$, we let $d_v$ denote the degree of $v$ which is the number of neighbors of $v$. Furthermore, if $u$ and $v$ are neighbors, we say that $u$ and $v$ are adjacent, and this relationship will be written $u \sim v$.

The degree sequence or degree distribution $d = (d_1, d_2, \ldots, d_n)$ lists the degrees of the vertices in $V$. Note that this sequence can be viewed as a $1 \times n$ vector, and we will treat all vectors as row vectors unless otherwise specified. For vectors such as $d$, it is assumed that the vector components correspond to the vertices in $V$, and we refer to the $v$th component, for $v \in V$, accordingly. Let $D = \text{diag}(d)$ denote the $n \times n$ diagonal degree matrix.

Besides $d$, there are several other important vectors that will be used throughout this dissertation. For a vertex $v \in V$, $1_v$ denotes the vector whose $v$th component is 1, and the rest are 0. $1$ is the all-ones vector of length $n$. A probability distribution over vertices, in general, is just a vector $s$ with $||s||_1 = 1$. 


For a subset of vertices $S \subseteq V$, we define the \textit{volume} of $S$ to be:

$$\text{vol}(S) = \sum_{v \in S} d_v.$$ 

By convention, the volume of a graph $G$, $\text{vol}(G)$, is just equal to the volume of its entire vertex set $V$. If $G$ has no self-loops, then $\text{vol}(G) = 2m$.

We denote by $A$ the \textit{adjacency matrix} of $G$, the $n \times n$ matrix where

$$A_{ij} = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E, \\ 0 & \text{otherwise.} \end{cases}$$

In many cases, we will refer to vectors and matrices restricted to a subset $S \subseteq V$. For a graph-theoretic matrix $M$, $M_S$ is the $|S| \times |S|$ matrix that is simply $M$ restricted to the rows and columns corresponding to vertices in $S$, without any re-normalizing of the matrix entries. For a vector $\mathbf{v}$, $\mathbf{v}_S$ is the vector of length $|S|$ containing components corresponding to the vertices in $S$.

2.1 Random walks and personalized PageRank

Many of the topics in this dissertation will involve studying, directly or indirectly, some kind of spreading or movement between nodes along a graph’s edges. There are many basic processes that fit this model; for example, Web users navigating from site to site by clicking hyperlinks, the spread of information based on word-of-mouth, and disease spreading among human populations based on direct contact. In this section, we present several tools that will be used throughout this dissertation to analyze these types of problems.

2.1.1 Random walks

A simple starting point for the study of diffusion on a graph, the \textit{random walk} is a Markov process where the states are the vertices $V$. At time step $t = 0$, the random walk starts at some vertex $u \in V$. Then at each time step $t$, the process moves from its current vertex $u$ to one of its neighbors $v$ chosen uniformly
at random from the neighbors of $u$. Formally, if $x(t)$ represents the current vertex at time $t$, then the state transitions can be described as

$$\Pr[x(t+1) = j \mid x(t) = i] = \begin{cases} \frac{1}{d_i} & \text{if } i \sim j, \\ 0 & \text{otherwise.} \end{cases}$$

These probabilities are collected in the random walk matrix $W = D^{-1}A$. The entries $W_{ij}$ are exactly the above probabilities.

If a random walk is started at vertex $x(0) = v$ and iterated for $t$ steps, then the probability distribution for $x(t)$ is given by $p(t) = 1_v W^t$. This process can be made more general to start from a probability distribution $s$ over vertices, but in most cases we are interested in the limiting behavior as $t \to \infty$. As a consequence of the Perron-Frobenius theorem (as shown in [35]), as long as $G$ is connected and not bipartite, $1_v$ is an eigenvector of $A$ with eigenvalue 1, and all other eigenvalues are smaller. Thus, in this case, the random walk has a stationary distribution $\pi$:

$$\pi = 1D^{-1} = \frac{d}{\text{vol}(G)},$$

and the random walk from any starting distribution $s$ will converge to $\pi$. (See Figure 2.1 for an example stationary distribution. Darker shades of red indicate more probability mass.) We will refer to this stationary distribution $\pi$ throughout the dissertation, and this implicitly assumes that the given $G$ is connected and not bipartite. For a vertex $v$, the component $\pi(v) = d_v/\text{vol}(G)$ is the stationary probability, and for a subset of vertices $S \subseteq V$, $\pi(S) = \sum_{v \in S} \pi(v) = \text{vol}(S)/\text{vol}(G)$.

This dissertation will also make reference to the lazy random walk characterized by the lazy random walk matrix $Z = (I + W)/2$. It is “lazy” because it models a random walk process that takes the random walk step with probability $1/2$ and otherwise stays put at the current vertex. The stationary distribution $\pi$ is the same for $W$ and $Z$, and the lazy random walk will converge even when $G$ is bipartite. Thus, $Z$ is sometimes used in place of $W$ without loss of generality.

The random walk is a well-studied model, but it turns out that that it is not always the most useful. In the next section, we will introduced personalized Page-Rank, another diffusion model. PageRank vectors are more flexible; parametrized for more control over their mixing properties.
Figure 2.1: The random-walk stationary distribution for a network [99], proportional to the degree distribution

2.1.2 Personalized PageRank vectors

Originally designed for Web search, PageRank [22] describes a modified random walk process. The diffusion process is modeled after a person “surfing” the Web, where the vertices represent Web pages and the edges represent hyperlinks. At each time step, with probability $(1 - \alpha)$, the surfer clicks a hyperlink and moves to a neighboring vertex. With the remaining probability $\alpha$, the surfer “jumps” to a vertex drawn from a probability distribution $s$.

Analogous to the random walk model from the previous section, as time goes to infinity, there is a stationary distribution of the probability that the random surfer is at a given node. We call this stationary distribution the *PageRank vector* $\text{pr}(\alpha, s)$. Note that there are two parameters: the *jumping constant* $\alpha$ and the *starting distribution* or *seed vector* $s$. It is assumed that $\|s\|_1 = 1$.

The above random surfer model can be expressed as a linear system of equations. For a given $\alpha$ and $s$, the PageRank vector is the unique solution to the
following PageRank equation:

\[ \text{pr}(\alpha, s) = \alpha s + (1 - \alpha)\text{pr}(\alpha, s)W. \]  

(2.1)

When \( \alpha = 0 \), this reduces to the random walk from the previous section, and when \( \alpha = 1 \), it is just the starting distribution \( s \). The original definition of PageRank used \( s = 1/n \), the uniform distribution. PageRank vectors with arbitrary \( s \) were introduced in [72, 75] and are usually called personalized PageRank vectors. Often times we are interested in the case where \( s = 1_v \) for some single vertex \( v \). We write \( \text{pr}(\alpha, v) \) in this case, a slight abuse of notation. When we are interested in just the \( i \)th component of the PageRank vector, we write \( \text{pr}(\alpha, s)_i \), and when we are interested in the PageRank of a subset of vertices \( S \), we write \( \text{pr}(\alpha, s)_S = \sum_{i \in S} \text{pr}(\alpha, s)_i \).

Several visualizations of PageRank vectors appear in Figure 2.2. Figures 2.2a and 2.2b show the effect of changing \( \alpha \) while keeping the same \( s \): when the jumping constant gets larger, the PageRank vector becomes more tightly concentrated around \( s \). Thus, we observe that \( \alpha \) controls the rate of diffusion of the PageRank process. Figs. 2.2c and 2.2d show the effect of changing \( s \) with the same \( \alpha \). In all figures, darker shades of red indicate more probability mass. The darkest red in each image corresponds to the largest component.

It turns out that we can substitute the lazy random walk \( Z \) for \( W \) in 2.1, and the two definitions are equivalent up to a change in \( \alpha \) [8]. The PageRank vector can also be expressed as a geometric sum of random walks:

\[ \text{pr}(\alpha, s) = \alpha s + \alpha \sum_{t=1}^{\infty} (1 - \alpha)^t s W^t, \]  

(2.2)

showing that PageRank vectors are linear in \( s \). Note that this series converges because the eigenvalues of \( W \) (and \( Z \)) are all in \([-1, 1]\) [35].

We shall see throughout this dissertation that PageRank vectors can be powerful algorithmic tools. One concern, however, is the complexity of computing them. The naive method of solving (2.1) directly takes \( O(n^3) \) time which is too expensive for large networks. A more common algorithm is to simulate the random surfer model directly, starting from \( s \), until the probability distribution converges.
Figure 2.2: PageRank vectors for a network [99], showing the effects of changing $\alpha$ and $s$

This is equivalent to repeatedly applying the linear transformation $R_\alpha = \alpha + \sum_{t=1}^{\infty} (1 - \alpha)^t W^t$ [8]. However, on the large networks that are of interest, such as the Internet and social graphs, even this method is too slow.

Instead of computing exact PageRank vectors, we will compute approximations [8]:

**Definition 1.** An $\epsilon$-approximate PageRank vector for $\text{pr}(\alpha, s)$ is a PageRank vector $\text{pr}(\alpha, s - r)$ where the vector $r$ is nonnegative and satisfies $r_v \leq \epsilon d_v$ for every vertex $v$ in the graph.
It has been shown [8] that for a subset of vertices \( S \subseteq V \), the approximation error is at most \( \epsilon \) \( \text{vol}(S) \):

\[
pr(\alpha, s)_S \geq pr(\alpha, s - r)_S \geq pr(\alpha, s)_S - \epsilon \text{vol}(S).
\]

The advantage of using \( \epsilon \)-approximate PageRank vectors is that they can be computed efficiently. The algorithm \textbf{ApproximatePR} in [8] takes \( O\left(\frac{1}{\alpha \epsilon}\right) \) time, and \textbf{SharpApproximatePR} in [46] takes \( O\left(\frac{m \log(1/\epsilon)}{\alpha}\right) \) time. These algorithms use computations that are local to the neighborhood around \( s \) and, in general, do not need to process the entire graph.

### 2.2 The normalized graph Laplacian and its spectrum

In this dissertation, we will spend several chapters analyzing various graph clustering algorithms (see Chapters 3, 5, and 6). Many clustering algorithms make extensive use of the spectrum of a graph: the eigenvalues and eigenvectors of a graph-theoretic matrix. Unfortunately, the usual adjacency matrix \( A \) often does not work very well, especially for graphs with non-uniform degree distributions. (This problem will appear prominently in Chapter 6.) Instead, many algorithms and analyses use the \textit{normalized Laplacian} \( L \), which has some useful properties.

For two vertices \( u \) and \( v \), the corresponding matrix entry in the normalized Laplacian is:

\[
L_{uv} = \begin{cases} 
1 & \text{if } u = v, \\
\frac{1}{\sqrt{d_u d_v}} & \text{if } u \sim v \\
0 & \text{otherwise},
\end{cases}
\]

where \( d_u \) and \( d_v \) are the degrees of \( u \) and \( v \).

We can also express \( L \) in terms of the adjacency matrix:

\[
L = D^{-1/2}(D - A)D^{-1/2} = I - D^{-1/2}AD^{-1/2}.
\]

Note that \( L \) is symmetric. There are other variants of the Laplacian, including asymmetric matrices, and several of them are discussed in Chapter 6. The matrix
\( \mathcal{L} \) has 0 as an eigenvalue, and the entire spectrum satisfies

\[
0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \leq 2.
\] (2.3)

The multiplicity of the eigenvalue 0 corresponds to the number of connected components in \( G \), and the largest eigenvalue is exactly 2 if and only if \( G \) is bipartite.

Of particular interest is the spectral gap, the second-smallest eigenvalue (\( \lambda_2 \) in (2.3)). We will denote by \( \lambda(\mathcal{L}) \) the spectral gap. As we will see in the following section, \( \lambda(\mathcal{L}) \) is related to bottlenecks in graphs, tying together diffusion and clustering. More facts and properties of the normalized Laplacian appear in [35].

### 2.3 Graph cuts and the Cheeger ratio

A cut in a graph \( G = (V, E) \) is a partition of the vertices into two disjoint sets \( S, \bar{S} \) (where \( \bar{S} = V \setminus S \)). Such a partition severs the connectivity of the graph, and one can talk about the edges crossing the cut. In particular, the edge boundary \( \partial S \) of a set \( S \) is the number of edges crossing the cut \( (S, \bar{S}) \):

\[
\partial S = \{ (u, v) \mid u \in S, v \notin S \}.
\]

We will also refer to the vertex boundary \( \delta S \):

\[
\delta S = \{ u \mid u \notin S \text{ and } \exists v \in S, (v, u) \in E \}.
\]

Graph cuts are important when studying clustering or community structure within a larger network. In particular, a good cluster \( S \) is often characterized as being well-connected within \( S \) compared to its connections outside of \( S \). For any vertex set \( S \), we quantify this property as the Cheeger ratio \( h(S) \) [35]:

\[
h(S) = \frac{|\partial S|}{\min(\text{vol}(S), \text{vol}(\bar{S}))}.
\]

The Cheeger ratio gives a good metric for describing a particular graph cut: small \( h(S) \) means that \( S \) is well-separated from \( \bar{S} \) as compared to its connectivity within \( S \). This can be seen when examining two cuts in the same network in Figure 2.3.
The cut in Figure 2.3a makes for a worse partitioning than that in Figure 2.3b, and it indeed has a much higher Cheeger ratio. For an entire graph $G$, we also have the related *Cheeger constant* $h(G)$ which is the Cheeger ratio of the “best” cut in $G$:

$$h(G) = \min_{S \subseteq V} h(S).$$

![Figure 2.3: Two different partitions of the same network [99] and their Cheeger ratios](image)

(a) $h(S) = 1$  (b) $h(S) = 0.0645$

This dissertation concerns both diffusion and clustering on large networks. While these two concepts may seem disparate at first, they are actually closely related. Good clusters are characterized by low Cheeger ratio, and it turns out that diffusive processes that start in a well-separated cluster find it difficult to cross the cluster boundary. Lovász and Simonovits [97, 98] first proved that if a cluster $S$ has low Cheeger ratio, there are many vertices $v \in S$ such that a random walk started at $v$ takes a long time, with high probability, to leave $S$. For PageRank, Andersen, Chung, and Lang proved the following analogous result:

**Lemma 2.1** ([7, 8]). *Let $C$ be a subset of $V$ with Cheeger ratio $h$, and let $\alpha$ be a constant in $(0, 1]$. Then there is a subset $C_\alpha \subseteq C$ with volume $\text{vol}(C_\alpha) \geq \text{vol}(C)/2$ such that for any vertex $v \in C_\alpha$, the PageRank vector $\text{pr}(\alpha, v)$ satisfies:

$$\text{pr}(\alpha, v)_C \geq 1 - \frac{h}{\alpha}.$$*
This lemma implies that for a cluster $C$ with small Cheeger ratio, there are many nodes $v$ within $C$ for which the personalized PageRank vector, using $v$ as a starting point, has low mass outside of $C$. We call the set of such $v$ the core $C_\alpha$ of $v$. This and similar lemmas will be used several times in this dissertation. For an example of the core of a partition, refer to Figure 2.4. Here $C$ is comprised of the blue and green vertices, the core $C_\alpha \subseteq C$ is the set of green vertices, and $\bar{C}$ is the set of red vertices. A personalized PageRank vector with starting distribution $C_\alpha$ has low mass outside of $C$ (Figure 2.5a), whereas it can still diffuse outside of $C$ when started within $C$ but outside the core $C_\alpha$ (Figure 2.5b).

![Figure 2.4: A partition of a network [99], showing the core of one side](image)

In the previous section, we discussed the spectral gap $\lambda(\mathcal{L})$ of the normalized Laplacian. The Cheeger constant and normalized Laplacian spectral gap are related by the following Cheeger inequality [35]:

$$2h(G) \geq \lambda(\mathcal{L}) \geq \frac{h(G)^2}{2}.$$ 

Both $\lambda(\mathcal{L})$ and $h_G$ are often used to characterize expansion or bottlenecks in graphs. This inequality shows that they are both good candidates and gives the ability to estimate one based on the other.
Figure 2.5: Comparison of PageRank vectors when the starting distribution starts from inside or outside the core of the partition given in Figure 2.4
Chapter 3

Global Graph Clustering Using PageRank Optimization

Finding smaller local communities within a larger graph is a well-studied problem with many applications. For example, advertisers can more effectively serve niche audiences if they can identify their target communities within the larger social web, and viruses on technological or population networks can be effectively quarantined by distributing antidote to local clusters around their origins [37].

As discussed in Section 1.1.2, there are numerous well-known algorithms for finding clusters within a graph, including $k$-means [96, 101], spectral clustering [112, 122], Markov cluster algorithms [57], and numerous others. Many of these algorithms require embedding a graph into low-dimensional Euclidean space using pairwise distances, but graph distance-based metrics fail to capture graph structure in real-world networks with small-world phenomena since all pairs of vertices are connected within short distances. PageRank provides essential structural relationships between vertices and is particularly well suited for clustering analysis. Furthermore, PageRank vectors can be computed more efficiently than performing a dimension reduction for a large graph.

In this chapter, we give clustering algorithms PageRank-Clustering that use PageRank vectors to draw attention to local graph structure within a larger network. Using PageRank can capture well the quantitative correlations between pairs or subsets of vertices, especially on small-world graphs where the usual graph
distances are all quite small. We use PageRank vectors to define a notion of PageRank distance which provides a natural metric space more appropriate for graphs than graph distance.

PageRank vectors require selecting the jumping constant $\alpha$; in our clustering algorithms, we will use $\alpha$ to control the scale of the clustering. In particular, we introduce two variance measures which can be used to automatically find the optimized values for $\alpha$. We then use PageRank vectors determined by this $\alpha$ to guide the selection of a set of centers of mass. Once we have $k$ centers, we can use them to find the clusters via PageRank distances.

The chapter is organized as follows: we describe the PageRank distance metric and two cluster variance measures using PageRank vectors in Section 3.1, and we give clustering algorithms in Section 3.2. In Section 3.3, we present several PageRank diffusion properties that are used to complete the algorithms’ analysis in Section 3.4.

### 3.1 A distance metric and PageRank variance measures for graph clustering

As previously mentioned, many clustering algorithms require a notion of pairwise distances. Such distance metrics are natural with points in Euclidean space, but in a graph-theoretic setting, the graph shortest-path distance is not as useful: with the small-world phenomenon, many graphs have short distance between all pairs of vertices [132]. Instead, we will define a new metric that is more suitable. For two vertices $u, v \in V$, we define the PageRank distance with jumping constant $\alpha$ as:

$$\text{dist}_\alpha(u, v) = \left\| \text{pr}(\alpha, u)D^{-1/2} - \text{pr}(\alpha, v)D^{-1/2} \right\|_2.$$

Choosing the $L_2$-norm will allow for differentiation in the optimization process later.

We can further generalize this distance to two probability distributions $p$ and $q$ defined on the vertex set $V$ of $G$. Namely, the PageRank distance, with
jumping constant \( \alpha \), between \( p \) and \( q \) is defined by

\[
\text{dist}_\alpha(p, q) = \sum_{u,v} p(u)q(v)\text{dist}_\alpha(u, v).
\]

We will use the PageRank distance in a precise definition of a center of mass for a subset of vertices \( S \subseteq V \). Because PageRank distance can be used for probability distributions, our notion of a “center” can be a single vertex \( v \) or a probability distribution \( c \) over vertices in \( S \). For a given \( \epsilon > 0 \), we say \( c \) is an \( \epsilon \)-center or center of mass for \( S \) if

\[
\sum_{v \in S} \text{dist}_\alpha(c, v) \leq \epsilon.
\]

Let \( C \) denote a set of \( k \) potential centers. The eventual goal is for each \( c \in C \) to be a representative center of mass for some cluster of vertices. We let \( R_c \) denote the set of all vertices \( x \) which are closest to \( c \) in terms of PageRank distance, provided the jumping constant \( \alpha \) is given:

\[
R_c = \{ x \in V : \text{dist}_\alpha(c, x) \leq \text{dist}_\alpha(c', x) \text{ for all } c' \in C \}.
\]

For a vertex \( v \) and a set of centers \( C \), let \( c_v \) denote the center that is closest to \( v \) (i.e., \( c_v \) is the center of mass \( c \in C \) such that \( v \in R_c \)).

We will follow the approach as in \( k \)-means by defining the following evaluative measure for a potential set of \( k \) centers \( C \), using PageRank instead of Euclidean distances:

\[
\mu(C) = \sum_{v \in V} d_v \left\| \text{pr}(\alpha, v)D^{-1/2} - \text{pr}(\alpha, c_v)D^{-1/2} \right\|_2^2
\]

\[
= \sum_{v \in V} d_v \text{dist}_\alpha(v, c_v)^2.
\]

The metric \( \mu(C) \) provides a good measure of internal cluster connectivity. If \( \mu(C) \) is small, then the PageRank vectors for vertices \( v \in V \) are close to the PageRank vector for the center of mass \( c_v \). We remark that this measure is essentially the analog of \( k \)-means in terms of PageRank distance, and it has a similar flavor as a heuristic given by Dyer and Frieze [54] for the traditional center selection problem.
Selecting the best set of representative centers within a graph is a hard optimization problem, known to be NP-complete to solve exactly [5]. There are many approximate and heuristic algorithms used in practice (see [120]). The algorithms presented in Section 3.2 will use personalized PageRank vectors to select the centers.

A small $\mu(C)$ indicates that the clusters are well-connected internally, but we also want well-separated clusters. To measure this, we use the following evaluative metric $\Psi_\alpha(C)$, suggesting the structural separation of the communities represented by centers in $C$:

$$\Psi_\alpha(C) = \sum_{c \in C} \text{vol}(R_c) \left\| \text{pr}(\alpha, c)D^{-1/2} - \pi D^{-1/2} \right\|_2^2 = \sum_{c \in C} \text{vol}(R_c) \text{dist}_\alpha(c, \pi)^2. \quad (3.1)$$

If $\Psi_\alpha(C)$ is large, then there are large discrepancies between the PageRank vector for the centers $c$ and the overall stationary distribution of the graph $\pi$. Note that the personalized PageRank vector with $\pi$ as the seed is $\pi$ itself, and if $\text{dist}_\alpha(c, \pi)^2$ is large, then the cluster represented by $c$ is distinct from the graph as a whole. If $\Psi_\alpha(C)$ is large, then the set of clusters is distinct from the graph, and therefore, also from each other.

Finding a $C$ that optimizes $\mu(C)$ and $\Psi_\alpha(C)$ is computationally intractable, so instead, we will work primarily with two related metrics, $\Phi(\alpha)$ and $\Psi(\alpha)$. These metrics will serve two purposes. First, they do not depend on a specific set of centers $C$, just the graph and jumping constant $\alpha$. They give insight into the graph structure and whether it is suitable for clustering in the first place. Second, it will turn out that $\Phi(\alpha)$ and $\Psi(\alpha)$ are good estimators for $\mu(C)$ and $\Psi_\alpha(C)$ when the centers $C$ are selected randomly according to various PageRank distributions. We will use this as an algorithmic tool in Section 3.2, and because these metrics only depend on $\alpha$, they are much more tractable to optimize. While we are really interested in optimizing $\mu(C)$ and $\Psi_\alpha(C)$, we can get away with solving the more tractable problem of finding an $\alpha$ that optimizes $\Phi(\alpha)$ and $\Psi(\alpha)$. The metrics are
as follows:

\[
\Phi(\alpha) = \sum_{v \in V} d_v \left\| \text{pr}(\alpha, v) D^{-1/2} - \text{pr}(\alpha, \text{pr}(\alpha, v)) D^{-1/2} \right\|_2^2 \\
= \sum_{v \in V} d_v \text{dist}_\alpha(v, \text{pr}(\alpha, v))^2,
\]

\[
\Psi(\alpha) = \sum_{v \in V} d_v \left\| \text{pr}(\alpha, \text{pr}(\alpha, v)) D^{-1/2} - \pi D^{-1/2} \right\|_2^2 \\
= \sum_{v \in V} d_v \text{dist}_\alpha(\text{pr}(\alpha, v), \pi)^2.
\]

The \textit{\(\alpha\)-PageRank-variance} \(\Phi(\alpha)\) measures discrepancies between the personalized PageRank vectors for vertices \(v\) and possible centers nearest to \(v\), represented by the probability distribution \(\text{pr}(\alpha, v)\). The \textit{\(\alpha\)-cluster-variance} \(\Psi(\alpha)\) measures large discrepancies between personalized PageRank vectors for vertices \(v\) and the overall stationary distribution \(\pi\). If the PageRank-variance \(\Phi(\alpha)\) is small, then the ‘guesses’ by using PageRank vectors for the centers of mass give a good upper bound for the \(k\)-means evaluation \(\mu\) using PageRank distance, indicating the formation of clusters. If the cluster-variance \(\Psi(\alpha)\) is large, then the centers of masses using the predictions from PageRank vectors are quite far from the stationary distribution, capturing a community structure. Thus, our goal is to find the appropriate \(\alpha\) such that \(\Phi(\alpha)\) is small but \(\Psi(\alpha)\) is large.

### 3.2 The PageRank-Clustering algorithms

These evaluative metrics presented in the previous section provide us a way to evaluate a set of community centers \(C\), leading to the \textbf{PageRank-Clustering} algorithms presented here. The problem of finding a set of \(k\) centers minimizing \(\mu(C)\) with large \(\Phi_\alpha(C)\) is then reduced to the problem of minimizing \(\Phi(\alpha)\) while \(\Psi(\alpha)\) is large for appropriate \(\alpha\). In particular, for a special class of graphs which consist of \(k\) clusters of vertices where each cluster has a bounded Cheeger ratio, the center selection algorithm is guaranteed to be successful with high probability.

A natural question is to find the appropriate \(\alpha\) for a given graph, if such \(\alpha\) exists and if the graph is clusterable in the first place. A direct method is by
computing the variance metrics for a sample of \( \alpha \) and narrowing down the range for \( \alpha \) using binary search. But we can also consider a systematic method for determining the existence of an appropriate \( \alpha \) and finding its value: differentiate \( \Phi(\alpha) \) and find roots \( \alpha \) satisfying \( \Phi'(\alpha) = 0 \). The derivative of \( \Phi \) is given by:

\[
\Phi'(\alpha) = \frac{1 - \alpha}{\alpha^3} \left( \sum_{v \in V} \left| \left| g_v(\alpha)D^{-1/2} \right| \right|^2_2 - 2\langle g_v(\alpha), \text{pr}(\alpha, g_v(\alpha))D^{-1} \rangle \right),
\]

where

\[
g_v(\alpha) = \text{pr}(\alpha, \text{pr}(\alpha, v)(I - W)).
\]

We present two versions of the clustering algorithm. For the sake of clarity, the first PageRank clustering algorithm, \textbf{PageRank-ClusteringA}, uses exact PageRank vectors without approximation. This allows for a clean analysis at the expense of slower running time. The second PageRank clustering algorithm, \textbf{PageRank-ClusteringB}, allows for the use of approximate PageRank vectors as well as approximate PageRank-variance and cluster-variance for faster performance.

\begin{algorithm}
\textbf{PageRank-ClusteringA}

\textbf{Input}: a graph \( G \), the number of clusters \( k \geq 2 \), and an approximation parameter \( \epsilon > 0 \).

\textbf{Output}: a set of centers \( C \) and partitions \( S \), or nothing.

- For all \( v \in G \), compute \( \text{pr}(\alpha, v) \).
- Find the roots of \( \Phi'(\alpha) \). (There can be more than one root if \( G \) has a layered clustering structure.)
- For each root \( \alpha \):
  - Compute \( \Phi(\alpha) \).
  - If \( \Phi(\alpha) \leq \epsilon \), then compute \( \Psi(\alpha) \). Otherwise, continue to the next \( \alpha \).
  - If \( k < \Psi(\alpha) - 2 - \epsilon \):
* Select $c \log n$ sets of $k$ potential centers, randomly chosen according to $\pi$.

* For each set $S = \{v_1, \ldots, v_k\}$:
  - Let $C$ be the set of centers of mass comprised of $c_i = \text{pr}(\alpha, v_i)$, for all $v_i \in S$.
  - Compute $\mu(C)$ and $\Psi_\alpha(C)$.
  - If $|\mu(C) - \Phi(\alpha)| \leq \epsilon$ and $|\Psi_\alpha(C) - \Psi(\alpha)| \leq \epsilon$, then determine the clusters $R_c$ according to PageRank distance and return them.
    - Otherwise, continue to the next $\alpha$.

We can further reduce the computational complexity by using approximate PageRank vectors in algorithm **PageRank-ClusteringB**.

**Algorithm PageRank-ClusteringB**

Input: a graph $G$, the number of clusters $k \geq 2$, and approximation parameters $\epsilon, \delta > 0$.

Output: a set of centers $C$ and partitions $S$, or nothing.

- For all $v \in G$, compute an approximate $\text{pr}(\alpha, v)$.
- Approximate the roots of $\Phi'(\alpha)$. This can be done using sampling techniques from [119] with $O(\log n)$ vertices, $\log(1/\epsilon)$ values of $\alpha$, and $\delta$-approximate PageRank vectors [8, 46].
- For each root $\alpha$:
  - Compute $\Phi(\alpha)$.
  - If $\Phi(\alpha) \leq \epsilon$, then compute $\Psi(\alpha)$. Otherwise, continue to the next $\alpha$.
  - If $k < \Psi(\alpha) - 2 - \epsilon$: 
* Select \( c \log n \) sets of \( k \) potential centers, randomly chosen according to \( \pi \).

* For each set \( S = \{v_1, \ldots, v_k\} \):
  
  \begin{itemize}
  \item Let \( C \) be the set of centers of mass comprised of \( c_i = \text{pr}(\alpha, v_i) \), for all \( v_i \in S \).
  \item Compute \( \mu(C) \) and \( \Psi_\alpha(C) \).
  \item If \( |\mu(C) - \Phi(\alpha)| \leq \epsilon \) and \( |\Psi_\alpha(C) - \Psi(\alpha)| \leq \epsilon \)
  then determine the clusters \( R_c \) according to PageRank distance and return them.
  \end{itemize}

  – Otherwise, continue to the next \( \alpha \).

We remark that by using the sharp approximate PageRank algorithm in [46], the error bound \( \delta = \epsilon/2 \) for PageRank can be set to be quite small since the time complexity is proportional to \( \log(1/\delta) \). If we choose \( \delta \) to be a negative power of \( n \) such as \( \delta = \epsilon/n^2 \), then approximate PageRank vectors lead to sharp estimates for \( \Phi \) and \( \Phi' \) within an error bound of \( \epsilon \). Thus for graphs with \( k \) clusters, the \textbf{PageRank-ClusteringB} algorithm will terminate after approximating the roots of \( \Phi', O(k \log n) \) approximations of \( \mu \) and \( \Psi_\alpha \) and \( O(n) \) approximate PageRank computations. By using approximation algorithms based on sampling, this can be done quite efficiently.

We also note that there might be no clustering output if the conditions set within the algorithms are not satisfied. Indeed, there exist graphs that inherently do not have a \( k \)-clustered structure within the error bound that we set for \( \epsilon \). Another reason for no output is the probabilistic nature of the above sampling method. We will provide evidence to the correctness of the above algorithm by showing that, with high probability, a graph with a \( k \)-clustered structure will have outputs that capture its clusters in a feasible manner which we will specify further.

We say a graph \( G \) is \((k, h, \beta, \epsilon)\)-clusterable if the vertices of \( G \) can be partitioned into \( k \) parts so that:
1. Each part $S_i$ has Cheeger ratio at most $h$,

2. each $S_i$ has volume at least $\beta \text{vol}(G)/k$ for some constant $\beta$, and

3. for each $S_i$, any subset $S'_i \subset S_i$, with $\text{vol}(S'_i) \leq (1-\epsilon)\text{vol}(S_i)$, has its Cheeger ratio at least $c\sqrt{h\log n}$ where $c = 8\sqrt{\beta/k}/\epsilon$.

We will prove the following theorem about PageRank-Clustering:

**Theorem 3.1.** Suppose a graph $G$ has an $(k, h, \beta, \epsilon)$-clustering and $\alpha, \epsilon \in (0, 1)$ satisfy $\epsilon \geq h\epsilon/(2\alpha\beta)$. Then with high probability, PageRank-Clustering returns a set $C$ of $k$ centers with:

1. $\Phi(\alpha) \leq \epsilon$,

2. $\Psi(\alpha) > k - 2 - \epsilon$, and

3. the $k$ clusters are near optimal according to the PageRank $k$-means measure $\mu(C)$ with an additive error term $\epsilon$.

To show the appropriateness of the method of selecting $\alpha$ in PageRank-Clustering, we consider a dumbbell graph $U$ as an example. This graph $U$ has two complete graphs $K_{20}$ connected by a single edge, yielding a Cheeger ratio of $h \approx 0.0026$. Plotting $\Phi(\alpha)$ (Figure 3.1a) and its derivative (Figure 3.1b) shows that there is a local minimum near $\alpha \approx 0.018$. When $\Psi$ is large, many individual vertices have personalized PageRank vectors that differ greatly from the overall distribution. This indicates that there are many vertices that are more representative of a small cluster than the entire graph. By plotting $\Psi(\alpha)$ (Figure 3.1c) and its derivative (Figure 3.1d), we can see that there is a distinct inflection point in the plot of $\Psi$ for the dumbbell graph $U$ as well.

### 3.3 Some PageRank diffusion lemmas

Before proceeding to show that the PageRank-Clustering algorithms are effective for treating clusterable graphs, we will first establish some useful tools for analyzing PageRank vectors. These tools concern the diffusion of PageRank vectors.
(a) $\Phi(\alpha)$

(b) $\Phi'(\alpha)$, with the line $y = 0$ for reference

(c) $\Psi(\alpha)$

(d) $\Psi'(\alpha)$

Figure 3.1: Graph variance metrics for the dumbbell graph $U$
vectors in a subset of vertices with small Cheeger ratio. Before we examine a general mixing inequality involving PageRank vectors, first we consider a diffusion lower bound which is a slightly modified version of the results in [8].

**Lemma 3.2** ([8]). For any set $S$ and any constants $\alpha, \delta$ in $(0, 1]$, there is a subset $S_\alpha \subseteq S$ with volume $\text{vol}(S_\alpha) \geq (1 - \delta)\text{vol}(S)$ such that for any vertex $v \in S_\alpha$, the PageRank vector $\text{pr}(\alpha, v)$ satisfies

$[\text{pr}(\alpha, v)](S) \geq 1 - \frac{h(S)}{2\alpha\delta}$.

We will use the notation that for any function $f : V \to \mathbb{R}$, $f(S) = \sum_{v \in S} f(v)$ for $S \subseteq V$. For a positive real value $x$, we define

$f(x) = \max \left\{ \sum_v \frac{\beta_v}{d_v} f(v) : \sum_v \beta_v = x, 0 \leq \beta_v \leq d_v \right\}$.

This leads to many nice properties of $f$ including, for example that $f$ is concave and that $f(\text{vol}(S)) \geq f(S)$ (see [8, 97]). We use $[f](x)$ for clarity when $f$ is a complex vector expression.

**Lemma 3.3.** For any set $S$ and any constants $\alpha, \delta$ in $(0, 1]$, there is a subset $S_\alpha \subseteq S$ with volume $\text{vol}(S_\alpha) \geq (1 - \delta)\text{vol}(S)$ such that for any vertex $v \in S_\alpha$, the PageRank vector $\text{pr}(\alpha, \text{pr}(\alpha, v))$ satisfies

$[\text{pr}(\alpha, \text{pr}(\alpha, v))](S) \geq 1 - \frac{h(S)}{\alpha\delta}$.

**Proof.** The proof is quite similar to that in [8]. Let $\chi_S$ denote the function of $S$ which assumes the value $\chi_S(x) = d_v/\text{vol}(S)$ if $x \in S$ and 0 otherwise. First we wish to show:

$[\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))](S) \leq h(S) \frac{1 - \alpha}{\alpha}$.\[
\]During a single step from $\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))$ to $\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))W$, the amount of probability that moves from $S$ to $\tilde{S}$ is bounded from above by

$[\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))W](\tilde{S}) \leq [\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))](S) + \frac{1}{2}[\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))](|\partial S|)$ (3.3)
where \( \partial S \) is the edge boundary of \( S \). By using the definition of PageRank (Equation (2.1)), we obtain

\[
[\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))](\bar{S}) = \alpha[\text{pr}(\alpha, \chi_S)](\bar{S}) + (1 - \alpha)[\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))W](\bar{S}) \\
\leq \frac{1 - \alpha}{2} h(S) + (1 - \alpha)[\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))W](\bar{S})
\]

by using Theorem 4 in [8] (inequality (8), specifically). From (3.3), we have

\[
[\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))W](\bar{S}) = \frac{1 - \alpha}{2} h(S) + (1 - \alpha)[\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))](\bar{S}) + \frac{1 - \alpha}{2}[\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))](|\partial S|).
\]

This implies

\[
[\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))](S) = \frac{1 - \alpha}{2} h(S) + \frac{1 - \alpha}{2}[\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))](|\partial S|).
\]

Now we use the monotonicity property from Lemma 4 in [8]; we have

\[
[\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))](|\partial S|) \leq [\text{pr}(\alpha, \chi_S)](|\partial S|) \\
\leq \chi_S(|\partial S|) \\
= \frac{|\partial S|}{\text{vol}(S)} \\
= h(S).
\]

Thus we have

\[
[\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))](\bar{S}) \leq \frac{1 - \alpha}{\alpha} h(S).
\]

To complete the proof, let \( S_\alpha \) denote the set of vertices \( v \in S \) satisfying

\[
[\text{pr}(\alpha, \text{pr}(\alpha, v))](\bar{S}) \leq \frac{h(S)}{\alpha \delta}.
\]

Let \( v \) be a vertex chosen randomly from the distribution \( d_v/\text{vol}(S) \), and define the random variable \( X = [\text{pr}(\alpha, \text{pr}(\alpha, v))](\bar{S}) \). The linearity property of PageRank vectors implies that

\[
\mathbb{E}[X] = [\text{pr}(\alpha, \text{pr}(\alpha, \chi_S))](\bar{S}) \leq \frac{1 - \alpha}{\alpha} h(S) \leq \frac{h(S)}{\alpha}.
\]
Applying Markov’s inequality, we have
\[ \Pr[v \notin S_{\alpha}] \leq \Pr[X \geq E[X]/\delta] \leq \delta. \]
This completes the proof of Lemma 3.3.

We will also need the quantitative estimates for PageRank vectors restricted to a subset \( S \) of vertices. By considering submatrices \( W_S \) restricted to rows and columns associated with vertices in \( S \), we can define the Dirichlet PageRank \( pr_S(\alpha, s) \) for a jumping constant \( 0 \leq \alpha < 1 \) and seed vector \( s \) satisfying:
\[ pr_S(\alpha, s) = \alpha s + (1 - \alpha)pr_S(\alpha, s)W_S. \]
Dirichlet PageRank will be studied in further depth in Chapter 7. When \( \alpha \) is appropriately chosen, the PageRank is a good estimate of personalized PageRank vectors. Lemma 5 and Theorem 6 in [33] can be rewritten as follows:

**Lemma 3.4 ([33]).** Suppose a subset \( S \) of vertices has its Cheeger ratio \( h(S) \) satisfying \( \epsilon \geq \frac{(1 - \alpha)h(S)}{2\alpha} \), for positive values \( \alpha, \epsilon \). Then \( pr_S \) satisfies the following: for any \( R \subseteq S \), there is a subset \( T \subseteq S \) with \( \text{vol}(T) \geq (1 - \delta)\text{vol}(S) \), so that for every \( v \in T \) we have
\[ [pr(\alpha, v)](R) - [pr_S(\alpha, v)](R) \leq \sqrt{\frac{\epsilon}{\delta}}. \]

For a probability distribution \( f : V \to \mathbb{R} \) and a real value \( x \), we define the Cheeger ratio \( h_f(x) \) of \( f \) up to \( x \) as follows. Order the vertices \( v_1, \ldots, v_n \) from highest to lowest probability-per-degree, so that \( p(v_i)/d_i \geq p(v_{i+1})/d_{i+1} \). This produces a collection of sets, called the segment subsets, with one set \( T_j^f = \{v_1, \ldots, v_j\} \) for each \( j \leq n \). For a positive value \( x \leq \text{vol}(G) \), we define
\[
\begin{align*}
    h_f(x) &= \max\{h(T_j^f) : j \text{ satisfies } \text{vol}(T_j^f) \leq x\}; \\
    h_f^*(x) &= \max\{h(T_j^f) : j \text{ satisfies } \text{vol}(T_j^f) \leq x(1 + h_f(x))\}.
\end{align*}
\]

**Lemma 3.5 ([8]).** For a vertex \( v \in G \), any constant \( \alpha \in (0, 1] \), and a non-negative integer \( t \), the PageRank vector \( pr(\alpha, v) \) satisfies the following:
\[ [pr(\alpha, v)](T) - \pi(T) \leq \alpha t + \sqrt{\text{vol}(T)} \left(1 - \frac{\phi^2}{8}\right)^t \]
where \( \phi \) is the Cheeger ratio \( h_f^*(\text{vol}(T)) \) with \( f = pr(\alpha, v) \).
Lemma 3.6. For subsets $S, T$ of vertices in $G$ with $\text{vol}(S), \text{vol}(T) \leq \frac{\text{vol}(G)}{2}$, any constant $\alpha$ in $(0, 1]$, and a non-negative integer $t$, the Dirichlet PageRank vector $\text{pr}_S(\alpha, v)$ for any vertex $v \in S$ satisfies the following:

1. $[\text{pr}_S(\alpha, v)](T) - [\text{pr}_S(\alpha, \text{pr}_S(\alpha, v))](T) \leq \alpha t + \sqrt{\text{vol}(T)} \left(1 - \frac{\phi^2}{8}\right)^t$, where $\phi$ is the Cheeger ratio $h_f^*(\text{vol}(T))$ with $f = \text{pr}_S(\alpha, v) - \text{pr}_S(\alpha, \text{pr}_S(\alpha, v))$.

2. $[\text{pr}_S(\alpha, \text{pr}_S(\alpha, v))](T) - [\text{pr}_S(\alpha, v)](T) \leq \alpha t + \sqrt{\text{vol}(T)} \left(1 - \frac{\phi'^2}{8}\right)^t$, where $\phi'$ is the Cheeger ratio $h'_f^*(\text{vol}(T))$ with $f' = \text{pr}_S(\alpha, \text{pr}_S(\alpha, v)) - \text{pr}_S(\alpha, v)$.

3. For two vertices $u$ and $v$,

$$[\text{pr}_S(\alpha, u)](T) - [\text{pr}_S(\alpha, v)](T) \leq \alpha t + \sqrt{\text{vol}(T)} \left(1 - \frac{\phi''^2}{8}\right)^t,$$

where $\phi''$ is the Cheeger ratio $h''_f^*(\text{vol}(T))$ with $f'' = \text{pr}_S(\alpha, u) - \text{pr}_S(\alpha, v)$.

Proof. We prove by induction on $t$ for the first property. For $t = 0$, it holds. Suppose the above inequality holds for some $t \geq 0$. Let $x$ denote $\text{vol}(T)$. We use Lemma 3 of [8] and apply the same method using the concavity of $f$ to obtain:

$$[\text{pr}_S(\alpha, v)](T) - [\text{pr}_S(\alpha, \text{pr}_S(\alpha, v))](T) = f(T)$$

$$f(T) \leq \alpha + (1 - \alpha) [fW](T)$$

$$\leq \alpha + (1 - \alpha) \left(\frac{1}{2} f(x - \phi x) + \frac{1}{2} f(x + \phi x)\right)$$

$$\leq \alpha + \left(\frac{1}{2} f(x - \phi x) + \frac{1}{2} f(x + \phi x)\right)$$

Using the induction assumption, we have

$$f(T) \leq \alpha(t + 1) + \frac{1}{2} \left(\sqrt{x - \phi x} + \sqrt{x + \phi x}\right) \left(1 - \frac{\phi^2}{8}\right)^t$$

$$\leq \alpha(t + 1) + \sqrt{x} \left(1 - \frac{\phi^2}{8}\right)^{t+1}.$$
3.4 Analysis of PageRank-Clustering

In this section, we consider an \((h, k, \beta, \epsilon)\)-clusterable graph \(G\), with the following further condition:

\[
\epsilon \geq \frac{hk}{2\alpha\beta}.
\]

We will use the diffusion lemmas in the previous section to eventually prove Theorem 3.1.

Lemma 3.2 implies that in a cluster \(R\) of \(G\), most of the vertices \(u\) in \(R\) have \(\text{pr}(\alpha, u)(S) \geq 1 - \epsilon/(2k)\). This fact is essential in the subsequent proof that \(\Psi(\alpha) \geq k - 2 - \epsilon\).

We proceed with a series of lemmas that show that if \(G\) is \((h, k, \beta, \epsilon)\)-clusterable, there is an \(\alpha\) for which \(\Phi(\alpha)\) is small and \(\Psi(\alpha)\) is large, corresponding to a set of centers chosen from the core of the partitions.

**Lemma 3.7 (Large \(\Psi(\alpha)\)).** If a graph \(G\) can be partitioned into \(k\) clusters having Cheeger ratio at most \(h\) and \(\epsilon \geq hk/(2\alpha\beta)\), then \(\Psi(\alpha) \geq k - 2 - \epsilon\).

**Proof.** Let \(S_1, \ldots, S_k\) be a partition of \(G\) into \(k\) clusters satisfying the lemma conditions. Then, by definition of \(\Psi\),

\[
\Psi(\alpha) = \sum_{v \in V} d_v \left\| \text{pr}(\alpha, \text{pr}(\alpha, v))D^{-1/2} - \pi D^{-1/2} \right\|^2_2 \\
= \sum_{i=1}^{k} \sum_{v \in S_i} d_v \left\| \text{pr}(\alpha, \text{pr}(\alpha, v))D^{-1/2} - \pi D^{-1/2} \right\|^2_2 \\
= \sum_{i=1}^{k} \sum_{v \in S_i} \sum_{x \in V} \left( \text{pr}(\alpha, \text{pr}(\alpha, v))D^{-1/2}(x) - \pi D^{-1/2}(x) \right)^2 \\
\geq \sum_{i=1}^{k} \sum_{v \in S_i} \sum_{x \in S_i} \left( \text{pr}(\alpha, \text{pr}(\alpha, v))D^{-1/2}(x) - \pi D^{-1/2}(x) \right)^2 \\
= \sum_{i=1}^{k} \sum_{v \in S_i} \left( \sum_{x \in S_i} \left( \text{pr}(\alpha, \text{pr}(\alpha, v))D^{-1/2}(x) - \pi D^{-1/2}(x) \right)^2 \sum_{x \in S_i} \frac{d_x}{\text{vol}(S_i)} \right) \\
= \sum_{i=1}^{k} \sum_{v \in S_i} \left( \sum_{x \in S_i} \frac{1}{d_x} \left( \text{pr}(\alpha, \text{pr}(\alpha, v))(x) - \pi(x) \right)^2 \sum_{x \in S_i} \frac{d_x}{\text{vol}(S_i)} \right).
Using the Cauchy-Schwarz inequality and then Lemma 3.3, we have

\[
\Psi(\alpha) \geq \sum_{i=1}^{k} \sum_{v \in S_i} \frac{d_v}{\text{vol}(S_i)} \left( \sum_{x \in S_i} (\text{pr}(\alpha, \text{pr}(\alpha, v))(x) - \pi(x)) \right)^2 \\
\geq \sum_{i=1}^{k} \sum_{v \in S_i} \frac{d_v}{\text{vol}(S_i)} \left( 1 - \frac{\epsilon}{2} - \frac{\text{vol}(S_i)}{\text{vol}(G)} \right)^2 \\
= \sum_{i=1}^{k} \left( 1 - \frac{\epsilon}{2} - \frac{\text{vol}(S_i)}{\text{vol}(G)} \right)^2 \\
\geq \frac{1}{k} \left( \sum_{i=1}^{k} \left( 1 - \frac{\epsilon}{2} - \frac{\text{vol}(S_i)}{\text{vol}(G)} \right) \right)^2 \\
= \frac{1}{k} \left( k - 1 - \frac{\epsilon}{2} \right)^2 \\
\geq k - 2 - \epsilon.
\]

We have shown that if \( G \) has a clustered structure, then there is an \( \alpha \) for which \( \Psi(\alpha) \) is large. We will also show that our algorithm will also yield \( \Phi(\alpha) \leq \epsilon \).

**Lemma 3.8** (Small \( \Phi(\alpha) \)). If \( G \) is \((k, h, \beta, \epsilon)-clusterable\), then we have \( \Phi(\alpha) \leq \epsilon \).

**Proof.** The proof follows from preceding lemmas. Within each cluster \( S \) of \( G \), we first use Lemma 3.3 which implies there is a subset \( S' \) of \( S \) such that \([\text{pr}(\alpha, v)](S) \geq 1 - \epsilon/k\) and \(\text{vol}(S') \geq (1 - \delta)\text{vol}(S)\) since \( S \) has Cheeger ratio at most \( h \).

We can apply Lemma 3.4 so that we can approximate PageRank vectors \( \text{pr}(\alpha, v) \) by the Dirichlet PageRank vectors \( \text{pr}_S(\alpha, v) \).

From the definition of an \((k, h, \beta, \epsilon)-clusterable\) graph, each subset \( T \) of \( S \) has Cheeger ratio at least \( c\sqrt{h \log n} \). This allows us to use Lemma 3.6 for any segment subset \( T^f_j \) (as defined in (3.4)) with volume at most \((1 - \epsilon/2)\text{vol}(S)\) defined by the function \( f \) as in Lemma 3.6. Together we have that, for any subset \( R \subset S \) with \( \text{vol}(R) \leq (1 - \epsilon/2)\text{vol}(S) \),

\[
||[\text{pr}(\alpha, v)](R) - [\text{pr}(\alpha, \text{pr}(\alpha, v))](R)||_2 \leq \alpha t + \sqrt{n}e^{-(c^2 h \log n)/8} \\
\leq \frac{\epsilon}{4}
\]
by the assumption that \( c = 8\sqrt{\beta/k}/\epsilon \), and choosing \( t = 1/(hc^2) \). This implies that for any subset \( R \subset S \) and any vertex \( v \), we have
\[
||[pr(\alpha, v)](R) - [pr(\alpha, pr(\alpha, v))](R)||_2 \leq \frac{\epsilon}{2}.
\]
Thus the total variation distance between the two PageRank vectors is:
\[
\Delta_{TV}(\alpha) = \max_v \max_{R \subseteq S} \left| \left| pr(\alpha, v) - pr(\alpha, pr(\alpha, v)) \right| \right|_2 \leq \frac{\epsilon}{2}.
\]
Note that \( \sqrt{\Phi(\alpha)} \) is the \( \chi \)-squared distance \( \Delta_\chi \). Using the same technique as in [4], we have
\[
\Delta_{TV} \leq \Delta_\chi \leq \sqrt{1 - (1 - 2\Delta_{TV})^2}.
\]
Thus, we conclude that \( \Phi(\alpha) \leq \epsilon \) as desired.

We will also show that the sampling methods in PageRank-ClusteringA will ensure that with high probability, the cluster centers \( \{c_1, \ldots, c_k\} \) will include one from the core of each of \( k \) partitions in a clusterable graph:

**Lemma 3.9.** Suppose \( G \) is \((h, k, \beta, \epsilon)\)-clusterable, and \( c \log n \) sets of \( k \) potential centers are chosen from \( G \) according to the stationary distribution \( \pi \), where \( c \) is some absolute constant. With probability \( 1 - o(1) \), at least one set will contain one vertex from the core of each of the \( k \) clusters.

**Proof.** Let \( S_1, \ldots, S_k \) be a partition of \((h, k, \beta, \epsilon)\)-clusterable \( G \), and let \( S'_i \) be the core of \( S_i \). Suppose vertices \( C = \{c_1, \ldots, c_k\} \) are chosen randomly according to \( \pi \), and let \( E(C) \) be the event that each \( c_i \in S'_i \). Then, we have
\[
\Pr[E(C)] \geq \prod_{i=1}^k \Pr[c_i \in S'_i] = \prod_{i=1}^k \frac{\text{vol}(S'_i)}{\text{vol}(G)} \geq \prod_{i=1}^k \frac{(1 - \epsilon)\text{vol}(S'_i)}{\text{vol}(G)} \geq \prod_{i=1}^k \frac{(1 - \epsilon)\beta\text{vol}(G)}{k\text{vol}(G)} = \left( \frac{\beta(1 - \epsilon)}{k} \right)^k.
\]
If \( c \log n \) sets \( C_1, \ldots, C_{c \log n} \) of \( k \) centers are sampled independently, the probability that at least one contains each \( c_i \in S'_i \) is:

\[
\Pr[E(C_1) \lor \cdots \lor E(C_{c \log n})] \geq 1 - \prod_{i=1}^{c \log n} \Pr[\neg E(C_i)]
\]

\[
= 1 - \prod_{i=1}^{c \log n} (1 - \Pr[E(C_i)])
\]

\[
\geq 1 - \left( 1 - \left( \frac{\beta (1 - \epsilon)}{k} \right)^k \right)^{c \log n}
\]

\[
= 1 - o(1).
\]

This series of lemmas then leads to the proof of Theorem 3.1, showing the correctness of **PageRank-ClusteringA**.

**of Theorem 3.1.** We note that \( \text{pr}(0, s) = \pi \) and \( \text{pr}(1, s) = s \) for any distribution \( s \). This implies that \( \Phi(0) = \Phi(1) = \Psi(0) = 0 \) and \( \Psi(1) = n - 1 \). It is not hard to check that \( \Psi \) is an increasing function since \( \Psi'(\alpha) > 0 \) for \( \alpha \in (0, 1] \). The function of particular interest is \( \Phi \). Since we wish to find \( \alpha \) such that \( \Phi \) is small, it suffices to check the roots of \( \Phi' \) for an \( \alpha \) where \( \Phi(\alpha) < \epsilon \), which our algorithm does. Such an \( \alpha \) exists due to Lemma 3.8.

Suppose \( \alpha \) is a root of \( \Phi' \). To find \( k \) clusters, we can further restrict ourselves to the case of \( \Psi(\alpha) \geq k - 2 - \epsilon \) by Lemma 3.7.

We note that by sampling \( c \log n \) sets of \( k \) vertices from \( \pi \), for sufficiently large \( c \), the values \( \mu(C) \) and \( \Psi(C) \) for one such random set of \( k \) centers are close to \( \Phi(\alpha) \) and \( \Psi(\alpha) \), respectively, with high probability (exponentially decreasing depending on \( c \) and \( \beta \)) by probabilistic concentration arguments. In this context, the upper bound \( \epsilon \) for \( \mu(C) \) implies that the set consisting of distributions \( \text{pr}(\alpha, c) \) for \( c \in C \) serves well as the set of centers of mass. Thus, the resulting clusters \( R_c \) for \( c \in C \) give the desired clusters. This proves the correctness of our clustering algorithm with high probability for \( (k, h, \beta, \epsilon) \)-clusterable graphs. \( \square \)
3.5 Acknowledgement

Material in this chapter has appeared in the following two articles:


Chapter 4

Drawing Clustered Graphs Using PageRank

The visualization of complex graphs provides many computational challenges. Graphs such as the World Wide Web and social networks are known to exhibit ubiquitous structure, including power-law distributions, small-world phenomena, and a community structure [3, 23, 59]. With large graphs, it is easy for such intricate structures to be lost in the sheer quantity of the nodes and edges, which can result in drawings that reflect a network’s size but not necessarily its structure.

In this chapter, we give an algorithm PageRank-Display that uses PageRank vectors to draw graphs, highlighting the local structure within larger networks. Given a set of nodes $S$, we can extract communities around each vertex in $S$ and determine the layout of the graph using personalized PageRank. With this information, the vertex arrangement can be done using a force-based graph layout algorithm such as the Kamada-Kawai algorithm [80]. This leads to an organic layout, based on physical systems. There are many other algorithms specifically designed for clustered graph visualization [55, 114] and highlighting high-ranking vertices [21], but they impose a lot of artificial hierarchical structure onto the drawing and often require precomputing the clusters. The algorithm presented here only requires the $k$ cluster centers to be precomputed.
4.1 The graph drawing algorithm

PageRank-Display

Given \( k \) cluster centers, we can use personalized PageRank vectors to quantify relationships between vertices. For a cluster center \( c \), the components of the vector \( \text{pr}(\alpha, c) \) indicate how closely related other vertices are to \( c \). Using one vector for each of \( k \) centers, we have \( k \)-dimensional geometric data about the graph structure, which can be projected onto 2-dimensional space.

A cluster \( C \) can be characterized by its Cheeger ratio \( h(C) \); if \( h(C) \) is small, then there are not many connections from \( C \) to the rest of the graph, compared with the connections within \( C \) itself. Lemma 2.1 quantifies the relationship between the Cheeger ratio and personalized PageRank; if \( C \) has small Cheeger ratio, for many vertices \( v \in C \), the personalized PageRank vector \( \text{pr}(\alpha, v) \) has low mass outside of \( C \). This fact can be exploited to effectively display local structure when drawing larger networks. Furthermore, there are efficient approximation algorithms [8, 46] for computing PageRank vectors.

The goal is to capture local communities; we can do this by assigning edges \( \{s, v\} \) for each \( s \in S \) and \( v \in V \setminus S \) with weight inversely proportional to the personalized PageRank. Then, we use the Kamada-Kawai algorithm [80] to lay out the vertices, using an implementation from Graphviz [65]. The Kamada-Kawai algorithm simulates an \( n \)-body physical system where each edge is a spring characterized by its edge weight, positioning the vertices with the goal of minimizing the energy in the system. Thus, unrelated nodes with low PageRank will be forced to be distant, and close-knit communities will remain close together. We also add edges \( \{s, s'\} \) for \( s, s' \in S \) with large weight to encourage separation of the individual communities.

We note that because force-based algorithms are simulations, they do not guarantee the exact cluster structure, but we will illustrate that it works well in practice. Once we have a layout for all the nodes in the graph, we can partition them by using a Voronoi diagram. For a set of points \( S = \{s_1, \ldots, s_n\} \) in Euclidean space, the Voronoi diagram is a partition of the space into disjoint regions.
Let $R_1, \ldots, R_n$ such that each $R_i$ contains $s_i$ and the region of space containing the set of points that are closer to $s_i$ than any other $s_j$. Voronoi diagrams are well-studied in the field of computational geometry. Here we consider Voronoi diagrams on graphs using PageRank vectors as a notion of closeness. We compute the Voronoi diagram efficiently using Fortune’s algorithm [62].

The complete algorithm **PageRank-Display** appears below.

```
Algorithm **PageRank-Display**

Input: A graph $G = (V, E)$, a set of centers $S$, the jumping constant $\alpha$, and an approximation parameter $\epsilon$.
Output: A graph drawing image.

- For each $s \in S$, compute an approximate PageRank vector $p_s = pr(\alpha, s)$.
- Initialize a new weighted graph $G' = (V, E')$ with $E'$ initially empty.
- For each $s \in S$ and $v \in V \setminus S$, add $\{s, v\}$ to $E'$ with weight $1/p_s(v)$, as long as $p_s(v) > 0$.
- For each pair $s, s' \in S$, add $\{s, s'\}$ to $G'$ with weight $10 \times \max_{s,v} 1/p_s(v)$.
- Use a force-based display algorithm [80] on $G'$ to determine coordinates $c_v$ for each $v \in V$.
- Compute the Voronoi diagram on $S$ using [62].
- Draw $G$ using the coordinates $c_v$, highlighting $S$ with a different color, and overlaying the Voronoi diagram.
```

The jumping constant $\alpha$ is associated with the scale of the clustering. If $\alpha$ is too large, then some vertices may have extremely low PageRank and will be positioned very far away from the rest of the graph. On the other hand, if $\alpha$ is too small, the PageRank within a community according to Lemma 2.1 may be small, and the community structure will be lost. We can determine $\alpha$ either by trial and error or by optimizing the two graph metrics $\Phi$ and $\Psi$ given in the previous
chapter (Section 3.2). At the very least, for a community $C$, good values of $\alpha$ have $\alpha \gg h(C)$. This is only feasible if $h(C)$ is small, but if this is not the case, then $C$ does not demonstrate structural properties that make it a distinct cluster.

As long as $G$ is connected, the PageRank vector will be nonzero on every vertex. Using the algorithms from [8, 46], the approximation factor $\epsilon$ acts as a cutoff, and any vertex $v$ with PageRank less than $\epsilon d_v$ will be assigned zero. This is advantageous because the support of the approximate PageRank vector will be limited to the local community containing its seed. In PageRank-Display, we give weights to the edges equal to $1/p_s(v)$, but this is problematic if $p_s(v) = 0$. In that case, we omit the edge from $G'\mathord{'}$ entirely, as $s$ and $v$ are not part of the same cluster.

We remark that the selection of $\epsilon$ will influence the size of the local communities: the subset of nodes with nonzero approximate PageRank has volume at most $\frac{2}{(1-\alpha)e}$ (see [8]). This implies that a good selection of $\epsilon$ is $O\left(\frac{|S|}{(1-\alpha)\text{vol}(C)}\right)$.

We also remark that the selection of $S$ is important. If $S$ contains vertices that are not part of distinct local communities, then there will be no specific structure to display. Furthermore, if two vertices in $S$ are part of the same local community, then they will gravitate to form one larger cluster, but the Voronoi diagram will split the community into sections for each of the two vertices. The algorithm PageRank-Display is most effective when $S$ contains vertices that are part of distinct local communities. In general, the selection of $S$ is similar to the geometric problem of finding a set of points with minimum covering radius, which can be intractable (see [68]). There are several algorithms that can automatically choose $S$, including PageRank-Clustering as presented in Chapter 3.

### 4.2 Graph drawing illustrations

We used our algorithm to demonstrate and highlight the existence of local structure in several real-world datasets. The first dataset is a social network among 62 dolphins [99]. While the graph exhibits traditional network structure such as small-world phenomena, one can see in Figures 4.1 and 4.2 that the dolphins can be
divided into two communities, with just a few connected to both sides. Note that with larger $\alpha$, the far-flung nodes become more isolated, making the communities appear denser.

A more interesting example is shown in Figures 4.3 and 4.4. The vertices represent 114 NCAA Division I American collegiate football teams, with edges connecting two teams if they played against each other during the 2000 football season. The league is divided into many smaller conferences of up to 12 teams; for each team, about half of its games are played against conference opponents, and the rest are played against nonconference teams. An appropriate selection of the 8 highlighted teams in Figures 4.3 and 4.4 reveal a partition that separates their 8 respective conferences, and teams from the remaining conferences are placed on the periphery of the drawing. Here, the larger $\alpha$ is more effective as the PageRank is more concentrated near the community centers. Several more graph drawing examples are shown in Figures 4.5 through 4.7.

![Graph Drawing Example](image)

**Figure 4.1:** Results of PageRank-Display ($\alpha = 0.03$) on the dolphin social network [99]

### 4.3 Acknowledgement

Material in this chapter has appeared in the following two articles:

- Fan Chung and Alexander Tsiatas. “Finding and visualizing graph clusters
Figure 4.2: Results of PageRank-Display ($\alpha = 0.3$) on the dolphin social network [99]


Figure 4.3: Results of PageRank-Display ($\alpha = 0.1$) on the football game network [66]
Figure 4.4: Results of PageRank-Display ($\alpha = 0.3$) on the football game network [66]
Figure 4.5: Results of PageRank-Display on Zachary’s karate network [135]
Figure 4.6: Results of PageRank-Display on a network of political books about the 2004 US presidential election [92]
Figure 4.7: Results of PageRank-Display on a network of US Air Force flying teams [113]
Chapter 5

Spectral Analysis of Communication Networks with Dirichlet Eigenvalues

The study of large networks is an important field filled with many applications, important problems, and open questions. Amongst the key characteristics of large networks are bottlenecks, capacity, and reliability. These related properties are critical in the design and analysis of any data network: information must be able to flow freely from any part of the network to any other, and no small segment can be so critical as to undermine the network upon failure.

One well-studied technique that captures these structural properties is spectral graph theory, examining the eigenvalues and eigenvectors of graph-theoretic matrices (see [35] for a survey). Through relationships such as the Cheeger inequality, one can see that there are direct connections between the spectral gap and sparse cuts or bottlenecks in networks. When a network has a bottleneck, overall capacity and reliability suffer degradations.

Many real-world networks are incredibly vast, encompassing topologies with millions or billions of vertices and edges. Networks this large are often treated as infinite, as evidenced by network generation models [14, 58, 132] that exhibit unique convergence properties (to power-law degree distributions or otherwise) as the size of the network grows to infinity. Furthermore, with large-scale networks,
computation may be feasible only on smaller portions of the overall structure. In this case, it is advantageous to study these portions not as separate networks, but as subsets of infinite topologies.

To see how infinite networks provide good fundamental models for large-scale networks, one can take the tree as an example. Regular trees are usually thought to be excellent expanders due to exponential growth radiating from the root as the tree gets deeper. This is reflected in the spectrum of the infinite tree: it has a finite spectral gap [27, 53, 63]. But with finite truncations, it is not hard to see that the spectral gap tends to zero as the tree gets larger, masking the inherent expansion properties of trees. The same phenomenon exists when considering the isoperimetric or Cheeger constant.

In this chapter, we will use Dirichlet spectral clustering to identify bottlenecks in ten IP-layer communication networks as measured and documented by previous researchers in the Rocketfuel database [126]. Instead of using the normalized Laplacian matrix $L$; we will use the Dirichlet Laplacian $L_D$: the Laplacian restricted to the rows and columns corresponding to non-boundary vertices. Since we are interested primarily in finding one bottleneck at a time, we use the top two eigenvectors of $L_D$ in order to partition the network into two sections. We show empirically that the spectral gap using $L_D$ better characterizes bottlenecks at the core of these subnetworks.

The emphasis on identifying core bottlenecks becomes more critical in the light of the recent observation that many real-world graphs exhibit large-scale curvature [77, 110]. It has been shown [77, 110] that such global network curvature leads to core bottlenecks with load (or betweenness) asymptotically much worse than flat networks, where “load” means the the maximum total flow through a vertex assuming unit traffic between every vertex pair along shortest paths [110]. As such, it is important to find and characterize bottlenecks at the core rather than the fringes, where they do not matter as much.

The rest of this chapter is structured as follows: in Section 5.1, we give the theoretical justification for using Dirichlet eigenvalues [34] instead of the traditional spectrum for analyzing and clustering finite portions of infinite graphs. In
Section 5.2, we then compare the spectral gap using Dirichlet eigenvalues to the traditional spectral gap on real, publicly-determined network topologies [126] that represent smaller portions of the wider telecommunications grid. In Section 5.3, we demonstrate how Dirichlet spectral clustering finds graph partitions that are more indicative of bottlenecks in the network core rather than the fringes.

5.1 Convergence of the Dirichlet spectral gap in finite trees

Throughout this chapter, we analyze general undirected connected graphs $G$ by using the normalized Laplacian $\mathcal{L}$, defined as in [35] and in Chapter 2. We denote by $\lambda = \lambda(\mathcal{L})$ the spectral gap, the smallest nonzero eigenvalue of $\mathcal{L}$.

For the infinite $d$-regular tree, the spectral gap and Cheeger constant have both been analytically determined [63, 103]. Using $\mathcal{L}$, the spectral gap is

$$\lambda = 1 - \frac{2}{d} \sqrt{d - 1}, \quad (5.1)$$

and the Cheeger constant is $h = d - 2$ [71]. Both of these values are nonzero, indicating good expansion. However, the Cheeger ratio for truncated $d$-regular trees ($TdT$)—those with all branches of the infinite tree cut off beyond some radius $r$ from the center—approaches zero as the tree gets deeper. By cutting off any one subtree $S$ from the root, there is only one edge connecting $S$ to $\bar{S}$, and as the tree gets deeper, this ratio gets arbitrarily small. Using the Cheeger inequality, it follows that the $\lambda_{TdT} \to 0$ as $r \to \infty$. Thus, the standard spectral properties of finite trees do not approach the infinite case as they get larger; in fact, they suggest the opposite. This is problematic when making qualitative observations about networks and their expansion, necessitating another tool for spectral analysis of networks.

The main reason why the traditional spectral gap does not capture expansion well in large, finite trees is the existence of a boundary. This is also problematic in network partitioning algorithms; often times the “best” partition is a bag of whiskers or combination of several smaller cuts near the boundary [95].
In this chapter, we will use Dirichlet eigenvalues to eliminate this problem.

Dirichlet eigenvalues are the eigenvalues of a truncated matrix, eliminating the rows and columns that are associated with vertices on the graph boundary. We will use $\mathcal{L}_D$, which is distinct from simply taking the Laplacian of an induced subgraph, as the edges leading to the boundary vertices are still taken into account; it is only the boundary vertices themselves that are ignored. We define the \textit{Dirichlet spectral gap} to be the smallest eigenvalue of $\mathcal{L}_D$.

Using Dirichlet eigenvalues, there is also a \textit{local Cheeger inequality} [34] for a subgraph $S$. We define the \textit{local Cheeger ratio} for a set of vertices $T$ as in [34]:

$$H(T) = \frac{e(T, \bar{T})}{\text{vol}(T)},$$

and the \textit{local Cheeger constant} $h_S$ for a subgraph $S$ is the minimum local Cheeger ratio over all $T \subset S$. The local Cheeger inequality is: [34]

$$h_S \geq \lambda_S \geq \frac{h_S^2}{2},$$

where $\lambda_S$ is the Dirichlet eigenvalue of the Laplacian restricted to the rows and columns corresponding to vertices in $S$. This inequality indicates a relationship between local expansion and bottlenecks.

The use of Dirichlet eigenvalues requires that the boundary of the graph $S$ be defined. If $S$ is a tree, the leaf vertices are a natural choice. When $S$ is actually a finite truncation of a larger graph, the boundary can also be defined naturally; we shall see that this is the case for the Rocketfuel data [126].

We first use Dirichlet eigenvalues on $d$-regular trees as prototypical evidence for their effectiveness in capturing true spectral properties on real-world networks. There is empirical evidence in Figure 5.1, showing that the Dirichlet spectral gap for 3-regular trees indeed converges to a nonzero value as tree depth increases, contrasting with the traditional spectral gap which converges to zero. This is made rigorous in the following theorem:

**Theorem 5.1.** For finite $d$-regular trees of depth $L$, the Dirichlet spectral gap converges to the true spectral gap (5.1) of the infinite tree as $L$ approaches infinity.
Figure 5.1: Dirichlet spectral gap for successively larger 3-regular trees, showing convergence to a nonzero value

Proof. To derive the Dirichlet spectral gap for finite trees using the leaves as the boundary, we will solve a recurrence that arises from the tree structure and the standard eigenvalue equation

\[ \mathcal{L}_d x = \lambda x. \] (5.2)

Let \( T \) be a \( d \)-regular tree of depth \( L + 1 \); the \((L + 1)\)st level is the boundary. We will make the assumption that the eigenvector \( x \) has the same value at every vertex at the same depth within \( T \); these eigenvectors are azimuthally symmetric, and it can be shown that all other eigenvectors can be written as linear combinations of azimuthally symmetric eigenvectors with the same eigenvalue. Thus, we can represent each eigenvector \( x \) as a sequence of values \((x_0, x_1, \ldots, x_L)\), where \( x_i \) is the uniform value at all vertices at depth \( i \), similar to the analysis of the infinite-tree spectral gap appearing in [63]. Using this eigenvector form for \( x \) in (5.2) leads to the recurrence:

\[ x_i - \frac{1}{d} x_{i-1} - \frac{d-1}{d} x_{i+1} = \lambda x_i, \quad 2 \leq i \leq L. \] (5.3)
At the leaves of the tree, we have the Dirichlet boundary condition:

\[ x_{L+1} = 0. \]  

(5.4)

We can solve (5.3) using the characteristic equation:

\[ \frac{d-1}{d} r^2 - (1 - \lambda) r + \frac{1}{d} = 0, \]

whose roots can be written as

\[ r_{1,2} = \gamma e^{\pm i \alpha}. \]  

(5.5)

It follows that \( \gamma = \frac{1}{\sqrt{d-1}} \) and

\[ \lambda = 1 - \frac{2}{d} \sqrt{d-1} \cos \alpha. \]  

(5.6)

Substituting the the boundary condition (5.4) yields a solution to (5.3) with the form

\[ x_n = Ar_1^k (r_1^{n-k} - r_2^{n-k}), \]  

(5.7)

for some constant \( A \) and \( r_{1,2} \) given in (5.5). There are two ways that we can proceed.

One possibility is that the eigenvector has all \( x_i = 0 \). This could be the trivial case where the individual components of the eigenvector are all zero at every layer (which we ignore), but it could also be the case that the components are strictly non-zero until a certain layer. In this case, the individual components all become zero after that critical layer, and every layer still averages to zero, corresponding with the assumption that \( x_i = 0 \). Suppose that \( n^* \) is the first layer with components \( \neq 0 \). Then, using the boundary condition \( x_{n^*-1} = 0 \) and (5.7), we derive

\[ \alpha = \frac{m \pi}{L - n^* + 2}, 1 \leq n^* \leq L, 1 \leq m \leq L - n + 1. \]  

(5.8)

Alternatively, if there is no all-zero layer in \( x \), then we simply enforce (5.2) at the root to derive the second boundary condition \( x_0 - x_1 = \lambda x_0 \). Imposing this condition on (5.7) implies that the values for \( \alpha \) are the solutions to

\[ \frac{\tan \alpha}{\tan(L + 1) \alpha} = -\frac{1}{3}. \]  

(5.9)
It is not hard to see (via counting) that (5.8) and (5.9) furnish all the nontrivial eigenvalues and eigenvectors of (5.2). In any case, we are interested in finding the smallest eigenvalue \( \lambda \). This occurs when \( \cos \alpha \) is closest to 1, or \( \alpha \) is closest to zero, since there are no imaginary roots to (5.9). One can see that in both (5.8) and (5.9), as \( L \) gets larger, the smallest \( \alpha \) approaches 0, showing that (5.6) does indeed converge to the true spectral gap (5.1) of the infinite tree as the depth approaches infinity.

This derivation shows that Dirichlet eigenvalues capture the expansion properties of trees much better than the traditional spectral gap which has been shown to approach zero for large finite trees. This behavior on trees suggests that Dirichlet eigenvalues are a good candidate for use in analyzing real-world networks. Such analysis appears in Section 5.2.

\section{Comparison of spectral gaps in Rocketfuel networks and the 2-D grid}

Our research is motivated by a series of datasets representing portions of network topologies using Rocketfuel [126]. Rocketfuel datasets are publicly-available, created using traceroute and other networking tools to determine portions of network topology corresponding to individual Internet service providers. Even though like most measured datasets, the Rocketfuel networks are not free of errors (see, for example, [129]), they provide valuable connectivity information at the IP-layer of service provider networks across the globe.

Because the datasets were created in this manner, they represent only subsets of the vast Internet; it becomes impossible to determine network topology at certain points. For example, corporate intranets, home networks, other ISP’s, and network-address translation cannot be explored. The networks used range in size from 121 to 10,152 vertices.

Because of the method of data collection, the Rocketfuel datasets contain many degree-1 vertices that appear at the edge of the topology. In actuality, the
Table 5.1: Structural and spectral properties of Rocketfuel datasets

<table>
<thead>
<tr>
<th>Dataset ID</th>
<th>Vertices</th>
<th>Edges</th>
<th>Spectral gap</th>
<th>Dirichlet spectral gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>1221</td>
<td>2998</td>
<td>3806</td>
<td>0.00386</td>
<td>0.07616</td>
</tr>
<tr>
<td>1239</td>
<td>8341</td>
<td>14025</td>
<td>0.01593</td>
<td>0.03585</td>
</tr>
<tr>
<td>1755</td>
<td>605</td>
<td>1035</td>
<td>0.00896</td>
<td>0.09585</td>
</tr>
<tr>
<td>2914</td>
<td>7102</td>
<td>12291</td>
<td>0.00118</td>
<td>0.04621</td>
</tr>
<tr>
<td>3257</td>
<td>855</td>
<td>1173</td>
<td>0.01045</td>
<td>0.04738</td>
</tr>
<tr>
<td>3356</td>
<td>3447</td>
<td>9390</td>
<td>0.00449</td>
<td>0.05083</td>
</tr>
<tr>
<td>3967</td>
<td>895</td>
<td>2070</td>
<td>0.00799</td>
<td>0.03365</td>
</tr>
<tr>
<td>4755</td>
<td>121</td>
<td>228</td>
<td>0.03570</td>
<td>0.06300</td>
</tr>
<tr>
<td>6461</td>
<td>2720</td>
<td>3824</td>
<td>0.00639</td>
<td>0.11036</td>
</tr>
<tr>
<td>7018</td>
<td>10152</td>
<td>14319</td>
<td>0.00029</td>
<td>0.09531</td>
</tr>
</tbody>
</table>

network extends beyond this point, but the datasets are limited to one ISP at a time. As such, it makes sense to view these degree-1 vs as the boundary of a finite subset of a much larger network, similar to our treatment of the boundary of finite trees. This idea can be extended to general large-scale graphs: starting from one or more centers of mass, one can take the subset of the network within distance $r$ as a smaller, more manageable subset, for larger and larger $r$. (Finding centers of mass can be done in many ways; see [40] and Chapter 3 for one method.)

Following the discussion of Section 5.1, we compute the Dirichlet spectral gap of these graphs and compare with their standard counterparts, as shown in Table 5.1 and Figure 5.2. It is apparent that the Dirichlet spectral gaps are much larger than the traditional spectral gaps for all the networks, implying a much higher degree of expansion than one would traditionally obtain. By contrast, the Dirichlet and traditional spectral gaps for the two-dimensional square Euclidean grid are also shown; the grid is known to be a poor expander, and accordingly, even the Dirichlet spectral gap is very small.
5.3 Spectral clustering of Rocketfuel networks using Dirichlet eigenvectors

One important application of the eigendecomposition of a graph is spectral clustering or partitioning [112, 122]. The problem is to group the vertices into partitions, clusters, or communities that are inherently well-connected within themselves, with sparser connections between clusters. This is closely related to finding bottlenecks; if a graph has a bottleneck, then a good partition is often found by dividing the graph at the bottleneck.

It is often desirable for a network partition to be balanced, and finding bottlenecks near the core or center of mass of a network is often more useful than simply clipping small subsets of vertices near the boundary. But according to [95], using the Cheeger ratio as a metric on real-world data, the “best” cuts larger than a certain critical size are actually “bags of whiskers” or combinations of numerous smaller cuts. Because many graph clustering algorithms, including spectral clustering, try to optimize for this metric, the resulting partitions often slice numerous smaller cuts off the graph, which is not always useful. For our Rocketfuel data, we know that the boundary of the network is imposed by the method of data collec-
tion. Thus, by eliminating the boundary from graph clustering, we can more easily find partitions that are more evenly balanced, and bottlenecks that are closer to the core of the network.

To do this, we use standard spectral clustering techniques from [112], but instead of using the normalized graph Laplacian $L$, we use the truncated Dirichlet version $L_D$. The eigenvectors used for clustering will therefore not include components for the degree-1 boundary vertices, but we can assign them to the same side of the partition as their non-boundary neighbor vertices. Specifically, we compute the first two eigenvectors of $L_D$ and cluster the vertices based on their components in these eigenvectors using $k$-means. For each vertex, we compute the distance to both centers and sort the vertices based on the difference. For a partition of size $k$, we take the top $k$ vertices.

We follow the experiments of Leskovec et al. in [95] by using both traditional spectral clustering and Dirichlet spectral clustering to find cuts of different sizes. Specifically, we find Dirichlet cuts of all possible sizes, and then we find cuts using traditional spectral clustering for those same sizes after adding boundary vertices back in. Thus, for each network of $N$ vertices, we calculate $N - B$ cuts, where $B$ is the number of boundary vertices.

For each cut, we measure the Cheeger ratio $h$ and the number of components $c$. Ideally, a logical cut would split the network into exactly $c = 2$ components, but as Leskovec et al. demonstrated, as cut size increases, spectral clustering and other algorithms that optimize for $h$ yield cuts with many components. This is precisely the problem we are trying to avoid using Dirichlet clustering, and our results show that Dirichlet clustering is effective in finding cuts with fewer components. Furthermore, even though our algorithm is not specifically optimizing for $h$, it does not find cuts that have significantly worse values for $h$ while finding cuts with far fewer components.

We outline some aggregate data in Table 5.2. For several datasets, we count the number of cuts in four different categories, comparing the Dirichlet Cheeger ratio and number of components ($h_D$ and $c_D$) with traditional spectral clustering ($h_T$ and $c_T$). It is evident that Dirichlet clustering finds cuts with fewer components
Table 5.2: Aggregate data comparing Dirichlet spectral clustering with traditional spectral clustering for several Rocketfuel datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of cuts in each category:</th>
<th>Average $h_D - h_T$</th>
<th>Average $c_D - c_T$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$c_D \leq c_T$</td>
<td>$c_D \leq c_T$</td>
<td>$c_D &gt; c_T$</td>
</tr>
<tr>
<td>1221</td>
<td>89</td>
<td>362</td>
<td>5</td>
</tr>
<tr>
<td>1755</td>
<td>124</td>
<td>107</td>
<td>11</td>
</tr>
<tr>
<td>3257</td>
<td>121</td>
<td>75</td>
<td>37</td>
</tr>
<tr>
<td>3356</td>
<td>509</td>
<td>420</td>
<td>24</td>
</tr>
<tr>
<td>3967</td>
<td>98</td>
<td>245</td>
<td>13</td>
</tr>
<tr>
<td>4755</td>
<td>23</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>6461</td>
<td>350</td>
<td>415</td>
<td>0</td>
</tr>
</tbody>
</table>

than traditional spectral clustering ($c_D \leq c_T$) for most cut sizes, indicating that while spectral clustering optimizes for Cheeger ratio, it often “cheats” by collecting whiskers as one cut. In addition, despite the use of Cheeger ratio optimization, Dirichlet clustering sometimes finds cuts with better Cheeger ratio as well. In the last two columns for each dataset, we give the difference in $h$ and $c$ averaged out over all cut sizes. It turns out that the Cheeger ratios, on average, are not drastically different between the two methods, and Dirichlet clustering gives cuts with far fewer components.

Along with our aggregate data, we illustrate each individual cut for our Rocketfuel datasets in Figure 5.3. (A few of the datasets were too large for accurate numerical computation.) For each cut size, we plot a point corresponding to the difference in Cheeger ratio $h$ and the number of components $c$ between Dirichlet and traditional spectral clustering. It should be clear that for the majority of cut sizes, Dirichlet clustering finds cuts with far fewer components, but there is generally little change in Cheeger ratio. This can be seen in the large variation on the $c$-axis with much smaller discrepancies on the $h$-axis. In other words, Dirichlet clustering avoids finding “bags of whiskers” while still maintaining good separation in terms of $h$, despite not explicitly optimizing for $h$.

It is clear that using Dirichlet eigenvalues improves the partition by ignoring the boundary, alleviating the tendency to find “bags of whiskers” without
drastically changing the Cheeger ratio. Although traditional spectral clustering does not always fail, there is clear evidence that Dirichlet spectral properties are an important tool in the analysis of real-world networks.

The spectral decomposition using Dirichlet eigenvalues also suggests a further connection to large-scale negative curvature in the Rocketfuel data. Traditional negatively-curved graphs such as trees and hyperbolic grids generally exhibit poor connectivity and core congestion. Standard clustering often yields combinations of smaller cuts, but using Dirichlet clustering, we can see that there tend to be bad larger-scale cuts as well in the Rocketfuel datasets. The presence of these larger-scale cuts is a hallmark of negative curvature or hyperbolicity, suggesting the possibility of this property in the Rocketfuel data.

5.4 Acknowledgement

Material in this chapter appears in the following article, being prepared for submission:

Figure 5.3: Comparison of Cheeger ratio $h$ and number of components $c$ for cuts for various datasets using Dirichlet ($D$) and traditional ($T$) spectral clustering.
Chapter 6

Spectral Clustering of Graphs with General Degrees in the Extended Planted Partition Model

Spectral clustering of similarity graphs is a fundamental tool in exploratory data analysis, which has enjoyed much empirical success [100, 112, 122, 136] in machine-learning. Some theoretical work has been done [13, 100, 112], but many theoretical aspects, such as tight conditions for when spectral clustering works, and the effects of using different graph Laplacians, remain ill-understood. In this chapter, we examine a spectral clustering algorithm for similarity graphs drawn from a simple random graph model, where vertices are allowed to have varying degrees, and we provide theoretical bounds on its performance. Such clustering problems arise in the context of partitioning social network graphs to reveal hidden communities, or partitioning communication networks to reveal groups of vertices that frequently communicate. The model we study is the extended planted partition model, a variant and natural extension of the popular planted partition model.

The planted partition model has long been used as a benchmark for evaluating graph-partitioning algorithms; early work on partitioning random graphs
generated by this model include [18, 49, 76]. The state-of-the-art on the planted partition problem is due to [104]; he provides a spectral algorithm to recover the planted partitions by using a projection of the vertices onto the top $k$ eigenspace of the adjacency matrix. However, McSherry’s work [104] and those of his predecessors only address the case when all vertices in the same cluster have the same expected degree.

The extended planted partition (EPP) model, introduced in [51], allows for a different expected degree for each vertex, and as such can be viewed an extension of $G(\mathbf{w})$, the random graph model with given expected degrees [38], to the planted partition setting. A graph $G = (V, E)$ generated from this model has a hidden partition $V_1, \ldots, V_k$, as well as a number $d_u$ associated with each vertex $u$. If two vertices $u$ and $v$ lie in the same cluster $V_i$, then the edge $(u, v)$ is present in $G$ with probability $d_u d_v$; otherwise, it is present with probability $d_u q d_v$. Given a graph $G$ generated from this model, the Extended Planted Partition problem is to recover the hidden partition $V_1, \ldots, V_k$ without prior knowledge of the specific model parameters.

The standard approach to spectral clustering of graphs is to compute the bottom $k$ singular vectors or eigenvectors of a suitable graph Laplacian, project the vertices of the graph onto these vectors, and then use an iterative clustering algorithm on the projected vertices. This approach performs well when vertices from different clusters are well-separated when projected on to the bottom $k$ singular space or eigenspace of the appropriate graph Laplacian. For graphs drawn from the EPP model, projecting onto the bottom few eigenvectors of the unnormalized Laplacian does not work; indeed it is shown by [105] that high-degree vertices skew the top eigenspace of the adjacency matrix in the direction of the indicator vectors of those vertices. Thus, to cluster such graphs, we must counteract this effect with suitable degree normalization. However, if the minimum degree of any vertex in the graph is low, then the usual normalized Laplacians have poor concentration properties, and their bottom singular space or eigenspace may not correspond to a subspace where the clusters are well-separated.

Dasgupta et al. [51] provide a spectral algorithm for partitioning graphs
that it generates, under the assumption that the parameter vector $d$ that generates the graph is known by the algorithm. This assumption is critical to the algorithm, as its analysis depends on the concentration of the normalized adjacency matrix $D^{-1/2} \hat{A} D^{-1/2}$. When $D = \text{diag}(d)$ is unknown, one must normalize $\hat{A}$ by the actual degrees; however, the concentration of this matrix degrades with the minimum degree of any vertex in the graph.

In this chapter, we resolve the issue of poor concentration by introducing the notion of a degree-corrected normalized graph Laplacian. For a constant $\tau \geq 0$ and a graph with adjacency matrix $\hat{A}$, the degree-corrected random-walk Laplacian is the matrix $I - (\hat{T} + \tau I)^{-1} \hat{A}$, where $\hat{T}$ is the diagonal matrix of degrees. For $\tau = 0$, the degree-corrected Laplacian reduces to the regular Laplacian. If all the vertices in the graph have high degrees relative to $\tau$, then the bottom $k$ singular subspace of the degree-corrected random walk Laplacian is close to the bottom $k$ singular subspace of the random-walk Laplacian. However, if the graph has a number of low degree vertices, then degree-correction has a regularizing effect on the Laplacian.

Coja-Oghlan and Lanka [48] provide a spectral algorithm for solving the extended planted partition problem which does not require access to $d$, using the top $k$ eigenspace of $\hat{T}^{-1} \hat{A} \hat{T}^{-1}$, where $\hat{T}$ is the diagonal matrix of degrees; however, it only recovers the correct partition under certain conditions – when the maximum expected degree is of lower order than $n$, the cluster sizes are well-balanced, and the degree of each vertex is at least a constant fraction of the average degree. In contrast, our algorithm succeeds with more imbalance, and does not require these constraints. Moreover, even if there are many low-degree vertices, we can still use them in the degree-corrected Laplacian to find the correct subspace and help cluster the high-degree vertices, even if they cannot be clustered themselves.

Rohe et al. [118] consider the problem of partitioning graphs generated by the stochastic block model, which is the same as the planted partition model; they show that for graphs drawn from this model, the top $k$ eigenvectors of the normalized Laplacian are consistent, in the sense that they converge to a “population” limit as the number of vertices $n$ grows to infinity. They also provide guarantees
on the performance of a spectral clustering algorithm based on the normalized 
Laplacian. Unlike our work, they only consider graphs where the expected degrees 
of vertices in the same cluster are equal.

Bshouty and Long [26] provide a nearly linear-time algorithm for partition-
ing graphs generated by the planted partition model. Their algorithm has faster 
running time but requires a larger separation conditions. [84] use a variant of the 
extended planted partition model to probabilistically model graphs, and provide a 
local heuristic algorithm for estimating the parameters of this model. Theoretical 
properties of their algorithm are not studied rigorously.

Finally, spectral clustering of data drawn from a mixture model is well-
understood theoretically; see, for example, [1, 83, 93]. This work specifically deals 
with spectral clustering in similarity graphs. Some of the mathematical techniques 
used in our analysis are related to the work on learning mixture models; examples 
include [1, 11, 29, 83, 93]; however, the techniques for dealing with the effects of 
varying degrees are specific to this problem.

Our spectral clustering algorithm projects the vertices in the graph onto the 
bottom $k$ right singular vectors of the degree-corrected random-walk Laplacian, 
and clusters the vertices in this subspace. We show guarantees on the performance 
of this algorithm, demonstrating that it outputs the correct partition under a wide 
range of parameter values. Unlike [48], our algorithm can find the correct partition 
even if the cluster sizes are not well-balanced. Our analysis is also tighter than [51]; 
while our bounds may generally be worse, particularly when the graph has many 
low degree vertices, under certain conditions, we can show that our performance 
guarantees are better than those of [51]. Finally, even if the graph has very many 
low degree vertices, which cannot be reliably clustered because we simply do not 
have enough adjacent edges available, our algorithm can still use these vertices 
in the degree-corrected Laplacian to compute a subspace for clustering the high 
degree vertices reliably.

A key tool in our analysis is a sharp concentration bound on the spectral 
norm of the degree-corrected random-walk graph Laplacian, which approximately 
degrades with $\tilde{O}\left(\frac{1}{\sqrt{p}}\right)$. These bounds are then used to show that if the clusters are
well-separated, then, after projection onto the bottom $k$ right singular subspace of the degree-corrected random walk Laplacian, vertices from different clusters are well-separated while vertices from the same cluster are close together. A simple thresholding algorithm can then be used to recover the clusters correctly.

Finally, we provide some statistical lower bounds on the performance of any algorithm for finding planted partitions in graphs generated by the EPP model. Our bounds show that when the vertices have uniform degrees, the separation between the clusters required by our algorithm is within a factor of $\tilde{O}(\frac{1}{\sqrt{w_{\min}}})$ of the optimal separation; here $w_{\min}$ is the fraction of vertices that belong to the smallest cluster in the graph.

### 6.1 Planted partition models and graph Laplacians

In this section we give rigorous definitions for the planted partition models and graph Laplacians that are used throughout the rest of the chapter. The notation will be slightly different here than in the rest of the dissertation; this is due to different parametrization in the random graph model. In this chapter, the notation $d_u$ is reserved for the specific parameters of the extended planted partition model, and we will use $\deg(u)$ to represent the actual degree of the vertex $u$.

Since we are discussion random graphs, $G = (V, E)$ represents a specific graph drawn from the extended planted partition model. We use $\hat{A}$ to represent the adjacency matrix of $G$, and $A = E[\hat{A}]$. The diagonal matrix of degrees is $\hat{T}$, and $T = E[\hat{T}]$. In addition, we use $\hat{S}$ to denote the diagonal matrix $\hat{T} + \tau I$, where $\tau$ is a constant to be specified later, and $S$ to denote its expectation $E[\hat{S}]$. All expectations are taken over graphs generated from the extended planted partition model.

For much of the chapter, we work with a subgraph of $G = (V, E)$ induced by some subset $P$ of vertices. In this case, we use the subscript $P$ to denote the relevant quantities for this subgraph. For example, $\hat{A}_P$ denotes the adjacency matrix of the subgraph on $P$, $\deg_P(u)$ denotes the total number of edges between
a vertex $u$ and the vertices in $P$, and so on.

### 6.1.1 Planted partition models

**Planted Partition Model.** The planted partition (PP) model is a generative model for random graphs. A graph $G = (V, E)$ generated according to this model has a hidden partition $V_1, \ldots, V_k$ such that $V_1 \cup V_2 \cup \ldots V_k = V$, and $V_i \cap V_j = \emptyset$ for $i \neq j$. If a pair of vertices $u$ and $v$ both lie in some $V_i$, then, $\Pr[(u, v) \in E] = p$; otherwise $\Pr[(u, v) \in E] = q$. Thus, in the planted partition model, if $u$ and $v$ are two vertices in the same cluster, then their expected degrees are equal.

In the planted partition problem, we are given a graph $G$ generated by the planted partition model, and our goal is to find the hidden partition $V_1, \ldots, V_k$ with high probability over graphs generated according to this model.

**Extended Planted Partition Model.** The extended planted partition (EPP) model extends this model to graphs with non-uniform degree distributions. A graph $G = (V, E)$ generated according to this model again has a hidden partition $V_1 \cup \ldots \cup V_k = V$. In addition, each vertex $u$ is associated with a value $d_u$. If two vertices $u$ and $v$ lie in the same cluster $V_i$, then, $\Pr[(u, v) \in E] = d_u p d_v$; otherwise $\Pr[(u, v) \in E] = d_u q d_v$.

An extended planted partition model is characterized by parameters $(\mathcal{V}, \mathbf{d}, p, q)$, where $\mathcal{V} = \{V_1, \ldots, V_k\}$ is the hidden partition, $\mathbf{d}$ is the vector of $d_u$’s and $p$ and $q$ are values between 0 and 1. We note that the description of a particular model is not unique; for example, for any constant $c > 0$, the parameters $(\mathcal{V}, \mathbf{d}, p, q)$ and $(\mathcal{V}, c \mathbf{d}, p/c^2, q/c^2)$ describe the same EPP model.

In the extended planted partition problem, we are given a graph $G$ generated by an extended planted partition model, and our goal is to find the hidden partition $V_1, \ldots, V_k$ with high probability over graphs generated according to this model. Observe that unlike the work of [51], we do not have access to the $\mathbf{d}$ vector.

We use the notation $\bar{d}$ to denote the average of the $d_u$’s in the graph: $\bar{d} = \frac{1}{n} \sum_{u \in V} d_u$, and for a cluster $C_i$, we use the notation $\bar{d}_i$ to denote the average of the $d_u$’s among vertices in cluster $C_i$: $\bar{d}_i = \frac{1}{|C_i|} \sum_{u \in V} d_u$. For a cluster $C_i$, we
use $w_i$ to denote the fraction of vertices in the graph that belong to cluster $C_i$. We use: $w_{\min} = \min_i w_i$. Moreover, we use the notation $\mathbf{1}$ to denote the all-ones vector of length $n$, and $\mathbf{1}_j$ to denote the vector of size $n$ the $u$-th entry of which is 1 if vertex $u$ belongs to cluster $j$ and 0 otherwise.

### 6.1.2 Variants on the graph Laplacian

**Laplacian.** The Laplacian of a graph $G = (V,E)$ is defined as the matrix $\hat{L} = \hat{T} - \hat{\Delta}$.

**Random-walk Laplacian.** The random-walk Laplacian of a graph $G = (V,E)$ is defined as the matrix $\hat{\Delta} = I - \hat{T}^{-1} \hat{\Delta}$.

**Degree-corrected random-walk Laplacian.** The degree-corrected random-walk Laplacian of a graph $G = (V,E)$ is defined as the matrix: $\hat{\Delta}' = I - \hat{S}^{-1} \hat{\Delta}$. We use the notation $\Delta'$ to represent the matrix $I - S^{-1} A$.

**Normalized Laplacian.** The normalized Laplacian of a graph $G = (V,E)$ is defined in the usual way from Chapter 2, but in this chapter, we use the notation $\hat{\mathcal{L}} = I - \hat{T}^{-1/2} \hat{A} \hat{T}^{-1/2}$. The notation $\mathcal{L}$ is used to represent the matrix $I - T^{-1/2} AT^{-1/2}$.

**Degree-corrected Normalized Laplacian.** The degree-corrected normalized Laplacian of a graph $G = (V,E)$ is defined as the matrix: $\hat{\mathcal{L}}' = I - \hat{S}^{-1/2} \hat{A} \hat{S}^{-1/2}$.

### 6.2 The EPPCluster algorithm for recovering the hidden partition

We provide the algorithm **EPPCluster**, an algorithm for finding a planted partition in a graph $G = (V,E)$ drawn from an EPP model. To facilitate the analysis, we split $V$ randomly into two parts $P$ and $Q$; $Q$ is then projected on to the bottom $k$ right singular subspace of the degree-corrected random-walk Laplacian computed based on $P$, and partitioned in this subspace; the vertices in $P$ are partitioned analogously. This procedure preserves independence between the
subspace-computation and the graph-partitioning steps, thus making the analysis easier.

One can also consider a variant of Algorithm \textbf{EPPCluster} that uses the degree-corrected normalized Laplacian $I - \hat{S}^{-1/2} \hat{A} \hat{S}^{-1/2}$; however, our calculations show that we can get tighter bounds on the separation requirement between the clusters by using the degree-corrected random-walk Laplacian instead.

\begin{algorithm}
\textbf{Algorithm EPPCluster}
\begin{itemize}
    \item Input: a graph $G = (V,E)$ generated from an EPP model and the number of clusters $k$.
    \item Output: A partition of $V$.
    \begin{enumerate}
        \item Split $V$ randomly into two sets $P$ and $Q$, each of size $n/2$.
        \item Compute the degree-corrected random-walk Laplacian on $P$
        \[ \hat{\Delta}_P' = I - \hat{S}_P^{-1} \hat{A}_P, \]
        where for $u \in P$, $\hat{S}_P = (\hat{T}_P + \tau I)$, for some $\tau$ to be determined later.
        \item Compute a singular value decomposition of $\hat{\Delta}_P'$. Compute $\hat{U}_P$, the subspace spanned by the bottom $k$ right singular vectors of $\hat{\Delta}_P'$.
        \item For each vertex $u$ in $Q$, let $X_u$ be the row of the adjacency matrix corresponding to $u$ restricted to the vertices in $P$. Let $Y_u = P_{\hat{U}_P}(\frac{X_u}{\deg_P(u)})$, and define:
        \[ \lambda_u = \frac{9\sqrt{k \ln(6kn/\delta)}}{\sqrt{2(\deg_P(u) - 8\sqrt{\deg_P(u) \ln(6n/\delta)})}}. \]
        Initially, all $u$ in $Q$ are unlabeled.
        \item While there exists an unlabeled vertex in $Q$:
            \begin{enumerate}
                \item Let $u$ be an unlabeled vertex in $Q$ that maximizes $\deg_P(u)$. Create a new label $l$, and assign label $l$ to vertex $u$.
                \item For each unlabeled vertex $v$ in $Q$, if $||Y_u - Y_v|| \leq \lambda_u + \lambda_v$, assign $v$ the label $l$.
            \end{enumerate}
    \end{enumerate}
\end{algorithm}
6. Let $C_l$ be the set of vertices in $Q$ that are labeled $l$. Output clusters $C_1, C_2, \ldots, C_r$.

7. Repeat Steps (2)-(6) to cluster the vertices in $P$.

Observe that one difference between Algorithm EPPCluster and [104] is that we use the degree-corrected random-walk Laplacian in Step 2; the degree-correction acts as a regularization step for the random-walk Laplacian matrix.

A second difference is that we project the vectors $\frac{X_u}{\deg_P(u)}$ instead of $X_u$ onto the bottom $k$ right subspace of the degree-corrected random walk Laplacian computed based on $P$. Unlike the planted partition model, in EPP, if $u$ and $v$ are drawn from the same partition $V_i$, then the vectors $\mathbb{E}[X_u]$ and $\mathbb{E}[X_v]$ are no longer equal; instead we have $\frac{\mathbb{E}[X_u]}{\mathbb{E}[\deg(u)]} = \frac{\mathbb{E}[X_v]}{\mathbb{E}[\deg(v)]}$. Thus, to ensure that vertices from the same partition are close together after projection, it is necessary to normalize by the degree before projection.

To demonstrate some of the benefits of EPPCluster, we show a direct comparison on a constructed example in Figure 6.1. A graph $G$ was generated from an EPP model with the following parameters:

- 200 vertices, divided into two clusters $V_1$ and $V_2$ of 100 vertices each.
- $d_u = 1$ for all vertices $u \in V_1$ and 75 vertices $u \in V_2$, and $d_v = 0.8$ for the remaining 25 vertices $v \in V_2$.
- $p = 0.6, q = 0.4$.

For this constructed $G$, we attempt to recover the planted partition $\{V_1, V_2\}$ without knowledge of $d$, $p$, or $q$. We use two algorithms: first, we use a more traditional spectral clustering algorithm by modifying EPPCluster to use the adjacency matrix $\hat{A}$, and then we use EPPCluster as written with the degree-corrected random-walk Laplacian. We show the confusion matrix for each attempt; Figure 6.1a for the adjacency matrix and Figure 6.1b for the degree-corrected random-walk Laplacian. The $i, j$-th component of these $2 \times 2$ matrices represent the number
of vertices in cluster $i$ classified as part of cluster $j$, normalized so the rows sum to 1. A perfect classification has the identity matrix $I$ as the confusion matrix.

\[
\begin{array}{c}
\text{(a) } \hat{A} \\
\text{(b) } \Delta' \text{ with } \tau = 0 \\
\text{(c) } \Delta' \text{ with } \tau = 90
\end{array}
\]

**Figure 6.1:** Comparison of spectral partitioning of a graph generated from an EPP model, with three different matrices

In Figure 6.1a, it is apparent that using the adjacency matrix is much worse than using the degree-corrected random-walk Laplacian in Figure 6.1b. This is because the expected degrees are skewed by the non-uniform $d$. This example uses $\tau = 0$; with increased $\tau$, EPPCluster can perform even better (see Figure 6.1c with $\tau = 90$), but numerical instability for even larger values of $\tau$ make direct comparison difficult.

### 6.3 Analysis of EPPCluster

We now provide performance guarantees for Algorithm EPPCluster. We begin with some basic notation. For a cluster $V_i$, we define the quantity $Z_i$ as:

\[
Z_i = nq\bar{d} + nw_i(p - q)\bar{d}_i.
\]

Observe that for a vertex $u$ in $Q$ and cluster $V_i$:

\[
E[\deg(u)] = \sum_{v \in V_i} d_u pd_v + \sum_{v \notin V_i} d_u qd_v = d_u Z_i.
\]

We define the vector $\mu_i$ as:

\[
\mu_i = q1D + (p - q)1_j D.
\]
We use an additional subscript $P$ for these quantities restricted to a subset $P$ of vertices. The notation $\bar{d}_P$ represents the average $d_u$ for vertices $u \in P$, and $\bar{d}_{i,P}$ is the average $d_u$ for vertices $u \in P \cap V_i$. We also use the notation $Z_{i,P}$ and $\mu_{i,P}$ accordingly:

$$Z_{i,P} = nq\bar{d}_P + nw_{i,P}(p - q)\bar{d}_{i,P},$$
$$\mu_{i,P} = q1D_P + (p - q)1_jD_P.$$

We first analyze clustering the vertices in $Q$ using a projection onto the Laplacian computed based on the vertices in $P$. The analysis for the other case is analogous.

**Theorem 6.1.** Let $G = (V, E)$ be a random graph drawn from an EPP model. Suppose $V$ can be split into two parts $P$ and $Q$ such that for all $u$, $E[\text{deg}_P(u)] \geq \frac{32}{9}\ln(6n/\delta)$. If $u$ and $v$ are two vertices in $Q$, and if for all pairs of clusters $i$ and $j$,

$$\left| \frac{\mu_{i,P}}{Z_{i,P}} - \frac{\mu_{j,P}}{Z_{j,P}} \right| > \frac{6\sqrt{\ln(2n/\delta)}}{Z_{i,P}\sqrt{\tau + \min_{u \in P} E[\text{deg}_P(u)]}} \cdot \left( \sum_{u \in V_i \cap P} \frac{d^2_u}{(E[\text{deg}_P(u)] + \tau)^2} \right)^{-1/2}$$

$$+ \frac{6\sqrt{\ln(2n/\delta)}}{Z_{j,P}\sqrt{\tau + \min_{u \in P} E[\text{deg}_P(u)]}} \cdot \left( \sum_{u \in V_j \cap P} \frac{d^2_u}{(E[\text{deg}_P(u)] + \tau)^2} \right)^{-1/2}$$

$$+ 2 \cdot \left( \min_{u \in V_i \cap Q} \lambda_u + \min_{v \in V_j \cap Q} \lambda_v \right)$$

then, with probability $1 - 2\delta$, the following statements hold:

1. If $u$ and $v$ belong to the same cluster in the EPP model, then Step 5(b) of EPPCluster assigns them the same label.

2. If $u$ and $v$ belong to different clusters in the EPP model, then Step 5(b) of EPPCluster assigns them different labels.
Statements 1 and 2 of Theorem 6.1 imply that the clustering output by \textbf{EPPCluster} is a correct clustering of \( Q \). We observe that the term
\[
\frac{6\sqrt{\ln(2n/\delta)}}{Z_{i,P}\sqrt{\tau + \min_{u \in P} E[\deg_P(u)]}}
\]
decreases with increasing \( \tau \), while \( \left( \sum_{u \in V_i \cap P} \frac{d_u^2}{E[\deg_P(u)]+\tau} \right)^{-1/2} \) increases as \( \tau \) increases. The right hand side of the condition in Theorem 6.1 is thus optimized when both terms are balanced.

Suppose \( V \) contains a number of low degree vertices \( L \) with large values of \( \lambda_u \) such that the separation conditions in Theorem 6.1 are satisfied for \( V \setminus L \) but not for \( L \). Observe that we can still apply Step 5 of \textbf{EPPCluster} on \( P \setminus L \) and \( Q \setminus L \) to cluster them; the proof of Theorem 6.1 can be easily extended to show that this will yield the correct clustering. Furthermore, we can still use the vertices in \( L \cap P \) to compute the subspace \( \hat{U} \) onto which vertices from \( Q \setminus L \) can be projected and vice versa, even if we cannot actually cluster the vertices in \( L \) reliably.

Theorem 6.1, combined with Lemma 6.7 leads to our main theorems. Suppose that the \( d_u \)'s are all equal; then we have the following result.

**Theorem 6.2** (Main Theorem, uniform \( d \)). \( G = (V, E) \) be a random graph drawn from an extended planted partition model with all \( d_u \)'s equal to \( d \). Suppose \( G \) satisfies the conditions of Lemma 6.7, \( q \) is a constant, and \( 1 - w_i - w_j \) is at least a constant for all pairs of vertices \( i \) and \( j \). If \( \tau = 0 \), and if:
\[
(p - q) \geq c \cdot \left( \frac{\sqrt{q \ln(2n/\delta)}}{d w_{\min} \sqrt{n}} + \frac{\sqrt{k \ln(6kn/\delta)}}{d^2 \sqrt{n w_{\min}}} \right)
\]
where \( c \) is a fixed constant, then \textbf{EPPCluster} outputs correct clusterings of \( P \) and \( Q \).

The lower bound on \( p - q \) in Theorem 6.2 has two terms; the first term corresponding to recovering the correct subspace, and the second term corresponding to distance concentration. Our bound is better than the bound of [51] by a factor of \( \sqrt{k} \); we believe that this is an artifact of our analysis. Observe from Theorem 6.18 that this bound is worse than the statistical lower bound by a factor of \( \frac{1}{\sqrt{w_{\min}}} \).
Theorem 6.2 is a direct consequence of the following more general result:

**Theorem 6.3** (Main Theorem, general case). Let $G = (V, E)$ be a random graph drawn from an extended planted partition model which satisfies the conditions in Lemma 6.7. If, for all $u$, $\mathbb{E}[\deg(u)] \geq \frac{128}{9} \ln(6n/\delta)$, and if for all pairs of clusters $V_i$ and $V_j$,

$$
\left(\frac{p}{Z_i} - \frac{q}{Z_j}\right)^2 \sum_{u \in V_i} d_u^2 + \left(\frac{p}{Z_j} - \frac{q}{Z_i}\right)^2 \sum_{u \in V_j} d_u^2 \\
\geq 64 \left(\frac{384 \sqrt{\ln(2n/\delta)}}{Z_i \sqrt{\tau} + \min_{u \in V_i} \mathbb{E}[\deg(u)]} \left(\sum_{u \in V_i} \frac{d_u^2}{(\mathbb{E}[\deg(u)] + \tau)^2}\right)^{-1/2}\right. \\
+ \frac{384 \sqrt{\ln(2n/\delta)}}{Z_j \sqrt{\tau} + \min_{u \in V_j} \mathbb{E}[\deg(u)]} \left(\sum_{u \in V_j} \frac{d_u^2}{(\mathbb{E}[\deg(u)] + \tau)^2}\right)^{-1/2} \\
\left. + \min_{u \in V_i, v \in V_j} 2(\lambda_u + \lambda_v)\right)^2
$$

then, w.p. $\geq 1 - 6\delta$, EPPCluster outputs a correct clustering.

6.3.1 Main lemmas

The main ingredients in the proofs of our main theorems are four key lemmas.

**Lemma 6.4** (Concentration of $\|\hat{\Delta}'_P - \Delta'_P\|$). Let $G$ be a graph on $n$ vertices drawn from the extended planted partition model with parameters $(V, d, p, q)$. Suppose $\delta \in (0, 1)$ and for each vertex $u$, $\mathbb{E}[\deg_P(u)] \geq 6 \ln(2n/\delta)$. Then, with probability $\geq 1 - \delta$,

$$
\|\hat{\Delta}'_P - \Delta'_P\| \leq \frac{6 \sqrt{\ln(2n/\delta)}}{\sqrt{\tau + \min_{u \in P} \mathbb{E}[\deg(u)]}}.
$$

**Proof.** (Of Lemma 6.4) We prove this lemma using techniques from [39]; following [39], we can bound $\|\hat{\Delta}'_P - \Delta'_P\|$ as:

$$
\|\hat{\Delta}'_P - \Delta'_P\| \leq \|S^{-1}_P(\hat{A}_P - A_P)\| + \|S^{-1}_P\hat{A}_P - S^{-1}_P\hat{A}_P\|.
$$

(6.1)
To bound the first term in Equation (6.1), we observe:

$$S_p^{-1} (\hat{A}_P - A_P) = S_p^{-1/2} \cdot S_p^{-1/2} (\hat{A}_P - A_P) S_p^{-1/2} \cdot S_p^{1/2}.$$ 

We can now apply Lemma 6.14 to conclude:

$$\left\| \left| S_p^{-1} (\hat{A}_P - A_P) \right| \right\| \leq \left\| \left| S_p^{-1/2} (\hat{A}_P - A_P) S_p^{1/2} \right| \right\| .$$

which from Lemma 6.17 is at most $\sqrt{\frac{3\ln(2n/\delta)}{\tau + \min_u \mathbb{E}[\deg_P(u)]}}$ with probability $\geq 1 - \delta/2$.

Let $\hat{\mathcal{L}}_P = I - \hat{S}_P^{-1/2} \hat{A}_P \hat{S}_P^{-1/2}$. Then, $\hat{A}_P = \hat{S}_P^{1/2} (I - \hat{\mathcal{L}}_P) \hat{S}_P^{1/2}$. To bound the second term in Equation (6.1), we observe:

$$\hat{S}_P^{-1} \hat{A}_P - S_p^{-1} \hat{A}_P = \hat{S}_P^{-1/2} (I - \hat{\mathcal{L}}_P) \hat{S}_P^{1/2} - S_p^{-1} \hat{S}_P^{1/2} (I - \hat{\mathcal{L}}_P) \hat{S}_P^{1/2} = (\hat{S}_P^{-1/2} - S_p^{-1} \hat{S}_P^{1/2}) \cdot \hat{S}_P^{1/2} = \hat{S}_P^{-1/2} \cdot (I - \hat{S}_P^{1/2} S_p^{-1} \hat{S}_P^{1/2}) \cdot (I - \hat{\mathcal{L}}_P) \cdot \hat{S}_P^{1/2}.$$

Now we can again apply Lemma 6.14 to conclude that:

$$\left\| \left| \hat{S}_P^{-1} \hat{A}_P - S_p^{-1} \hat{A}_P \right| \right\| \leq \left\| \left| I - \hat{S}_P^{1/2} S_p^{-1} \hat{S}_P^{1/2} \right| \right\| \cdot \left\| \left| I - \hat{\mathcal{L}}_P \right| \right\| .$$

Recall that from Lemma 6.16, $\left\| \left| I - \hat{\mathcal{L}}_P \right| \right\| \leq 1$. The theorem follows by combining this fact with Lemma 6.15 with error bound $\delta/2$, noting that $|P| = n/2$. \hfill $\square$

**Lemma 6.5** (Distance Concentration). Let $u$ be a vertex in $Q$, and let $X_u$ be the subset of the row of the adjacency matrix $\hat{A}$ corresponding to vertex $u$ restricted to the vertices in $P$. If $\mathbb{E}[\deg_P(u)] \geq \frac{32}{n} \ln(6n/\delta)$, and if $U$ is any fixed $k$-dimensional subspace, then, with probability $\geq 1 - \delta$, for all $u$,

$$\left\| \mathbb{P}_U \left( \frac{X_u}{\deg_P(u)} \right) - \mathbb{P}_U \left( \frac{\mathbb{E}[X_u]}{\mathbb{E}[\deg_P(u)]} \right) \right\| < \frac{9 \sqrt{k \ln(6kn/\delta)}}{\sqrt{2\mathbb{E}[\deg_P(u)]}}.$$

**Proof.** (Of Lemma 6.5) We can write:

$$\left\| \mathbb{P}_U \left( \frac{X_u}{\deg_P(u)} \right) - \mathbb{P}_U \left( \frac{\mathbb{E}[X_u]}{\mathbb{E}[\deg_P(u)]} \right) \right\| \leq \left\| \mathbb{P}_U \left( \frac{X_u - \mathbb{E}[X_u]}{\mathbb{E}[\deg_P(u)]} \right) \right\| + \left\| \mathbb{P}_U \left( \frac{X_u}{\deg_P(u)} - \frac{X_u}{\mathbb{E}[\deg_P(u)]} \right) \right\| .$$
To bound the first term, we can use Lemma 6.12 with $X_u = X$ and $\delta' = \delta/6n$. This implies that w.p. $\geq 1 - \delta/3$,

$$\left\| \mathbf{P}_u \left( \frac{X_u - \mathbb{E}[X_u]}{\mathbb{E}[\text{deg}_P(u)]} \right) \right\| \leq \frac{1}{\mathbb{E}[\text{deg}_P(u)]} \sqrt{\frac{k \ln(6kn/\delta)}{2}}.$$

To bound the second term, we note that:

$$\left\| \mathbf{P}_u \left( \frac{X_u}{\text{deg}_P(u)} - \frac{X_u}{\mathbb{E}[\text{deg}_P(u)]} \right) \right\| \leq \left\| \frac{X_u}{\text{deg}_P(u)} - \frac{X_u}{\mathbb{E}[\text{deg}_P(u)]} \right\| \leq ||X_u|| \cdot \left( \frac{|\mathbb{E}[\text{deg}_P(u)] - \text{deg}_P(u)|}{\text{deg}_P(u) \mathbb{E}[\text{deg}_P(u)]} \right) \leq \sqrt{\text{deg}_P(u)} \cdot \left( \frac{|\mathbb{E}[\text{deg}_P(u)] - \text{deg}_P(u)|}{\text{deg}_P(u) \mathbb{E}[\text{deg}_P(u)]} \right).$$  \quad (6.2)

where the last step follows because as $X_u$ is a 0/1 vector, $||X_u|| = \sqrt{\text{deg}_P(u)}$. Now we can use the Chernoff bound (Lemma 6.9) with $\delta' = \delta/6n$ to conclude that w.p. $\geq 1 - \delta/3$,

$$|\text{deg}_P(u) - \mathbb{E}[\text{deg}_P(u)]| \leq \frac{1}{3} \ln(6n/\delta) + 2\sqrt{\mathbb{E}[\text{deg}_P(u)] \ln(6n/\delta)}$$

$$\leq 4\sqrt{\mathbb{E}[\text{deg}_P(u)] \ln(6n/\delta)}. \quad (6.3)$$

Combining with equation (6.2) and some algebra gives:

$$\left\| \mathbf{P}_u \left( \frac{X_u}{\text{deg}_P(u)} - \frac{X_u}{\mathbb{E}[\text{deg}_P(u)]} \right) \right\| \leq \frac{4\sqrt{\ln(6n/\delta)}}{\mathbb{E}[\text{deg}_P(u)] \sqrt{\text{deg}_P(u)}}.$$

Finally, we note that using the Chernoff bound in Lemma 6.10, with probability at least $1 - \delta/3$, for all $u$, $\text{deg}_P(u) \geq \frac{1}{k} \mathbb{E}[\text{deg}_P(u)]$ as long as $\mathbb{E}[\text{deg}_P(u)] \geq \frac{32}{9} \ln(6n/\delta)$. The lemma follows after more algebra and recognizing that $k \geq 2$, and it all happens with probability $1 - \delta$. \qed

**Lemma 6.6 (Subspace Concentration).** For all clusters $i$,

$$\left\| \mathbf{P}_{\mathcal{S}_P \setminus \hat{\mathcal{A}}_P} (\mu_i) - \mu_i \right\| \leq 2 \left\| \Delta_P - \hat{\Delta}_P \right\| \left( \sum_{u \in P \cap V_i} \frac{d_u^2}{(d_u Z_i + \tau)^2} \right)^{-1/2}.$$
Proof. (Of Lemma 6.6) Recall that row $u$ of the matrix $I - L_P = S_P^{-1}A_P$ is:

$$\frac{1}{\mathbb{E}[\text{deg}_P(u)]} \frac{d_u \mu_i, P}{\mu_i, P} = \frac{d_u}{d_u Z_i + \tau \mu_i, P}.$$  

From Lemma 6.13 applied to $(S_P^{-1}A_P)^\top$, there exists a column vector $\beta$ such that $(S_P^{-1}A_P)^\top \beta = \mu_i, P$. Moreover,

$$||\beta|| = \left( \sum_{u \in P \cap C_i} \frac{d_u^2}{(d_u Z_i + \tau)^2} \right)^{-1/2}. \quad (6.4)$$

Let $P_{\hat{S}_P^{-1}\hat{A}_P}$ be the operator that projects a column vector onto the row-space of $\hat{S}_P^{-1}\hat{A}_P$. We can write:

$$\left\| P_{\hat{S}_P^{-1}\hat{A}_P}(\mu_i, P) - \mu_i, P \right\| = \left\| (I - P_{\hat{S}_P^{-1}\hat{A}_P})(S_P^{-1}A_P)^\top \beta \right\| \leq \left\| (I - P_{\hat{S}_P^{-1}\hat{A}_P})(S_P^{-1}A_P)^\top \right\| \cdot ||\beta||. \quad \text{(6.5)}$$

Equation (6.4) provides a bound on $||\beta||$. To bound the other term, we can write:

$$\left\| (I - P_{\hat{S}_P^{-1}\hat{A}_P})(S_P^{-1}A_P)^\top \right\| \leq \left\| (I - P_{\hat{S}_P^{-1}\hat{A}_P})(\hat{S}_P^{-1}\hat{A}_P)^\top \right\| + \left\| (I - P_{\hat{S}_P^{-1}\hat{A}_P})(S_P^{-1}A_P)^\top - (\hat{S}_P^{-1}\hat{A}_P)^\top \right\|. \quad \text{(6.5)}$$

The second term in Equation (6.5) is at most $\left\| S_P^{-1}A_P - \hat{S}_P^{-1}\hat{A}_P \right\|$; this is because the transformation $I - P_{\hat{S}_P^{-1}\hat{A}_P}$ cannot increase norms. To bound the first term, observe that $P_{\hat{S}_P^{-1}\hat{A}_P}(\hat{S}_P^{-1}\hat{A}_P)^\top$ is the best rank $k$ approximation to $\hat{S}_P^{-1}\hat{A}_P$. As $S_P^{-1}A_P$ is rank $k$,

$$\left\| (\hat{S}_P^{-1}\hat{A}_P)^\top - P_{\hat{S}_P^{-1}\hat{A}_P}\hat{S}_P^{-1}\hat{A}_P \right\| \leq \left\| \hat{S}_P^{-1}\hat{A}_P - S_P^{-1}A_P \right\|.$$

The lemma now follows by simple algebra, recognizing that

$$\left\| \Delta_P - \hat{\Delta}_P \right\| = \left\| \hat{S}_P^{-1}\hat{A}_P - S_P^{-1}A_P \right\|. \quad \Box$$

Lemma 6.7. Let $G = (V, E)$ be a random graph drawn from an extended planted partition model with parameters $(\mathcal{V}, d, p, q)$ such that for any cluster $V_i$, 

1. \( w_i \geq \frac{8\ln(4k/\delta)}{n} \).

2. \( \sum_{u \in V_i} d_u \geq \frac{8}{3} \sqrt{\ln(4k/\delta)} \sqrt{\sum_{u \in V_i} d_u^2} \).

3. \( \sum_{u \in V_i} d_u^2 \geq \frac{8}{3} \sqrt{\ln(4k/\delta)} \sqrt{\sum_{u \in V_i} d_u^4} \).

4. For any \( \tau \),
   \[
   \sum_{u \in V_i} \frac{d_u^2}{(E[\deg(u)] + \tau)^2} \geq \frac{8}{3} \sqrt{\ln(4k/\delta)} \sqrt{\sum_{u \in V_i} \frac{d_u^4}{(E[\deg(u)] + \tau)^4}}.
   \]

Then, with probability \( \geq 1 - \delta \) over the splitting of the vertices in \( V \) into \( P \) and \( Q \), for all clusters \( V_i \),

1. \( w_{i,P} \geq \frac{2\ln(4k/\delta)}{n} \).

2. \( \sum_{u \in V_i \cap P} d_u \geq \frac{1}{8} \sum_{u \in V_i} d_u \).

3. \( \sum_{u \in V_i \cap P} d_u^2 \geq \frac{1}{8} \sum_{u \in V_i} d_u^2 \).

4. For any \( \tau \),
   \[
   \sum_{u \in V_i \cap P} \frac{d_u^2}{(E[\deg_P(u)] + \tau)^2} \geq \frac{1}{8} \sum_{u \in V_i} \frac{d_u^2}{(E[\deg(u)] + \tau)^2}.
   \]

Proof. (Of Lemma 6.7) The cluster weight \( w_{i,P} \) can be written as a sum independent 0/1 random variables:

\[
\sum_{u \in V_i} X_u,
\]

where \( X_u = 1 \) with probability \( w_i/2 \). Applying the Chernoff bound in Lemma 6.10 with \( \delta = \delta'/4K \) and taking the union bound over all \( V_i \) gives the result for \( w_{i,P} \) with probability \( \geq 1 - \delta/4 \).

For \( \sum_{u \in V_i \cap P} d_u \), note that it can be written as \( \sum_{u \in V_i} d_u X_u \), where \( X_u = 1 \) with probability 1/2. We again use the Chernoff bound in Lemma 6.10 and the union bound with \( \delta = \delta'/4k \) to get the result in part 2 with probability \( \geq 1 - \delta/4 \). Part 3 follows from the same argument, replacing \( d_u \) by \( d_u^2 \). Part 4 follows from the same argument, replacing \( d_u \) by \( \frac{d_u^2}{(E[\deg(u)] + \tau)^2} \), and the observation that \( E[\deg_P(u)] \leq E[\deg(u)] \). Finally, the lemma follows from an union bound over the four parts. \(\square\)

A similar statement also holds for \( Q \).
6.3.2 Proofs of the main theorems

Proof. (Of Theorem 6.3) For a set \( P \) of vertices, let \( E_P \) denote the event that the consequences of Lemma 6.7 hold true for \( P \). Under the assumption that the conditions Lemma 6.7 hold, \( E_P \) occurs with probability at least \( 1 - \delta \). For the rest of the proof, it is assumed that \( E_P \) occurs. We define \( \sigma_{u,v} = \lambda_u + \lambda_v \). Recall that

\[
\left| \frac{\mu_{i,P}}{Z_i,P} - \frac{\mu_{j,P}}{Z_j,P} \right|^2 \geq \sum_{v \in V_i \cap P} d^2_v \left( \frac{p}{Z_i,P} - \frac{q}{Z_j,P} \right)^2 + \sum_{v \in V_j \cap P} d^2_v \left( \frac{p}{Z_j,P} - \frac{q}{Z_i,P} \right)^2.
\]

Recall, \( Z_i,P \leq Z_i \) and \( Z_j,P \leq Z_j \). Conditioned on \( E_P \), from Lemma 6.7, part 2:

\[
pZ_j,P - qZ_i,P = \sum_{u \in V_j \cap P} d_u (p^2 - q^2) + \sum_{u \notin V_i \cup V_j, u \in P} d_u q(p - q) \geq \frac{1}{8} (pZ_j - qZ_i).
\]

Again conditioning on \( E_P \), combining the two above inequalities with Lemma 6.7, part 3, we can write:

\[
\left| \frac{\mu_{i,P}}{Z_i,P} - \frac{\mu_{j,P}}{Z_j,P} \right|^2 \geq \frac{1}{64} \left( \frac{p}{Z_i,P} - \frac{q}{Z_j,P} \right)^2 \sum_{u \in V_i} d^2_u + \left( \frac{p}{Z_j,P} - \frac{q}{Z_i,P} \right)^2 \sum_{u \in V_j} d^2_u.
\]

If the conditions of the theorem hold, then,

\[
\left| \frac{\mu_{i,P}}{Z_i,P} - \frac{\mu_{j,P}}{Z_j,P} \right| \geq \frac{384 \sqrt{\ln(2n/\delta)}}{Z_i \cdot \sqrt{\tau + \min_{u \in V_i} \mathbb{E}[\deg(u)]}} \cdot \left( \sum_{u \in V_i} d^2_u \left( \frac{d^2_u}{(\mathbb{E}[\deg(u)] + \tau)^2} \right) \right)^{-1/2} + 2\sigma_{u,v}
\]

\[
+ \frac{384 \sqrt{\ln(2n/\delta)}}{Z_j \cdot \sqrt{\tau + \min_{u \in V_j} \mathbb{E}[\deg(u)]}} \cdot \left( \sum_{u \in V_j} d^2_u \left( \frac{d^2_u}{(\mathbb{E}[\deg(u)] + \tau)^2} \right) \right)^{-1/2} + 2\sigma_{u,v}
\]

\[
\geq \frac{6 \sqrt{\ln(2n/\delta)}}{Z_i,P \cdot \sqrt{\tau + \min_{u \in V_i} \mathbb{E}[\deg(u)]}} \cdot \left( \sum_{u \in V_i \cap P} d^2_u \left( \frac{d^2_u}{(\mathbb{E}[\deg_P(u)] + \tau)^2} \right) \right)^{-1/2} + 2\sigma_{u,v}
\]

\[
+ \frac{6 \sqrt{\ln(2n/\delta)}}{Z_j,P \cdot \sqrt{\tau + \min_{u \in V_j} \mathbb{E}[\deg_P(u)]}} \cdot \left( \sum_{u \in V_j \cap P} d^2_u \left( \frac{d^2_u}{(\mathbb{E}[\deg_P(u)] + \tau)^2} \right) \right)^{-1/2} + 2\sigma_{u,v}.
\]
Here, the second inequality assumes that $E_P$ occurs, following from Lemma 6.7, part 4.

Again conditioning on $E_P$, Lemma 6.7, part 1, implies that if $E[\deg(u)] \geq \frac{128}{9} \ln(2n/\delta)$, then $E[\deg_P(u)] \geq \frac{32}{9} \ln(2n/\delta)$. Hence, the preconditions of Theorem 6.1 are satisfied. Let $C_Q$ be the event that $\text{EPPCluster}$ outputs a correct clustering of $Q$; conditioned on $E_P$, Theorem 6.1 implies that $C_Q$ occurs with probability at least $1 - 2\delta$.

We can define the analogous events $E_Q$ and $C_P$; note that $E_P$ and $E_Q$ are independent. Lemma 6.7 implies that $E_Q$ occurs with probability at least $1 - \delta$, and conditioned on $E_Q$, Theorem 6.1 occurs with probability at least $1 - 2\delta$. $C_P$ and $C_Q$ are independent, and $\text{EPPCluster}$ correctly clusters both $P$ and $Q$ if both events occur. The theorem follows.

**Proof.** (Of Theorem 6.1) For any pair of vertices $u$ and $v$ in $Q$, we define $\sigma_{u,v} = \lambda_u + \lambda_v$. Let $E$ be the event that (a) for all pairs $u$ and $v$ that lie in the same cluster $V_t$, $||Y_u - Y_v|| \leq \sigma_{u,v}$ and (b) for all pairs $u$ and $v$ that lie in different clusters, $||Y_u - Y_v|| > \sigma_{u,v}$. Lemma 6.11 shows that if the conditions of the theorem hold, then, $E$ happens with probability $\geq 1 - 2\delta$. We assume for the rest of the proof that $E$ happens.

We now show the theorem by induction over the iterations of the while loop in Step 5 of $\text{EPPCluster}$. The induction hypothesis we maintain is that iteration $t$ of Step 5 correctly identifies a partition $V_t$, and assigns all vertices in $V_t \cap Q$ (and no other vertices) the same label $t$. The base case is at the beginning when there are no labeled vertices, and hence the induction hypothesis holds trivially.

Suppose the induction hypothesis holds after iteration $t$ of Step 5. This means that $t$ clusters in the graph, say clusters $V_1, \ldots, V_t$ have been correctly identified. Suppose $u^*$ is the vertex selected in Step 5(a) of the next iteration of Step 5; then $u^*$ cannot belong to $V_1 \cup \ldots \cup V_t$. Without loss of generality, let $u^* \in V_{t+1}$. Conditioned on $E$, if any unlabeled vertex $v$ belongs to $V_{t+1}$, then $v$ is assigned label $t + 1$; if $v \notin V_{t+1}$, then $v$ is left unlabeled. Therefore, the cluster $V_{t+1}$ is also recovered correctly. The theorem follows.

**Proof.** (Of Theorem 6.2) Note that when all the $d_u$’s are equal to $d$, $Z_i = \Theta(ndq)$
and $E[\deg_P(u)] = \Theta(nd^2q)$. With $\tau = 0$,

$$\left(\frac{p}{Z_i} - \frac{q}{Z_j}\right)^2 \sum_{u \in V_i} d_u^2 = \Theta\left(\frac{(p - q)^2 w_i}{nq^2}\right),$$

$$\frac{384 \sqrt{\ln(2n/\delta)}}{Z_i \sqrt{\tau + \min_{u \in V_i} E[\deg(u)]}} \left(\sum_{u \in V_i} \frac{d_u^2}{(E[\deg(u)] + \tau)^2}\right)^{-1/2} = \Theta\left(\frac{\sqrt{\ln(2n/\delta)}}{nd \sqrt{qw_i}}\right),$$

$$\lambda_u = \Theta\left(\frac{\sqrt{k \ln(6kn/\delta)}}{nd^2 q}\right).$$

These asymptotic bounds are symmetric for the terms involving $V_j$. The theorem follows by applying Theorem 6.3.

\[\square\]

### 6.3.3 Concentration lemmas

**Lemma 6.8.** Let $X_1, \ldots, X_n$ be independent 0/1 random variables, and $X = \sum_{i=1}^n \alpha_i X_i$, with all $\alpha_i \in (0, 1)$. Let $||\alpha||^2 = \sum_{i=1}^n \alpha_i^2$. With probability at least $1 - 2\delta$,

$$|X - E[X]| \leq \sqrt{\frac{||\alpha||^2 \ln(1/\delta)}{2}}.$$

**Proof.** The proof follows from the standard Hoeffding bound (see [73]) for independent random variables $Y_i$, with each $Y_i \in [a_i, b_i]$ and $Y = \frac{1}{n} \sum_{i=1}^n Y_i$:

$$\Pr[|Y - E[Y]| \geq \lambda] \leq 2 \exp\left(\frac{-2\lambda^2 n^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

Note that if the $X_i$’s are independent, then $Y_i = \alpha_i X_i$ are as well, and based on this change of variable, $Y_i \in [0, \alpha_i]$ and $Y = \frac{1}{n} X$. Thus, we can write

$$\Pr[|X - E[X]| \geq \lambda] \leq 2 \exp\left(\frac{-2\lambda^2 n^2}{\sum_{i=1}^n \alpha_i^2}\right).$$

The lemma follows by solving for a $\lambda$ that makes this probability $\leq \delta$. \[\square\]

**Lemma 6.9.** Let $X_1, \ldots, X_n$ be independent 0/1 random variables, and $X = \sum_{i=1}^n \alpha_i X_i$, with all $\alpha_i \in (0, 1)$. Let $\nu = \sum_{i=1}^n \alpha_i^2 E[X_i]$. If $\nu \geq \frac{1}{18} \ln(1/\delta)$, then with probability at least $1 - 2\delta$,

$$|X - E[X]| \leq \frac{1}{3} \ln(1/\delta) + 2\sqrt{\nu \ln(1/\delta)}.$$
Proof. Using standard Chernoff bounds appearing in [38], we have the following:

\[
\Pr[X \leq E[X] - \lambda] \leq \exp \left( \frac{-\lambda^2}{2\nu} \right), \tag{6.6}
\]
\[
\Pr[X \geq E[X] + \lambda] \leq \exp \left( \frac{-\lambda^2}{2(\nu + \lambda/3)} \right). \tag{6.7}
\]

Setting (6.6) ≤ δ requires λ ≥ √2ν ln(1/δ). For the upper tail (6.7), we need:

\[
\lambda \geq \frac{1}{3} \ln(1/\delta) + \sqrt{\frac{1}{9}(\ln(1/\delta))^2 + 2\nu \ln(1/\delta)}.
\]

As long as ν ≥ \frac{1}{18} ln(1/δ), choosing λ ≥ \frac{1}{3} ln(1/δ) + 2√ν ln(1/δ) gives lower bounds of δ for each tail; the lemma follows. Note that if all α_i = 1, then this is a special case and ν = E[X].

\[\square\]

Lemma 6.10. Let X_1, ..., X_n be independent 0/1 random variables, and X = \sum_{i=1}^{n} \alpha_i X_i, with all \alpha_i > 0. Let ν = \sum_{i=1}^{n} \alpha_i^2 E[X_i]. If (E[X])^2 ≥ \frac{32\nu}{9} \ln(1/\delta), then with probability at least 1 − δ,

\[X \geq \frac{1}{4} E[X].\]

Proof. Follows directly from the standard Chernoff bound (6.6) appearing in [38].

\[\square\]

### 6.3.4 Other lemmas

**Lemma 6.11.** Let G = (V, E) be a random graph drawn from an EPP model. Suppose V can be split into two parts P and Q such that for all u, E[deg_P(u)] ≥ \frac{32}{9} ln(6n/δ). If u and v are two vertices in Q, and if for all pairs of clusters i and

\[\]
\[
\begin{align*}
\left| \frac{\mu_{i,P}}{Z_{i,P}} - \frac{\mu_{j,P}}{Z_{j,P}} \right| &> \frac{6\sqrt{\ln(2n/\delta)}}{Z_{i,P}\sqrt{\tau + \min_{u \in P} \mathbb{E}[\text{deg}_P(u)]}} \cdot \left( \sum_{u \in V_i \cap P} \frac{d_u^2}{(\mathbb{E}[\text{deg}_P(u)] + \tau)^2} \right)^{-1/2} \\
&+ \frac{6\sqrt{\ln(2n/\delta)}}{Z_{j,P}\sqrt{\tau + \min_{u \in P} \mathbb{E}[\text{deg}_P(u)]}} \cdot \left( \sum_{u \in V_j \cap P} \frac{d_u^2}{(\mathbb{E}[\text{deg}_P(u)] + \tau)^2} \right)^{-1/2} \\
&+ 2 \cdot \left( \min_{u \in V_i \cap Q} \lambda_u + \min_{v \in V_j \cap Q} \lambda_v \right),
\end{align*}
\]

then with probability \( 1 - 2\delta \), the following statements hold:

1. If \( u \) and \( v \) belong to the same cluster in the EPP model, then \( ||Y_u - Y_v|| \leq \lambda_u + \lambda_v \).

2. If \( u \) and \( v \) belong to different clusters in the EPP model, then \( ||Y_u - Y_v|| > \lambda_u + \lambda_v \).

\textbf{Proof.} (Of Lemma 6.11) For any pair of vertices \( u \) and \( v \) in \( Q \), we define \( \sigma_{u,v} = \lambda_u + \lambda_v \). Suppose \( u \) and \( v \) are two vertices in the same cluster \( V_i \) of the graph, and suppose that \( u \) and \( v \) lie in \( Q \). Then, recall that \( \frac{\mathbb{E}[X_u]}{\mathbb{E}[\text{deg}_P(u)]} = \frac{\mathbb{E}[X_v]}{\mathbb{E}[\text{deg}_P(v)]} = \frac{\mu_{i,P}}{Z_{i,P}} \), and therefore:

\[
||Y_u - Y_v|| \leq \left| Y_u - \mathbf{P}_{\breve{S}_P^{-1} \breve{A}_P} \left( \frac{\mathbb{E}[X_u]}{\mathbb{E}[\text{deg}_P(u)]} \right) \right| + \left| Y_v - \mathbf{P}_{\breve{S}_P^{-1} \breve{A}_P} \left( \frac{\mathbb{E}[X_v]}{\mathbb{E}[\text{deg}_P(v)]} \right) \right|
\]

Since the projection \( \mathbf{P}_{\breve{S}_P^{-1} \breve{A}_P} \) is computed independently of the edges adjacent to \( u \) and \( v \), we can apply Lemma 6.5 and (6.3) to conclude that \( ||Y_u - Y_v|| \) is at most \( \sigma_{u,v} \) for all \( u \) and \( v \) with probability \( \geq 1 - \delta \).

Now suppose \( u \) lies in cluster \( V_i \) and \( v \) lies in cluster \( V_j \), and both \( u \) and \( v \) lie in \( Q \). Then,

\[
||Y_u - Y_v||
\]

\[
\geq \left| \frac{\mu_{i,P}}{Z_{i,P}} - \frac{\mu_{j,P}}{Z_{j,P}} \right| - \left| \frac{\mu_{i,P}}{Z_{i,P}} - \mathbf{P}_{\breve{S}_P^{-1} \breve{A}_P} \left( \frac{\mu_{i,P}}{Z_{i,P}} \right) \right| - \left| \frac{\mu_{j,P}}{Z_{j,P}} - \mathbf{P}_{\breve{S}_P^{-1} \breve{A}_P} \left( \frac{\mu_{j,P}}{Z_{j,P}} \right) \right|
\]

\[
- \left| Y_u - \mathbf{P}_{\breve{S}_P^{-1} \breve{A}_P} \left( \frac{\mathbb{E}[X_u]}{\mathbb{E}[\text{deg}_P(u)]} \right) \right| - \left| Y_v - \mathbf{P}_{\breve{S}_P^{-1} \breve{A}_P} \left( \frac{\mathbb{E}[X_v]}{\mathbb{E}[\text{deg}_P(v)]} \right) \right|.
\]
Again using Lemma 6.5 and (6.3),
\[
||Y_u - Y_v|| \geq \left|\mu_{i,P} - \mu_{j,P}\right| - \frac{1}{Z_{i,P}} \left|\mu_{i,P} - P_{S_{P_{\hat{A}}}}(\mu_{i,P})\right|
\]
\[
- \frac{1}{Z_{j,P}} \left|\mu_{j,P} - P_{\hat{S}_{P_{\hat{A}}}}(\mu_{j,P})\right| - \sigma_{u,v}.
\]
Recall that from Lemmas 6.4 and 6.6,
\[
\left|\mu_{i,P} - P_{\hat{S}_{P_{\hat{A}}}}(\mu_{i,P})\right| \leq 6\sqrt{\ln(2n/\delta)},
\]
for all \(i\) with probability \(1 - \delta\); a similar statement is therefore true for \(\left|\mu_{j,P} - P_{\hat{S}_{P_{\hat{A}}}}(\mu_{j,P})\right|\). Thus, if the conditions of the theorem hold, then,
\[
||Y_u - Y_v|| > \sigma_{u,v}.
\]
The lemma follows.

**Lemma 6.12.** Let \(U\) be a \(k\)-dimensional subspace and \(X\) a 0/1 random vector. Then, w.p. \(1 - 2\delta'\),
\[
||P_U(X - E[X])|| \leq \sqrt{\frac{k \ln(k/\delta')}{2}}.
\]

**Proof.** Let \(\{v_1, \ldots, v_n\}\) be an orthonormal basis of \(U\). The projection onto \(U\) can be written as:
\[
P_U(X - E[X]) = \sum_{i=1}^{k} (\langle X, v_i \rangle - \langle E[X], v_i \rangle) v_i.
\]
Applying the Hoeffding bound (Lemma 6.8) to \(\langle X, v_i \rangle\) with \(\delta' = \delta/k\) and taking the union over all \(i\) gives that for all \(i\), with probability at least \(1 - \delta'\):
\[
|\langle X, v_i \rangle - \langle E[X], v_i \rangle| \leq \sqrt{\frac{\ln(k/\delta')}{2}},
\]
noting that ||\(\alpha||^2 = 1\) because \(v_i\) is a unit vector. Using the triangle inequality,
\[
||P_U(X - E[X])|| \leq \sqrt{\sum_{i=1}^{k} |\langle X, v_i \rangle - \langle E[X], v_i \rangle|^2},
\]
from which the lemma follows. 

\(\square\)
Lemma 6.13. Let $\mathbf{x} \in \mathbb{R}^n$, $\alpha \in \mathbb{R}^k$ be column vectors, and let $M$ be an $n \times k$ matrix such that the $i$th column of $M$ is $\alpha_i \mathbf{x}$. Then, there exists a column vector $\beta \in \mathbb{R}^k$ such that: $M \beta = \mathbf{x}$, and $||\beta|| = \frac{1}{||\alpha||}$.

Proof. Let $\beta$ be the following vector: $\beta_i = \frac{1}{||\alpha||^2} \alpha_i$, for $i \in \{1, \ldots, k\}$. Then, $||\beta|| = \frac{1}{||\alpha||}$. Moreover,

$$M \beta = \left( \sum_i \alpha_i \beta_i \right) \mathbf{x} = \sum_i \frac{\alpha_i^2}{||\alpha||^2} \mathbf{x} = \mathbf{x}$$

The lemma follows. □

Lemma 6.14. Let $M$ be a symmetric matrix, let $H$ be a diagonal matrix, and let $M' = H^{-1/2} M H^{1/2}$. If $\mathbf{x}$ is an eigenvector of $M$ with eigenvalue $\lambda$, then:

1. $xH^{-1/2}$ is a right eigenvector of $M'$ with eigenvalue $\lambda$.

Proof. Let $y = xH^{-1/2}$.

$$yM' = y H^{-1/2} M H^{1/2} = x M H^{-1/2} = \lambda x H^{-1/2} = \lambda y.$$ 

The first part of the lemma follows. To prove the second part, we observe that $y$ is also an eigenvector of $M'^T M'$ with eigenvalue $\lambda^2$. □

Lemma 6.15. Let $G = (V, E)$ be a random graph drawn from an EPP model, let $\hat{\mathbf{S}} = (\deg(u) + \tau)$ for $\tau > 0$, and let $S = \mathbb{E}[\hat{\mathbf{S}}]$. If for some $\delta' \in (0, 1)$ and all $u \in V$, $\mathbb{E}[\deg(u)] \geq \frac{1}{18} \ln(2n/\delta')$, then, w.p. $\geq 1 - \delta'$,

$$\left| I - \hat{\mathbf{S}}^{1/2} S^{-1} \hat{\mathbf{S}}^{1/2} \right| \leq \frac{4 \sqrt{\ln(2n/\delta')}}{\sqrt{\min_u \mathbb{E}[\deg(u)]^2 + \tau}}.$$ 

Proof. We use a Chernoff bound (Lemma 6.9) on $\deg(u)$ with $\delta = \delta'/2n$ and use the union bound over $V$; for all $u$, with probability $\geq 1 - \delta'$,

$$|\deg(u) - \mathbb{E}[\deg(u)]| \leq \frac{1}{3} \ln(2n/\delta') + 2 \sqrt{\mathbb{E}[\deg(u)] \ln(2n/\delta')}.$$ 

Therefore,
\[
\left\| I - \hat{S}^{1/2} S^{-1/2} \hat{S}^{1/2} \right\| = \max_u \left| 1 - \frac{\text{deg}(u) + \tau}{\text{E}[\text{deg}(u)] + \tau} \right|
\]
\[
= \max_u \left| \frac{\text{E}[\text{deg}(u)] - \text{deg}(u)}{\text{E}[\text{deg}(u)] + \tau} \right|
\]
\[
\leq \max_u \frac{\frac{1}{2} \ln(2n/\delta') + 2\sqrt{\text{E}[\text{deg}(u)]} \ln(2n/\delta')}{\text{E}[\text{deg}(u)] + \tau}
\]
\[
\leq \max_u \frac{4\sqrt{\text{E}[\text{deg}(u)]} \ln(2n/\delta')}{\text{E}[\text{deg}(u)] + \tau}
\]
\[
\leq \max_u \frac{4\sqrt{\ln(2n/\delta')}}{\sqrt{\text{E}[\text{deg}(u)] + \tau}},
\]

from which the lemma follows.

\[\Box\]

Lemma 6.16. Let \( \mathcal{L}_P' \) be the degree-corrected normalized Laplacian; then,

\[\left\| I - \mathcal{L}_P' \right\| \leq 1.\]

Proof. Recall that:

\[ I - \mathcal{L}_P' = \hat{S}^{-1/2} \hat{A}_P \hat{S}^{-1/2} = \hat{S}^{-1/2} \hat{T}_P^{-1/2} \cdot \hat{T}_P^{-1/2} \hat{A}_P \hat{T}_P^{-1/2} \cdot \hat{T}_P^{-1/2} \hat{S}^{-1/2}.\]

Therefore,

\[\left\| I - \mathcal{L}_P' \right\| \leq \left\| \hat{S}^{-1/2} \hat{T}_P^{-1/2} \right\| \cdot \left\| \hat{T}_P^{-1/2} \hat{A}_P \hat{T}_P^{-1/2} \right\| \cdot \left\| \hat{S}^{-1/2} \hat{T}_P^{-1/2} \right\|.\]

As \( I - \hat{T}_P^{-1/2} \hat{A}_P \hat{T}_P^{-1/2} \) is the actual normalized Laplacian of \( G \), its eigenvalues are in \([0, 2]\) (as seen in [35]), so \( \hat{T}_P^{-1/2} \hat{A}_P \hat{T}_P^{-1/2} \) has eigenvalues in \([-1, 1]\) and spectral norm \( \leq 1 \). Moreover, each entry of the diagonal matrix \( \hat{S}^{-1/2} \hat{T}_P^{-1/2} \) is at most 1 and therefore its spectral norm is also \( \leq 1 \). The lemma thus follows.

\[\Box\]

Lemma 6.17. If a random graph \( G \) is drawn from the EPP model, then, with probability \( \geq 1 - \delta \),

\[\left\| S^{-1/2} (\hat{A} - A) S^{-1/2} \right\| \leq \sqrt{\frac{3 \ln(2n/\delta)}{\tau + \min_u \text{E}[\text{deg}(u)]}}.\]
Proof. We heavily use tools from [39]. Following Theorem 2 of [39], we define $E^{uv}$ to be a matrix in which the $(u, v)$-th entry and the $(v, u)$-th entry is 1, and the rest of the entries are 0. Let $p_{uv}$ be the probability that the edge $(u, v)$ exists in the graph, and let $Z_{uv}$ be a 0/1 random variable which is 1 with probability $p_{uv}$ and 0 with probability $1 - p_{uv}$. Then, $\hat{A} = \sum_{u,v} p_{uv} Z_{uv}$.

Let $H_{uv} = \frac{Z_{uv} - p_{uv}}{S_{uu}S_{vv}} A_{uv}$. We observe that for any $u$ and $v$, $||H_{uv}|| \leq 1\sqrt{S_{uu}S_{vv}}$. Furthermore, for any $u \neq v$, $E(H_{uv}^2) = \frac{1}{S_{uu}S_{vv}} (p_{uv} - p_{uv}^2)(A_{uu} + A_{vv})$, and $E(H_{uu}^2) = 0$. Therefore,

$$\nu^2 = \left| \sum_{u,v} E(H_{uv}^2) \right| = \left| \sum_u \sum_v \frac{p_{uv} - p_{uv}^2}{S_{uu}S_{vv}} A_{uu} \right| = \max_u \left( \sum_v \frac{p_{uv} - p_{uv}^2}{S_{uu}S_{vv}} \right)$$

$$\leq \max_u \sum_v \frac{p_{uv}}{S_{uu}S_{vv}} \leq \frac{1}{\min_v S_{vv}} \sum_v \frac{p_{uv}}{S_{uu}} \leq \frac{1}{\min_v S_{vv}}.$$

Here the second to last step follows because $\sum_v p_{uv} = E[\deg(u)] \leq S_{uu}$. Similar to the proof of the first part of Theorem 2 of [39], the lemma now follows by an application of Theorem 5 of [39], with $M = 1$, $\nu^2 = \frac{1}{\min_v S_{vv}} = \frac{1}{\tau + \min_u E[\deg(u)]}$, and $a = \sqrt{\frac{3 \ln(2n/\delta)}{\tau + \min_u E[\deg(u)]}}$.

6.4 A statistical lower bound for the EPP problem

In this section, we show a lower bound required on the separation between clusters in the extended planted partition model for any algorithm to be able to correctly discover the clusters. This is a statistical lower bound, in the sense that it depends on statistical properties of the model, regardless of computational considerations.

Theorem 6.18. Let $G = (V, E)$ be a graph generated by the EPP model with $k = 3$ and parameters $(V, d, p, q)$. If $nv_{\text{min}}$ is the minimum size of any cluster in $G$, then, in order to correctly determine the cluster assignments of all vertices in
\( G \) w.p. \( \geq 3/4 \), we need:

\[
(p - q) \geq \frac{\sqrt{\ln 2}}{2d^2\sqrt{3nw_{\text{min}}}}
\]

Proof. (Of Theorem 6.18) Fix constants \( d > 0 \), \( 0 < \delta < \frac{1}{6d^2} \), \( w_{\text{min}} > 0 \), and let \( p = \frac{1}{2d^2} + \delta \), \( q = \frac{1}{2d^2} - \delta \), \( d = d1 \). Without loss of generality, let \( n = |V| \) and \( w_{\text{min}} \) be such that \( nw_{\text{min}} \) and \( n(1 - 2w_{\text{min}}) \) are integers, and let \( A \) be a fixed subset of \( n(1 - 2w_{\text{min}}) \) vertices. For a subset of vertices \( S \subset V \setminus A \) of size \( nw_{\text{min}} \), let \( \bar{S} = V \setminus (A \cup S) \). Thus, \( \{S, \bar{S}, A\} \) is a partition of \( V \), and since \( A \) is fixed, the partition is determined by the selection of \( S \). We denote by \( G_S \) the EPP with partitions given by a specific \( S \), and let \( F \) be the family of all such \( G_S \).

Suppose we are given \( G \) generated from some \( G_S \in F \), and we have an arbitrary algorithm or estimator \( \psi(G) \) for a specific member \( i \in F \). Then Fano’s inequality \([50]\) gives:

\[
\sup_{i \in F} \Pr[i \neq i] \geq 1 - \frac{\beta + \ln 2}{\ln r},
\]

where \( KL(G_S, G_{S'}) \leq \beta \) for all \( G_S, G_{S'} \in F \), and \( r = |F| - 1 \).

For a specific EPP \( G_S \in F \), the probability \( \Pr[G] \) of generating \( G \) is a product of independent Bernoulli distributions over the edges. Suppose that for a possible edge \( e \), the edge probability in \( G_S \) is \( \rho(e) \) and in \( G_{S'} \neq G_S \) is \( \rho'(e) \). From \([50]\), we write

\[
KL(G_S, G_{S'}) = \sum_e KL(\rho, \rho').
\]

For each possible edge \( e \), the KL-divergence is zero if \( \rho = \rho' \), and otherwise:

\[
KL(\rho, \rho') \leq d^2 p \ln \frac{d^2 p}{d^2 q} + (1 - d^2 p) \ln \frac{1 - d^2 p}{1 - d^2 q}
= (1/2 + d^2 \delta) \ln \frac{1/2 + d^2 \delta}{1/2 - d^2 \delta} + (1/2 - d^2 \delta) \ln \frac{1/2 - d^2 \delta}{1/2 + d^2 \delta}
= 2d^2 \delta \ln \frac{1/2 + d^2 \delta}{1/2 - d^2 \delta}
= 2d^2 \delta \ln \left(1 + \frac{2d^2 \delta}{1/2 - d^2 \delta}\right) \leq 2d^2 \delta \frac{2d^2 \delta}{1/2 - d^2 \delta}
\leq 2d^2 \delta \frac{2d^2 \delta}{1/3} \leq 6(d^2 \delta)^2 = \frac{3}{2} (d^2(p - q))^2.
\]
Let $N_e$ be the number of edges $e$ for which $\rho(e) \neq \rho'(e)$. Because $A$ is fixed, we only need to consider edges within $S \cup \bar{S}$:

$$\text{KL}(G_S, G_S') \leq \frac{3}{2} N_e (d^2(p - q))^2$$

$$\leq \frac{3}{2} \left(2nw_{\min}\right)(d^2(p - q))^2 \leq 3n^2w_{\min}^2(d^2(p - q))^2.$$

To bound $|F|$, we consider the number of ways to split $V \setminus A$ into $S$ and $\bar{S}$. For clarity, let $x = nw_{\min}$. Then

$$|F| = \frac{1}{2} \binom{2x}{x} \geq \frac{2^{2x-2}}{\sqrt{x}},$$

using Stirling’s approximation. Therefore,

$$\log(|F| - 1) \geq (2x - 3) \ln 2 - \frac{1}{2} \ln x \geq \frac{\ln 2}{2} x = \frac{\ln 2}{2} nw_{\min}.$$

Substituting into (6.8),

$$\sup_{i \in F} \Pr[\psi \neq i] \geq 1 - \frac{3n^2w_{\min}^2(d^2(p - q))^2}{\ln(2)nw_{\min}},$$

which means that w.p. $\geq 3/4$, there is no algorithm that can discern between $G_S$ and $G'_S$ if:

$$p - q \geq \sqrt{\frac{\ln 2}{2d^2/3nw_{\min}}}.$$

\[\square\]

### 6.5 Acknowledgement

Material in this chapter appears in the following article:

Chapter 7

Dirichlet PageRank and Ranking Algorithms Based on Trust and Distrust

PageRank has proven to be a useful tool for vertex ranking in many contexts, but some refinements are needed to address many increasingly complex but crucial problems. For example, PageRank is susceptible to manipulation by link spammers, and it treats all links between vertices as positive votes for importance even if some links are meant to show distrust.

To illustrate the need for incorporating several different types of “trust” and “distrust”, we consider the following four examples:

Problem 1. Suppose a smaller community holds an election. A community can be represented by a subgraph in a social network, and only edges incident to vertices in the subgraph can be used to determine the ranking of the vertices. The original interpretation of PageRank treats each edge as a vote for determining the “importance” of the vertices. A local community’s election should not be influenced by interests outside of the community; one way to deal with this is to set the influence of all outside vertices to be zero. This is the so-called Dirichlet boundary condition that we will discuss in this chapter.

Problem 2. In the WWW graph, there are many vertices whose importance is not
properly reflected by the link structure of the graph. For example, the websites of some governmental agencies are known to have high impact and authority, but they may not be highly connected to other websites. With prior knowledge of the network, it is often desirable to be able to effectively adjust the ranking of exceptional Web pages.

**Problem 3.** Another factor that many ranking models should address is the notion of “distrust”. Distrust can appear in many different ways; for example, if several vertices are known to be spammers, their neighbors are likely to be spammers as well. It is desirable to be able to quantify distrust which can then be used for protection as well as for penalizing spammers. In some cases, distrust between vertices can be built into the graph as negatively weighted edges, presenting computational challenges that the usual PageRank definition does not address. As we will see in later sections, we can use negative links to build a new network with boundary conditions chosen to appropriately propagate distrust.

**Problem 4.** Another vertex ranking problem arises from a distinction between types of social networks. Some networks, such as Facebook, model closer relationships between people: a social friendship where edges form presumably only between people who know each other personally. This is in direct contrast with systems such as Twitter and Google+ where the act of “following” does not necessarily indicate such a connection. Presumably, a person trusts his or her friends but is interested in “following” not just friends but many others, including celebrities, political figures, acquaintances, corporations, and even enemies.

When agents participate in both networks, it is useful to be able to rank the vertices of a larger network based on the smaller. A vertex $v$ can compute a ranking on the smaller, more close-knit network and then use this to calculate a ranking of vertices in the larger network that do not appear in the more personal network. Current ranking mechanisms such as PageRank compute a global ranking and therefore are not suitable for this situation.

In this chapter, we will show how Dirichlet PageRank can be used to model ranking problems, such as the above four, involving trust and distrust. We will
give an efficient algorithm to compute Dirichlet PageRank approximately, which leads to efficient algorithms to solve these problems.

7.0.1 Related Work

The idea of ranking vertices in a graph has a rich history starting from the introduction of PageRank by Brin and Page [22]. The original PageRank definition was designed for Web search, but many researchers have developed more tailored ranking systems such as personalized PageRank [72, 75] which gives a ranking relative to some specified starting distribution $s$.

One pitfall with PageRank as a ranking system is the fact that all edges contribute positively. In practice, an edge such as a link from one Web page to another can also represent a negative interaction or distrust between the vertices. Several related mathematical models of propagating trust and distrust in a network ranking system are given in [67], and there are numerous empirical results. Another algorithm [69] relies on a small hand-picked set of trusted vertices, but one must be careful not to allow malicious vertices to be included.

There are many other algorithms derived from PageRank that use specific heuristics to model trust or distrust in ranking schemes. [6] considers axioms that a ranking system should satisfy and develops several ranking systems accordingly. [20] and [88] systematically model distrust by modifying the PageRank equations to consider negatively-weighted edges, and [81] gives an algorithm with a similar flavor using random walks. Many of these algorithms are closely related, but rigorous analysis is desired for capturing specific phenomena. We will show that these related models can be represented by Dirichlet PageRank with appropriate boundary conditions.

Another area of research concerns spam vertices when they are identified. It has been shown that if agents can collude [12] or easily create pseudonyms [31], they can artificially boost their ranking in PageRank and other ranking systems. There has been some work done in how to effectively penalize these vertices [15], and our Dirichlet PageRank can be efficiently used to achieve the same goal.
7.0.2 Results in this Chapter

Motivated by the continual development of new PageRank-based algorithms and the analysis of Dirichlet eigenvectors in [45], we develop and analyze Dirichlet PageRank vectors as well as an efficient algorithm to compute them. For a connected graph $G$, we give a Dirichlet PageRank equation and show how to compute the unique solution with Dirichlet boundary conditions: $pr(\alpha, s)_v = 0$ for vertices $v$ on the boundary of a specified vertex subset $S$.

After giving the algorithm for computing Dirichlet PageRank vectors, we generalize the boundary conditions to arbitrary values $pr(\alpha, s)_v = \sigma(v)$ for boundary vertices $v$. We will give an efficient algorithm $\text{ApproxDirichPR}$ to compute approximate Dirichlet PageRank vectors with any boundary condition $\sigma$. We also give a full analysis leading to the following theorem. Detailed definitions will be given in Section 7.1.

We illustrate several applications of Dirichlet PageRank with boundary conditions below. Many of the specific PageRank variations are covered by this general framework, and we will show how its use can allow the efficient consideration of several models in [6, 15, 20].

7.0.3 Several applications of Dirichlet PageRank

Allowing negative edges in the graph While trust between two vertices is denoted by a positive weight, it is natural to quantify distrust as negative weights for the associated edges. There are many other types of relations in a network that can be represented with negative edges as well, and the usual PageRank vectors do not consider negative weights. We will use Dirichlet PageRank as a tool to deal with graphs containing negative edges in Section 7.4.1.

Diminishing known spammers’ influence Many Web pages can be identified as spammers based on content or user reports. It is desirable to have a network ranking scheme that takes such considerations into account by penalizing both the known spammer vertices and others with many links to them.
We will show that Dirichlet PageRank is useful for dealing with spammer vertices in Section 7.4.2.

Considering trusted friends’ opinions A single vertex in a graph may have a set of trusted friends or neighbors whose opinions need to be considered strongly in designing a vertex ranking scheme. If these trusted vertices have their own independent ranking opinions, we can use Dirichlet PageRank with appropriate boundary conditions to compute a trust-based ranking. We will give an algorithm \textbf{PRTrustedFriends} for this problem in Section 7.4.3.

Validating ranking for newly-created vertices Suppose that a new person enters a social network but is unsure about which vertices are trustworthy. Personalized PageRank is a useful tool for deriving quantitative information, but it raises the question of whether this ranking is susceptible to unknown spammers. Without a specific set of trusted friends, it may seem hopeless for the newcomer, but Dirichlet PageRank with boundary conditions can be used to validate and adjust its ranking with a randomly-selected pool of established vertices within the network. We will give the details for an algorithm \textbf{PRValidation} in Section 7.4.3.

Reconciling rank in personal and global social networks Several interesting questions arise when analyzing different types of social networks. Some, like Facebook, offer a more personal viewpoint as reflected in the network structure, where edges are formed only by mutual consent between two people who usually know each other. This is in contrast with a network such as Twitter, where the connections are often (though not always) impersonal. For example, people “follow” each other based not only on friendship, but also on subject matters, celebrity appeal, advertising, and many other conceivable reasons.

With the vast array of information available on a network such as Twitter, it is important for a user to know who is trustworthy or worth following. This is a difficult problem, but a user does have some information at hand, such as its own, more close-knit, smaller social networks or even a trusted sub-
graph of the larger network. Using Dirichlet PageRank, a user can compute
a ranking on the smaller network, and then use boundary conditions appro-
priately to infer a ranking on the remaining vertices in the larger network,
taking its personal associations into account. We will develop an algorithm
\textbf{PRTrustNetwork} for this problem in Section 7.4.3.

Finally, we can use similar ideas to tackle the problem in the reverse direction:
Suppose a global ranking of the vertices in a larger, loose social network such
as Twitter is known, and a user wants to develop a personalized ranking for
a small subgraph or its own trusted network taking the global ranking into
account. We again can use Dirichlet PageRank with appropriate boundary
conditions, as outlined in an algorithm \textbf{PRInferRanking}, also in Section
7.4.3.

The rest of the chapter proceeds as follows. Section 7.1 develops the the-
ory of PageRank with Dirichlet boundary conditions, and Section 7.2 extends this
theory for arbitrary boundary conditions $\sigma$. We develop and analyze the \textbf{Ap-
proxDirichPR} algorithm in Section 7.3 and give algorithms for the previously-
discussed applications in Section 7.4.

\section{PageRank with Dirichlet boundary

conditions}

Throughout this chapter, we will use the standard graph-theoretic termi-
nology as outlined in Chapter 2, including the notation $M_S$ to represent a matrix
restricted to the rows and columns corresponding to vertices in a set $S \subseteq V$. We
will exclusively use the lazy random walk matrix $Z$, the PageRank definition we
use here corresponds to the traditional definition from Equation (2.1) with $W$ re-
placed by $Z$. As mentioned in Section 2.1, the two definitions are equivalent up to
a change in the jumping constant $\alpha$.

The essential question proposed in Problem 1 is how to take into account
the boundary edges. We remark that in the classical areas of differential equa-
tions defined on some geometric spaces, the boundary conditions are referred to
the constraints defined on the boundaries of specified regions and are imposed on
the solutions of the equations. For the PageRank equation, the lazy random walk
transition matrix $Z$ is closely related to the normalized Laplacian $L$ which is the
analog of the Laplace-Bertromi operator in differential geometry. Thus, the Page-
Rank equation (2.1) can be regarded as a discrete analog to a set of differential
equations, and Dirichlet PageRank can be viewed as a solution of the PageRank
equation with boundary conditions. The basic problem of deriving PageRank vec-
tors with Dirichlet boundary conditions was also previously examined in [33].

Let $S$ be a subset of $G$; for a function (or vector) $f : V \rightarrow \mathbb{R}$, we say $f$
satisfies the Dirichlet boundary condition if

$$f(v) = 0 \text{ for all } v \in \delta S.$$  

The PageRank vector satisfying the Dirichlet boundary condition is the
solution for the following equation, for all vertices $v$:

$$pr(\alpha, s)_v = \begin{cases} 
\alpha s_v + (1 - \alpha) \sum_{u \in V} pr(\alpha, s)_u W_{uv} & \text{if } v \in S \\
0 & \text{otherwise.} 
\end{cases} \quad (7.1)$$

Let $pr_S(\alpha, s)$ and $s_S$ denote the vectors $pr(\alpha, s)$ and $s$ restricted to $S$, re-
spectively. The restricted Green’s function $G_{S,\beta}$ is defined by:

$$G_{S,\beta}(\beta I_S + L_S) = I_S,$$

where $\beta \geq 0$. Note that $L_S$ is positive definite [45], so $G_{S,\beta}$ is well-defined.

**Theorem 7.1.** For a connected graph $G$, vector $s$, and $\alpha > 0$, the above PageRank
equation (7.1) has one and only one solution. With $\beta = \frac{2\alpha}{1-\alpha}$, it is given by

$$pr_S(\alpha, s) = \beta s_S D_S^{-1/2} G_{S,\beta} D_S^{1/2}.$$  

**Proof.** Since $pr(\alpha, s)_v = 0$ when $v \notin S$, the PageRank equation (7.1) is equivalent to

$$pr_S(\alpha, s) = \alpha s_S + (1 - \alpha) pr_S(\alpha, s) Z_S.$$
Since
\[ Z = \frac{1}{2} (I + D^{-1} A) = I - \frac{1}{2} D^{-1/2} L D^{1/2}, \]
and \( D \) is diagonal matrix, we have
\[ Z_S = I_S - \frac{1}{2} D_S^{-1/2} L_S D_S^{1/2}. \]
Thus, we have
\[
pr_S(\alpha, s) = \alpha s_S + (1 - \alpha) pr_S(\alpha, s) \left( I_S - \frac{1}{2} D_S^{-1/2} L_S D_S^{1/2} \right).
\]
Solving for \( pr_S(\alpha, s) \) yields the theorem; uniqueness follows from the uniqueness of \( G_{S, \beta} \).

Using Dirichlet PageRank vectors with Dirichlet boundary conditions, we can solve Problem 1 in a straightforward way, and it is clear that vertices outside of \( S \) will not influence the ranking. However, it is not immediately apparent as to how the boundary edges affect the ranking result. In Figure 7.1, comparisons are given between rankings obtained by three methods. For a small graph, we compute the Dirichlet PageRank vector and compare it with the following two alternative methods:

**Method 1** Compute the PageRank for the entire graph and simply use these values on the subgraph.

**Method 2** Delete the rest of the graph including boundary edges, then compute the PageRank for the remaining induced subgraph.

We note that the difference between Method 1 and the Dirichlet PageRank is the relative ranking of vertices \( b \) and \( d \). Using Method 1, \( b \) and \( d \) have the same ranking, while with the Dirichlet PageRank, \( b \) has a higher ranking. This is consistent with the fact that in this subgraph, \( b \) is trusted by all other vertices, while \( d \) is trusted by only two out of three other vertices. Note that \( d \) is also trusted by another vertex outside of the subgraph, but the influence of the outside vertex is not taken as significantly.
Figure 7.1: Comparison of three vertex rankings on small graph

The difference between Method 2 and Dirichlet PageRank is the relative ranking of vertices $c$ and $d$. By Method 2, $c$ and $d$ have the same ranking; by the Dirichlet PageRank, $d$ has a slightly higher ranking. This is consistent with the fact that $c$ and $d$ trust each other and are trusted by $b$. In addition, $d$ is trusted by one vertex from outside, reflected by a higher ranking of $d$.

We give another Dirichlet PageRank example in Figure 7.2. In Figure 7.2a, a social network [99] have vertices colored according to their PageRank (for the case of $\alpha = 0.1$). Now, suppose we have identified two spammers and want to penalize their ranking and influence, as described in Problem 3. In Figure 7.2b, we simply compute the usual PageRank and set the two spammers' rank to zero. This is equivalent to the ranking algorithm proposed in [69]. In Figure 7.2c, we compute the Dirichlet PageRank with the boundary condition $\sigma(u) = \sigma(v) = 0$ for the spammers $u$ and $v$. It is apparent that the rankings of the vertices surrounding the spammers have been decreased, illustrating the effects of propagation of distrust.

These two examples serve as an illustration of the contributions of the boundary edges to the ranking, made more rigorous in the following lemma.
Let $pr'(\alpha, s)$ be the PageRank vector computed by Method 1 and let $pr''(\alpha, s)$ denote the PageRank computed by Method 2. It is easy to see that for every $v \in S$, $pr(\alpha, s)_v \leq pr'(\alpha, s)_v$.

We define two vertex sets $S_o$ and $S_i$:

$$S_o = \{v \in S \mid \exists u \notin S : (u, v) \in E\} \text{ and } S_i = S \setminus S_o.$$

Let $Z_{ii}$ and $Z_{oi}$ denote $Z$ restricted to $S_i \times S_i$ and $S_o \times S_i$, respectively. Similarly, we define two vectors $w_o$ and $w_i$:

$$w_o = (1 - \alpha)1Z_{oi}^T \text{ and } w_i = 1 - (1 - \alpha)1Z_{ii}^T.$$ 

**Lemma 7.2.** Suppose that both $S_o$ and $S_i$ are non-empty. Then we have

$$\frac{pr'_{S_o}(\alpha, s)w_o^T}{pr'_{S_i}(\alpha, s)w_i^T} \geq \frac{pr_{S_o}(\alpha, s)w_o^T}{pr_{S_i}(\alpha, s)w_i^T} \geq \frac{pr''_{S_o}(\alpha, s)w_o^T}{pr''_{S_i}(\alpha, s)w_i^T}.$$ 

**Proof.** Let $Z''$ denote the lazy random walk transition probability matrix of $G_S$. 

\[\begin{align*}
\end{align*}\]
where $G_S$ is the subgraph of $G$ restricted to $S$. Then the following equations hold:

\[
\begin{align*}
pr'_S(\alpha,s) &= \alpha s_S + (1-\alpha) \left( pr'_S(\alpha,s)Z_{ii} + pr'_S(\alpha,s)Z_{oi} \right), \quad (7.2) \\
pr'_S(\alpha,s) &= \alpha s_S + (1-\alpha) \left( pr'_S(\alpha,s)Z_{ii} + pr'_S(\alpha,s)Z_{oi} \right), \quad (7.3) \\
pr''_S(\alpha,s) &= \alpha s_S + (1-\alpha) \left( pr''_S(\alpha,s)Z''_{ii} + pr''_S(\alpha,s)Z''_{oi} \right). \quad (7.4)
\end{align*}
\]

Let $c_1 = \alpha s_S 1^\top$; the definitions of $w_o$ and $w_i$ give:

\[
\frac{pr''_S(\alpha,s)w_o^\top}{pr'_S(\alpha,s)w_i^\top} = \frac{pr''_S(\alpha,s)w_i^\top - c_1}{pr'_S(\alpha,s)w_i^\top}.
\]

Subtracting (7.3) from (7.2) yields:

\[
(pr'_S(\alpha,s) - pr'_S(\alpha,s)) (I - (1-\alpha)Z_{ii}) = (pr'_S(\alpha,s) - pr'_S(\alpha,s)) ((1-\alpha)Z_{oi}).
\]

Let $c_2 = (pr'_S(\alpha,s) - pr'_S(\alpha,s)) (I - (1-\alpha)Z_{ii}) 1^\top$. Since for all $v \in S, pr(\alpha,s)_v \leq pr'(\alpha,s)_v$, it follows that $pr''_S(\alpha,s) - pr'_S(\alpha,s)$ and $c_2$ are both nonnegative. Thus, we have

\[
\frac{pr''_S(\alpha,s)w_o^\top}{pr'_S(\alpha,s)w_i^\top} = \frac{pr''_S(\alpha,s)w_i^\top - c_1}{pr'_S(\alpha,s)w_i^\top} = \frac{pr'_S(\alpha,s)w_i^\top + c_2 - c_1}{pr'_S(\alpha,s)w_i^\top + c_2}.
\]

Since for every $v \in S_o$, the degree of $v$ in $G_S$ is strictly smaller than the degree of $v$ in $G$, we have $Z''_{v,u} \geq Z_{v,u} (1 + \frac{1}{d})$, where $d$ is the maximum degree among vertices in $S_o$ in $G_S$. For $v \in S_i$, there is no change in degree from $G$ to $G_S$, so $Z_{ii} = Z''_{ii}$.

Hence, we have

\[
\begin{align*}
pr''_S(\alpha,s) (I - (1-\alpha)Z_{ii}) &= \alpha s_S + pr''_S(\alpha,s) ((1-\alpha)Z''_{oi}) \\
&\geq \alpha s_S + pr''_S(\alpha,s) ((1-\alpha)Z_{oi}) \left( 1 + \frac{1}{d} \right),
\end{align*}
\]

Therefore,

\[
\frac{pr''_S(\alpha,s)w_o^\top}{pr''_S(\alpha,s)w_i^\top} \leq \frac{pr''_S(\alpha,s)w_i^\top - c_1}{(1 + \frac{1}{d}) pr'_S(\alpha,s)w_i^\top},
\]

and the lemma follows from

\[
\frac{pr_S(\alpha,s)w_i^\top + c_2 - c_1}{pr_S(\alpha,s)w_i^\top + c_2} \geq \frac{pr_S(\alpha,s)w_i^\top - c_1}{pr_S(\alpha,s)w_i^\top - c_1} \geq \frac{pr_S(\alpha,s)w_i^\top - c_1}{(1 + \frac{1}{d}) pr_S(\alpha,s)w_i^\top}.
\]

\[\square\]
We remark that $\text{pr}''$ tends to underestimate the ranking since it ignores all the boundary edges. In the other direction, the influence of boundary vertices should be taken into consideration but not so much as to overestimate the PageRank on $S_o$ in comparison with $\text{pr}'S_o$. Lemma 7.2 shows that $\text{pr}_{S_o}$ is bounded between $\text{pr}'S_o$ and $\text{pr}''S_o$ as desired.

### 7.2 Dirichlet PageRank with given boundary conditions

In this section, we generalize Dirichlet PageRank to use arbitrary boundary conditions given by $\sigma : \delta(S) \to \mathbb{R}$. Note that $\sigma$ can have negative values.

The Dirichlet PageRank vector with given boundary conditions $\sigma$ is defined by the equations:

$$
\text{pr} = \begin{cases}
\alpha s + (1 - \alpha) \sum_{u \in V} \text{pr}(\alpha, s) Z_{uv} & \text{if } v \in S, \\
\sigma(v) & \text{if } v \in \delta S.
\end{cases}
$$

(7.5)

Here, we use the convention that $||\sigma||_1 \leq 1$. Let $Z_{\delta S}$ denote $Z$ restricted to $\delta S \times S$.

**Theorem 7.3.** For a connected graph $G$, vector $s$, $\alpha > 0$ and given boundary conditions $\sigma$, the above PageRank equation (7.5) has one and only one solution. With $\beta = \frac{2\alpha}{1-\alpha}$, it is given by

$$
\text{pr}_S(\alpha, s) = (\beta s + 2\sigma_{\delta S}Z_{\delta S})D_S^{-1/2}g_{S,\beta}D_S^{1/2}.
$$

**Proof.** Note that

$$
\text{pr}_S(\alpha, s) = \alpha s + (1 - \alpha) (\text{pr}_S(\alpha, s) Z_S + \sigma_{\delta S}Z_{\delta S})
$$

(7.6)

$$
= \frac{1 - \alpha}{2} (\beta s + 2\sigma_{\delta S}Z_{\delta S}) + (1 - \alpha)\text{pr}_S(\alpha, s) Z_S.
$$

The theorem follows by expressing $Z_S$ in terms of $L_S$ and solving for $\text{pr}_S(\alpha, s)$, as in the proof of Theorem 7.1.

To solve Problem 2, one can adjust the ranking by setting boundary conditions $\sigma$ and solving the Dirichlet PageRank equation. We can use $\sigma$ to specify known quantitative distrust which will then propagate to the rest of vertices in $S$. 

7.3 The ApproxDirichPR algorithm and analysis

Solving the PageRank equations [(7.1) or (7.5)] with boundary conditions requires both matrix-vector multiplication and solving a linear system of the form:

\[ x(\beta I_S + L_S) = y. \]

The running time of solving the PageRank equation is dominated by the complexity of solving the linear system. Since the matrix \( \beta I_S + L_S \) is diagonally dominant, it can be solved approximately in nearly-linear time with a Spielman-Teng Solver [125], but we will also give a simpler algorithm ApproxDirichPR to compute approximate Dirichlet PageRank vectors. This approximation algorithm is faster and has a better approximation ratio if the constant \( \alpha \) is not too small.

The algorithm ApproxDirichPR is outlined as follows: we initialize \( \text{pr}_S(\alpha, s) \) as 0 and maintain a residue \( r \), which is the difference between the right side and left side of equation (7.6). Then we gradually move the ‘mass’ from \( r \) to \( \text{pr}_S(\alpha, s) \) while maintaining the following invariant:

\[
\text{pr}_S(\alpha, s) + r = \alpha s_S + (1 - \alpha) (\text{pr}_S(\alpha, s)Z_S + \sigma \delta_S Z_{\delta S})
\]

until we have \( r_v \leq \epsilon'd_v \) for every \( v \in S \). At the start, we set \( \epsilon' = 1 \). After each iteration, we decrease \( \epsilon' \) by half until \( \epsilon' \leq \epsilon \) which is the given desired approximation ratio.

Algorithm ApproxDirichPR

Input: a graph \( G = (V, E) \), a subset of vertices \( S \subseteq V \), the jumping constant \( \alpha \), a seed vector \( s \), the boundary conditions \( \sigma \), and an approximation parameter \( \epsilon > 0 \). Output: an approximate Dirichlet PageRank vector \( \text{pr}_S(\alpha, s) \).

- Initialize \( \text{pr}_S(\alpha, s) = 0, \epsilon' = 1, r = \alpha s_S + (1 - \alpha)\sigma \delta_S Z_{\delta S} \).

- While \( \epsilon' > \epsilon \):

  - While \( |r_v| \geq \epsilon'd_v \) for some \( v \):

    * Let \( \text{pr}_S(\alpha, s)_v = \text{pr}_S(\alpha, s)_v + r_v \).
For each neighbor $u$ of $v$, let $r_u = r_u + (1 - \alpha)r_v/2d_v$.

* Let $r_v = (1 - \alpha)r_v/2$.

- Let $\epsilon' = \epsilon'/2$.

The correctness and running time of the above algorithm are as follows:

**Theorem 7.4.** For any $\epsilon \in (0, 1)$ and any jumping constant $\alpha \in (0, 1)$, the algorithm $\text{ApproxDirichPR}$ outputs an $\epsilon$-approximate Dirichlet PageRank vector $\tilde{\text{pr}}_S(\alpha, s)$ in time $O\left(\frac{\text{vol}(S)\log \frac{1}{\epsilon}}{\alpha}\right)$, which, compared to the exact Dirichlet PageRank $\text{pr}_S(\alpha, s)$, satisfies:

$$||\text{pr}_S(\alpha, s) - \tilde{\text{pr}}_S(\alpha, s)||_1 < \frac{\epsilon\text{vol}(S)}{\alpha}.$$  

**Proof.** To bound the running time, we first show that in each iteration of the inner loop, $||\text{r}||_1$ will decrease by at least $\alpha\epsilon'd_v$. Let $\text{r}^b$ be $\text{r}$ before the iteration and $\text{r}^a$ be $\text{r}$ after the iteration. We have:

$$||\text{r}^a||_1 = |r_v^a| + \sum_{u \neq v} |r_u^a|$$

$$= \frac{1 - \alpha}{2} |r_v^b| + \sum_{u \neq v} |r_u^b| + \frac{1 - \alpha}{2d_v} |r_v^b|$$

$$\leq \frac{1 - \alpha}{2} |r_v^b| + \sum_{u \neq v} \left( |r_u^b| + \frac{1 - \alpha}{2d_v} |r_v^b| \right)$$

$$= (1 - \alpha) |r_v^b| + \sum_{u \neq v} |r_u^b|$$

$$= ||\text{r}^b||_1 - \alpha |r_v^b|.$$  

We note that the above equations hold for both positive and negative values of $r_v^b$. Since $v$ is chosen satisfying $|r_v| \geq \epsilon'd_v$, it follows that $||\text{r}^a||_1 \leq ||\text{r}^b||_1 - \alpha\epsilon'd_v$.

Note that at the beginning of each iteration of the outer loop, $||\text{r}||_1 \leq 2\epsilon'\text{vol}(S)$. Let $T$ be the number of iterations of the inner loop and $v_i$ be the vertex selected at the $i$th iteration for $1 \leq i \leq T$. We have:

$$\sum_{i=1}^{T} \alpha\epsilon'd_{v_i} \leq 2\epsilon'\text{vol}(S),$$
which implies
\[ \sum_{i=1}^{T} d_{v_i} \leq \frac{2\text{vol}(S)}{\alpha}. \]

Since a FIFO queue can be used to store every vertex \( v \) such that \( |r_v| \geq \epsilon' d_v \), each iteration of the inner loop can be completed in \( O(d_v) \) time. Therefore, the running time of one outer iteration is \( \sum_{i=1}^{T} d_{v_i} \), which is bounded from above by \( \frac{2\text{vol}(S)}{\alpha} \).

There are \( \log\left(\frac{1}{\epsilon'}\right) \) iterations of the outer loop; therefore, the overall running time is \( O\left(\frac{\text{vol}(S) \log\left(\frac{1}{\epsilon'}\right)}{\alpha}\right) \).

To prove the correctness of the approximation ratio, we will first show that the following invariant is maintained during the entire algorithm:

\[ \text{pr}_S(\alpha, s) + r = \alpha s_S + (1 - \alpha) (\text{pr}_S(\alpha, s) Z_S + \sigma S Z_S). \]

This equation holds trivially in the beginning where \( \text{pr}_S(\alpha, s) = 0 \) and \( r = \alpha s_S + (1 - \alpha) \sigma S Z_S \). For each inner iteration, let \( r^b, \text{pr}_S^b(\alpha, s) \) be \( r, \text{pr}_S(\alpha, s) \) before the iteration and \( r^a, \text{pr}_S^a(\alpha, s) \) be \( r, \text{pr}_S(\alpha, s) \) after the iteration. We have

\[
\begin{align*}
\text{pr}_S^a(\alpha, s)_v + r_v^a &= \text{pr}_S^b(\alpha, s)_v + r_v^b + \frac{1 - \alpha}{2} r_v^b \\
&= \alpha s_S(v) + (1 - \alpha) \left( [\text{pr}_S^b(\alpha, s)_v]_v + [\sigma S Z_S]_v \right) + \frac{1 - \alpha}{2} r_v^b \\
&= \alpha s_S(v) + (1 - \alpha) \left( \frac{1}{2} \text{pr}_S^b(\alpha, s)_v + \sum_{w \neq v} \frac{1}{2d_w} \text{pr}_S^b(\alpha, s)_w + [\sigma S Z_S]_v \right) \\
&\quad + \frac{1 - \alpha}{2} r_v^b \\
&= \alpha s_S(v) + (1 - \alpha) \left( \frac{1}{2} \left( [\text{pr}_S^a(\alpha, s)_v]_v + [\sigma S Z_S]_v \right) \right) + \sum_{w \neq v} \frac{1}{2d_w} [\text{pr}_S^a(\alpha, s)_w + [\sigma S Z_S]_w]_v \\
&= \alpha s_S(v) + (1 - \alpha) \left( \frac{1}{2} \left( [\text{pr}_S^a(\alpha, s)_v]_v + [\sigma S Z_S]_v \right) \right) + \sum_{w \neq v} \frac{1}{2d_w} [\text{pr}_S^a(\alpha, s)_w + [\sigma S Z_S]_w]_v \\
&= \alpha s_S(v) + (1 - \alpha) \left( [\text{pr}_S^a(\alpha, s)_v]_v + [\sigma S Z_S]_v \right),
\end{align*}
\]
and for \( u \neq v \),

\[
pr_S^a(\alpha, s)_u + r_u^a = \frac{1 - \alpha}{2d_u} r_v^b + r_u^b
\]

\[
= \alpha s_S(u) + \frac{1 - \alpha}{2d_u} r_v^b + (1 - \alpha) ([pr_S^b(\alpha, s)Z_S]_u + [\sigma_{\delta S} Z_{\delta S}]_u)
\]

\[
= \alpha s_S(u) + \frac{1 - \alpha}{2d_u} r_v^b
\]

\[
+ (1 - \alpha) \left( \frac{1}{2} pr_S^b(\alpha, s)u + \sum_{w \neq u} \frac{1}{2d_w} pr_S^b(\alpha, s)_w - \frac{1}{2d_u} r_S^b(v) + [\sigma_{\delta S} Z_{\delta S}]_u \right)
\]

\[
= \alpha s_S(u) + (1 - \alpha) \left( \frac{1}{2} pr_S^b(\alpha, s)u + \sum_{w \neq u} \frac{1}{2d_w} pr_S^a(\alpha, s)_w + [\sigma_{\delta S} Z_{\delta S}]_u \right)
\]

\[
= \alpha s_S(u) + (1 - \alpha) ([pr_S^a(\alpha, s)Z_S]_u + [\sigma_{\delta S} Z_{\delta S}]_u).
\]

Thus, the invariant is maintained during the entire algorithm. Note that the equations hold for both positive and negative values of \( r \). As a result, the output \( \tilde{pr}_S(\alpha, s) \) satisfies:

\[
\tilde{pr}_S(\alpha, s) + r = \alpha s_S + (1 - \alpha) (\tilde{pr}_S(\alpha, s)Z_S + \sigma_{\delta S} Z_{\delta S}),
\]

where \( |r_v| < \epsilon d_v \) for all vertices \( v \in S \), and the exact solution \( pr_S(\alpha, s) \) satisfies

\[
pr_S(\alpha, s) = \alpha s_S + (1 - \alpha) (pr_S(\alpha, s)Z_S + \sigma_{\delta S} Z_{\delta S}).
\]

Taking the difference of these two equations, we get

\[
pr_S(\alpha, s) - \tilde{pr}_S(\alpha, s) = r + (1 - \alpha) ((pr_S(\alpha, s) - \tilde{pr}_S(\alpha, s)Z_S).
\]

Since

\[
||pr_S(\alpha, s) - \tilde{pr}_S(\alpha, s)Z_S||_1 \leq ||pr_S(\alpha, s) - \tilde{pr}_S(\alpha, s)||_1,
\]

we have

\[
||pr_S(\alpha, s) - \tilde{pr}_S(\alpha, s)||_1 \leq ||r||_1 + (1 - \alpha) ||(pr_S(\alpha, s) - \tilde{pr}_S(\alpha, s))Z_S||_1
\]

\[
\leq ||r||_1 + (1 - \alpha) ||pr_S(\alpha, s) - \tilde{pr}_S(\alpha, s)||_1.
\]
which implies
\[ \| \text{pr}_S(\alpha, s) - \tilde{\text{pr}}_S(\alpha, s) \|_1 \leq \frac{1}{\alpha} \| r \|_1 < \frac{\epsilon \text{vol}(S)}{\alpha}. \]

\[ \Box \]

7.4 Applications of Dirichlet PageRank

7.4.1 Allowing negative edges in the graph

Negative edges arise in many network problems, making PageRank less suitable (see [88]) for finding a desirable ranking. There have been numerous attempts to address this problem; one way is to ignore the entries corresponding to the negative links, as seen as in [81, 94]. By treating a negative link between two vertices the same as no link [81, 94], the PageRank vector can be computed and used as a ranking. Unfortunately, the information contained in those negative links is lost. To remedy this, in [67], a trust ranking is computed based on only positive links and one single step of propagation of distrust. Namely, the distrust is only propagated to immediate neighbors without influencing the rest of the vertices. A more sophisticated algorithm PageTrust is proposed in [88], which uses a fairly complicated update rule relying on a relatively large number of iterations until convergence. However, the running time for each iteration is quite large: $O(\overline{d} \text{nn}$), where $\overline{d}$ is the average degree and $n^-$ is the number of vertices receiving negative links. Thus, the worst case complexity is $O(n^3)$.

Using Dirichlet PageRank, we develop a simple and fast algorithm to propagate distrust in graphs with negative edges. The key idea can be outlined as follows: We first compute the usual PageRank based on positive edges, then based on the ranking result, we convert negative links to boundary conditions and then compute Dirichlet PageRank.

Let $E^+$ be the set of positive edges, $E^-$ be the set of negative edges, and $V^-$ be the set of vertices incident to negative edges. Let $d^-(u)$ and $d^+(u)$ denote the numbers of negative and positive edges incident to $u$, respectively. For each vertex $u \in V^-$, we create a shadow vertex $u^s$. Let $V^s$ denote the set of all shadow
vertices, and define

\[ E^* = \{ \{u^*, v\}, \{u, v^*\} \mid \{u, v\} \in E^- \}. \]

We then form a new graph \( \hat{G} = (\hat{V}, \hat{E}) \), where \( \hat{V} = V \cup V^* \), and \( \hat{E} = E^+ \cup E^* \). In the following algorithm, we will set boundary conditions on \( V^* \) in \( \hat{G} \) to propagate distrust.

**Algorithm NegLinkPageRank**

**Input:** a graph \( G = (V, E) \), a vertex \( v \in V \), the jumping constant \( \alpha \), and an approximation parameter \( \epsilon \).

**Output:** A PageRank vector \( pr(\alpha, s) \).

- Determine \( \hat{G} \), \( V^* \) and \( E^+ \) as given above.
- Compute \( pr^+(\alpha, s) = SharpApproximatePR(v, \alpha, \epsilon) \) using the graph \((V, E^+)\).
- Let \( \sigma(u) \leftarrow \frac{d^-}{d^+} pr^+(\alpha, s)_u \) for each vertex \( u \) in \( V^* \).
- Compute \( pr(\alpha, s) = ApproxDirichPR(\hat{G}, V, \alpha, v, \sigma, \epsilon) \).

The intuition behind setting the values of the boundary condition \( \sigma(u) \) is as follows: We let \( \sigma(u)/d^- = pr^+(\alpha, s)_u / d^+ \). Namely, for a vertex, the amount of distrust propagated via the negative edge is equal to the amount of trust propagated via the positive edge. The running time of this algorithm is nearly linear time, using our ApproxDirichPR algorithm.

As an example of using Dirichlet PageRank on a graph with positive and negative edges, we examine a network of tribes in New Guinea, studied in the mid-twentieth century [70, 117]. A positive edge indicates a tribal alliance, and negative edges represent enemy relationships. As illustrated in Figure 7.3a, the positive edges are in light green and the negative edges are in light red. One way to calculate a vertex ranking is to ignore the negative edges as shown in Figure 7.3a. The PageRank vector computed in this manner is actually the uniform distribution.
In Figure 7.3b, we use the Dirichlet PageRank to compute a vertex ranking taking the negative edges into account. It is apparent that vertices are appropriately ranked by taking advantages of their trusting and distrusting relationships.

Figure 7.3: Comparison of vertex rankings on a network [70, 117] with negative edges

7.4.2 Adjusting Spammers’ Influence

One disadvantage of pure link-based ranking systems such as PageRank is that they interpret all vertices as honest agents and all links as votes or validation between vertices. However, real-world networks such as the World Wide Web often contain malicious vertices or spammers. Of interest is to find ranking systems that can better represent the true ranking of vertices in the graph.

There are many schemes developed to tackle such problems [6, 15, 20, 31, 67, 69, 81, 88]. It turns out that many of these schemes can be modeled using the Dirichlet PageRank with different boundary conditions. For example, [15] outlines an algorithm SpamRank which penalizes spam vertices. This algorithm uses sampling to find vertices whose PageRank vectors are significantly different from their neighbors’, and gives a heuristic penalty score for each vertex. It then uses these
penalty scores as a seed vector for personalized PageRank computation. However, if the penalties is represented by a probability distribution, the sampling techniques can be problematic for vertices with low degree. Using the Dirichlet PageRank, we can penalize known spammers $v$ by enforcing the condition $pr(\alpha, s)_v = 0$. This is done in the example given in Figure 7.2. Furthermore, one can adjust the ranking even further by enforcing $pr(\alpha, s)_v = -1$.

In [20], the trust and distrust are propagated within a network through a weighted random walk with a trusted seed vertex $s$ by assigning at the start, the rank $pr(\alpha, s)_s = 1$. This can be regarded as the Dirichlet PageRank with the boundary condition $\sigma(s) = 1$. There is a subtle difference in the way distrust is handled (the algorithm in [20] does not allow for the propagation of trust scores less than 0). We note that the Dirichlet PageRank allows us to efficiently consider these and many other models.

### 7.4.3 Adjusting Rank Based on Trust

While it is important to devise ranking systems that take known spammers into account, it is also crucial to be able to calculate a ranking based on various notions of trust in a network. There are numerous scenarios for which the Dirichlet PageRank with boundary conditions can serve as a useful algorithmic tool.

We consider the following problem: In a network $G$, the vertex $v$ wants to compute a personalized ranking of the vertices, but $v$ trusts its own friends and wants its ranking on the top $\rho$ fraction of vertices to be similar to its friends’. This quantifies the assumptions that one’s friends’ actions carry a great deal of weight in one’s own decisions. Vertex $v$ can efficiently compute a personalized PageRank vector as its ranking function using algorithms from [46]. However, PageRank alone will not take into account the implied trust between $v$ and its friends. But using Dirichlet PageRank with boundary conditions, we can take $v$’s trusted friends into account. We illustrate this in the algorithm \textbf{PRTrustedFriends}.

<table>
<thead>
<tr>
<th>Algorithm \textbf{PRTrustedFriends}</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A graph $G = (V, E)$, a vertex $v \in V$, the jumping constant $\alpha$, the vertex</td>
</tr>
</tbody>
</table>
fraction $\rho$, and an approximation parameter $\epsilon$.

Output: A ranking vector $p$.

- Compute $v$’s ranking using $\text{SharpApproximatePR}$ from [46]:
  \[ p = \text{SharpApproximatePR}(v, \alpha, \epsilon) \]

- Compute $v$’s neighbors’ rankings: for each neighbor $u$:
  \[ p_u = \text{SharpApproximatePR}(u, \alpha, \epsilon) \]

- Take a weighted average of $v$’s neighbors’ rankings:
  \[ p' = \frac{1}{\sum_{u \sim v} p_u} \sum_{u \sim v} p(u) p_u \]

- Compute $v$’s neighbors’ rankings: for each neighbor $u$,
  \[ p'_u = \frac{1}{\sum_{u \sim v} p_u} \sum_{u \sim v} p_u \]

- Take a set $S$ of vertices that $v$ ranks highly:
  \[ S = \arg \max_{S \subseteq V, |S| \leq \rho |V|} \sum_{s \in S} p_s \]

- Finally, use $v$’s friends’ ranking of $S$ to adjust $p$
  \[ p = \text{ApproxDirichPR}(G, V \setminus S, \alpha, v, p', \epsilon) \]

A natural extension of $\text{PRTrustedFriends}$ can be described as the case that $v$ is a newcomer to a network and is therefore unsure about what other vertices are trustworthy. In such a scenario, the only available information to $v$ is the network itself. For ranking purposes, $v$ can select a small number of vertices to compare with its own ranking. If these vertices are well distributed, this provides some control to ensure that $v$’s own ranking function is not too distorted.
by the presence of nearby spam or malicious vertices. We give the algorithm PRValidation.

**Algorithm PRValidation**

Input: A graph $G = (V, E)$, a vertex $v \in V$, the sample size $k$, the jumping constant $\alpha$, the vertex fraction $\rho$, and an approximation parameter $\epsilon$.

Output: A ranking vector $p$.

- Compute $v$’s ranking using $\text{SharpApproximatePR}$ from [46]:
  \[ p = \text{SharpApproximatePR}(v, \alpha, \epsilon) \]

- Draw $v_1, \ldots, v_k$ independently from $V$, with probability distribution $p$.

- Compute rankings for the sampled vertices:
  \[ p_k = \text{SharpApproximatePR}(v_k, \alpha, \epsilon) \]

- Take a weighted average of these sampled rankings:
  \[ p' = \frac{1}{\sum_{i=1}^{k} p_{v_k}} \sum_{i=1}^{k} p_{v_k} p_k \]

- Take a set $S$ of vertices that $v$ ranked highly:
  \[ S = \arg \max_{S \subseteq V, |S| \leq \rho |V|} \sum_{s \in S} p_s \]

- Use the sampled rankings to adjust $p$:
  \[ p = \text{ApproxDirichPR}(G, V \setminus S, \alpha, v, p', \epsilon) \]

A third, more complex situation arises in the context of different types of social networks. Although the setup here appears somewhat complicated, it is a natural model for a common social phenomenon, addressing distinctions among different types of social networks.
Suppose that we have two networks $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $V_1 \subseteq V_2$. We interpret $G_1$ as a closely-knit social network where edges represent a deep mutual trust for one another. $G_2$ is a larger network where edges are formed by relatively weak reasons, such as acquaintance or curiosity. We assume that a vertex $v$ does not know much about the many sources of information acquired through in $G_2$. An important question for $v$ is to determine which vertices in $G_2$ are trustworthy? How should the vertices of $G_2$ be ranked by taking advantage of $G_1$?

One effective way of finding such a ranking for vertices $v$ in $G_2$ is to compute the ranking on $G_1$ first and then compute Dirichlet PageRank on $G_2$ using $G_1$’s ranking as the boundary condition. This is outlined in the algorithm PRTrust-Network.

**Algorithm PRTrustNetwork**

Input: Two graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, a vertex $v \in V_1 \cap V_2$, the jumping constant $\alpha$, and an approximation parameter $\epsilon$.

Output: A ranking vector $q$.

- Compute $p = \text{SharpApproximatePR}(v, \alpha, \epsilon)$ [46] for $G_1$.
- Compute $q = \text{ApproxDirichPR}(G_2, V_2 \setminus V_1, \alpha, v, p, \epsilon)$.

The Dirichlet PageRank can also be used to solve a related problem when a global ranking for $G_2$ is already known or pre-computed. Suppose that such a ranking for $G_2$ exists, and a vertex $v \in G_1$ wishes to be able to rank $G_1$ by taking this into account. One way to do this is to compute a Dirichlet PageRank vector for $G_1$, but using the adjacent vertices in $G_1$ as a boundary with ranking given by the global ranking on $G_2$. This procedure is given in the algorithm PRInferRanking as follows.
Algorithm **PRInferRanking**

**Input:** Two graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ with $G_1 \subseteq G_2$, a vertex $v \in V_1$, a ranking vector $p$ for $G_1$, the jumping constant $\alpha$, and an approximation parameter $\epsilon$.

**Output:** A ranking vector $q$ for $G_2$.

- Let $\partial E_1 = \{(w, x) \in E_2 \mid w \in V_1, x \notin V_1\}$.
- Let $\partial V_1 = \{w \in V_2 \setminus V_1 \mid w$ is an endpoint of an $e \in \partial E_1\}$.
- Compute $q = \text{ApproxDirichPR}((V_1 \cup \partial V_1, E_1 \cup \partial E_1), V_1, \alpha, v, p, \epsilon)$. 

### 7.5 Acknowledgement

Material in this chapter appears in the following published articles:


Chapter 8

Distributing Antidote to Combat Network Epidemics Using PageRank

The spreading and containment of epidemics on networks is a widely-studied problem with many applications in modeling both disease outbreaks in human and animal populations as well as the spread of viruses and worms on technological networks such as the Internet, online social networks, and email. Many analytical models have been used to address numerous crucial problems, such as the conditions for disease spreading, the critical threshold for the infection rate, the duration of persistent epidemics, and the effective distribution of limited amounts of antidote. We will examine a well-studied contact process model [19, 111], coupled with an inoculation scheme using PageRank vectors. In this chapter, we give a complete analysis of our scheme, showing the improved efficiency of the inoculation scheme without affecting the performance guarantee as in previous results in [19].

A contact graph consists of a set of nodes together with prescribed pairs of nodes where direct contact can take place and infections can spread (see [30, 64, 111]). Analysis of spreading on the contact graph is performed with the contact process, a continuous-time Markov process originally studied in the first half of the twentieth century [89]. Since then, it has been applied specifically to network epidemics in many contexts, including social networks [131], Internet viruses [16],
and crop disease [61].

Previously, most analysis of network infection models concerned determining the critical infection threshold [16, 64, 111]. There is a parameter, known as the infection rate, that models the virulence or resistance of a given epidemic, and with it comes a threshold: if the infection rate exceeds that point, then an epidemic will persist indefinitely. In these analyses, the infected nodes became healthy all at the same rate. In the contact process, this occurs when an equal amount of antidote is sent indiscriminately to all nodes, requiring a large amount of antidote. In practice, this is often undesirable; in this chapter, we will give a model that avoids such widespread antidote distribution.

Another approach is to combat epidemics by using contact tracing, or inoculating neighbors of infected nodes, using a total amount of antidote that depends only on the sum of the degrees of the infected nodes. However, both simulation and mathematical analysis have shown that contact tracing can be ineffective, especially on large real-world graphs that exhibit small-world phenomena and power-law degree distributions [52, 90, 102, 115, 116, 131].

In [19], Borgs et al. show that for the contact process on a contact graph $G$, inoculating every node with antidote equal to its degree will result in any infection dying out in $O(\log n)$ time with high probability, where $n$ is the number of nodes in $G$. This scheme uses a total amount of antidote equal to the sum of the degrees in $G$. For special graphs such as the expanders, it was shown [19] that such a large amount of antidote is necessary (up to a constant factor) when a constant proportion of the nodes are initially infected. Our proposed model will not improve this result on expander graphs, but for many classes of graphs, we will require antidote only on smaller portions of the network.

In this chapter, we analyze an inoculation scheme using PageRank vectors. PageRank was first introduced by Brin and Page [22] for Web search algorithms. Although the original definition is for the Web graph, PageRank is well defined for any graph, including the contact graphs that we study. Here, we will use a modified version of PageRank, known as personalized PageRank, for the contact graph with the initially infected nodes as seeds. PageRank captures the quantitative
correlation between pairs or subsets of nodes. For example, if the contact graph has some small cuts or bottlenecks, it is likely that an infection will not propagate through them, and nodes on the other side will have low PageRank. Our inoculation scheme using PageRank specifies the selected nodes for sending antidote and provides a probabilistic guarantee for the termination of any epidemic. Furthermore, the number of selected nodes for inoculation is usually much smaller than the total size of the contact graph, thus improving the previous schemes which inoculated all the nodes. The number of selected nodes depends only on the number of initially infected nodes, the probabilistic guarantee bound and the isoperimetric invariant of the graph, the Cheeger ratio (to be defined later). Hence, it is independent of the total size of the contact network.

Previously, an empirical study [106] found that inoculating nodes according to their PageRank works well in combating epidemics for certain examples of contact networks. Our analysis complements this experimental work and is applicable to any given general contact network. Note that their work uses the original definition of PageRank [22]; we will use personalized PageRank vectors [72, 75] in this chapter. The analysis in Section 8.1 provides a trade-off between the probabilistic guarantee of termination and the time required.

### 8.1 Infection model and inoculation scheme

We model an epidemic spreading on a contact graph $G = (V, E)$, an undirected graph where the vertices are the agents and the edges are connections along which infections can spread.

For a given value $h$, we say that $H$ is an $h$-cluster if its Cheeger ratio $h(H)$ satisfies $h(H) \leq h$. For an $h$-cluster $H$ and a given $\alpha$, the $\alpha$-core $C$ of $H$ is the set of all vertices $u$ so that the personalized PageRank on $H$, with seed $u$ and jumping constant $\alpha$, is at least $1 - \frac{h}{\alpha}$. Using PageRank notation from Chapter 2:

$$C = \left\{ u \in V \mid \text{pr}(\alpha, u)_H \geq 1 - \frac{h}{\alpha} \right\}.$$  \hspace{1cm} (8.1)

It has been shown [8] that if $C$ is the $\alpha$-core of $H$, then $\text{vol}(C) \geq \frac{1}{2} \text{vol}(H)$. This
indicates that there are many nodes $u \in H$ for which the personalized PageRank vector $\text{pr}(\alpha, u)$ has very little mass outside of $H$ if $\alpha$ is larger than $h$.

We use the following contact process (see [89]) as our infection model, which is also used in [16]. The contact process is a continuous-time Markov process parametrized by $\beta$, the infection rate, with $0 \leq \beta < 1$ and $\mathbf{c} = (c_1, c_2, \ldots, c_n)$, the cure vector. We assume that at time $t = 0$, a seed set $S \subseteq V$ is infected. We will use $\mathbf{1}_S$ to denote the indicative vector associated with $S$, and $\mathbf{x}(0) = \mathbf{1}_S$.

Each node $v_i$ has an infection state $x_i(t)$; a node is considered “healthy” if $x_i(t) = 0$, and “infected” if $x_i(t) = 1$. Thus, the entire process is characterized by a state vector $\mathbf{x}(t) = (x_1(t), x_2(t), \ldots, x_n(t))$. The state transitions are as follows:

- If a node $x_j$ is infected, an adjacent node $x_i$ becomes infected at rate $\beta$. We refer to this transition as a spread event.
- An infected node $v_i$ becomes healthy at rate $c_i$. We refer to this transition as a cure event.

In any continuous-time Markov process, for a transition (e.g., spread or cure event) that occurs with rate $\lambda$, the elapsed time until that transition takes place assumes an exponential random variable with parameter $\lambda$, which is independent of any state information. Such a random variable has a probability density function $f(x) = \lambda e^{-\lambda x}$ for $x \geq 0$, and 0 otherwise. We denote such a random variable as $\text{Expo}(\lambda)$.

Using this contact process as a model, our goal is to choose $\mathbf{c}$ such that with high probability, the infection dies out quickly, and the total amount of antidote used is small. Furthermore, we want $\mathbf{c}$ to only depend on the seed set $S$ and the degree distribution $\mathbf{d}$, but not on $t$ or $\mathbf{x}(t)$. Our main theorem describes how to find such a cure vector $\mathbf{c}$. First, we establish the relationship between PageRank and the infection starting from $S$ but leaving a specified area.

**Theorem 8.1.** Suppose that an infection starts in $S \subseteq H \subseteq V$ (where $|S| = s$) with infection rate $\beta$, and each node $v \in H$ is inoculated with $c_v = d_v$. Let $\mathcal{E}_H$ denote the event that an infection started in $S$ ever leaves the set $H$. Then $\mathcal{E}_H$ can
be upper bounded by the PageRank vector as follows:

\[ \Pr[\mathcal{E}_H] \leq \frac{s}{\beta} \Pr \left( 1 - \beta, \frac{1}{s} \right)_H. \]

The proof of Theorem 8.1 will be given in Section 8.3. Using Theorem 8.1, we can further derive the following:

**Theorem 8.2.** Let \( G \) be a contact graph with \( n \) nodes, \( S \) be an initial set of infected nodes with \( |S| = s \), and \( \beta \) be the infection rate with \( 0 \leq \beta < 1 \). Suppose that \( H \) is an \( h \)-cluster that contains \( S \) in its \((1 - \beta)\)-core. If all nodes in \( H \) are inoculated with antidote equal to their degrees, then with probability at least \( 1 - 2\frac{sh}{\beta(1 - \beta)} \), any infection starting from \( S \) will die out in at most \( c \log(1/h) + c' \) time, where \( c \) and \( c' \) depend only on \( \beta \) and not on \( n \).

Theorem 8.2 will be proved in Section 8.3.

We remark that Theorem 8.2 implies a tradeoff between the Cheeger ratio \( h \) and the probabilistic bound. If the initial set of infected nodes \( S \) lies within the \((1 - \beta)\)-core of a \( h \)-cluster \( H \), the probability of the infection dying out in \( O(\log s) \) time is high, as long as the product \( sh \) is small. In particular, if the seed set \( S \) lies on one side of a small cut, it will likely lie within the core of an \( h \)-cluster with small Cheeger ratio \( h \). If there is no such small cut, then the infection is likely to spread about the graph. This leads to the following corollary:

**Corollary 8.3.** For any \( \epsilon > 0 \) and an infection starting from a seed set \( S \), if \( S \) lies within the \((1 - \beta)\)-core of an \( h \)-cluster \( H \), and \( h \leq \frac{\epsilon}{s} \beta(1 - \beta) \), then with probability at least \( 1 - 2\epsilon \), the infection will die out in \( c \log s + c' \) time, where \( c \) and \( c' \) depend only on \( \beta \) and \( \epsilon \) and not on \( n \).

The proof of the corollary follows from applying the bound on \( \epsilon \) to Theorem 8.2. We note that the above corollary explicitly relates the desired probabilistic guarantee \( \epsilon \) with the Cheeger ratio \( h \) of the \( h \)-cluster containing \( S \) in its core.

The above theorems suggest the following inoculation scheme:
Algorithm **InoculationScheme**

**Input:** A contact graph $G = (V,E)$, an initial set $S$ of $s$ infected nodes, the infection rate $\beta$, and the error bound $\epsilon$

**Output:** A cure vector $c$

- Set $h = \frac{s}{\epsilon} \beta (1 - \beta)$.
- Use **PageRank-Nibble** from [8] or **Local Partition** from [7] to find an $h$-cluster $H$ containing $S$, if one exists.
- Check to see if $S$ is in the $(1 - \beta)$-core of $H$. If so, then inoculate each node $v \in H$ with $c_v = d_v$.
- If $S$ is not in the core of $H$, or an $h$-cluster could not be found, then let $H = G$ and inoculate every node with $c_v = d_v$.

This inoculation scheme relies on being able to find a cluster $H$ that has small Cheeger ratio $h$ and contains $S$ in its core. If such an $H$ does not exist, then the algorithm will terminate with the entire graph inoculated. For example, in the case of expander graphs, the algorithm will soon terminate and is reduced to same scenario as in [19]. Nevertheless, it is likely for a general contact graph to contain small $h$-clusters. In such cases, only a small portion of the graph needs to be inoculated, while the desired performance guarantee is maintained.

Next, we consider the case that the initially infected nodes are randomly distributed in an $h$-cluster $H$. We might expect that the infection is not likely to escape $H$. This is not strictly true, because if $S$ contains some nodes near the boundary of $H$, it is still quite likely that the infection will escape. Nevertheless, we will be able to establish an upper bound for such probability by proving the following theorem:

**Theorem 8.4.** Suppose $H \subseteq G$ is an $h$-cluster, and the set $S$ of initially infected nodes consists of $s$ nodes randomly and independently selected from $H$ with probability proportional to their degrees. Suppose the infection rate is $\beta$. Then, for a
given $\epsilon$ satisfying $sh \leq \epsilon$ and $s \geq \log(1/\epsilon)/\epsilon$, if all nodes in $H$ are inoculated with antidote equal to their degrees, then with probability at least $1 - \epsilon$, any infection starting from $S$ will die out in $c \log s + c'$ time, where $c$ and $c'$ depend only on $\epsilon$ and $\beta$.

The proof of Theorem 8.4 will be given in Section 8.3. This randomized model is relevant when a disease outbreak originates in a subpopulation, which can be represented as an $h$-cluster in a larger population graph. Theorem 8.4 implies that if the subpopulation is relatively isolated from the rest of the population (i.e. the Cheeger ratio $h$ is small), then we can effectively combat the infection by only attacking the epidemic within that $h$-cluster.

### 8.2 Probabilistic analysis of infection paths

In order to prove Theorems 8.1 and 8.2, we will prove several basic facts which are need in the following brief outline of our analysis of the inoculation scheme:

- The probability of the infection spreading to a distant node is small (Lemma 8.5).
- The probability that a nearby node will remain infected for a long time is small (Lemmas 8.6, 8.7).
- The probability that an infection persists within the inoculated nodes is small (Lemma 8.8).
- The probability that an infection escapes the inoculated nodes depends on the PageRank on the uninoculated nodes $\bar{H}$, proving Theorem 8.1.
- After proving Theorem 8.1, Theorem 8.2 follows by showing that the personalized PageRank on $\bar{H}$ is small.

In this section, we will first give some definitions and then proceed to prove Lemmas 8.5–8.8.
Consider that if a vertex $v_k$ is infected at some time $t$, then the infection must have traversed some walk in the graph from a vertex $v_0 \in S$. Suppose $p = (v_0, \ldots, v_k)$ is a path in $G$ of length $k$. Let $S_p$ denote the event that $v_0$ is infected at time 0, and the infection spreads to $v_k$ before time $t$ along the path $p$. It is important to note that if $S_p$ occurs, $v_k$ is not necessarily infected at time $t$, because it could have been cured before time $t$; however, $v_k$ cannot be infected at time $t$ if no $S_p$ occurred.

For a vertex $v$, let $C(v, t)$ denote the time of the first cure event at vertex $v$ after time $t$. From this, we define a realization of a walk as the sequence of random variables $X_v$ as follows:

- A spread event from vertex $v_i$ to $v_{i+1}$ occurs at time $X_{i+1}$.
- $0 < X_1 < C(v_0, 0)$.
- For all $i \geq 1$, $X_i < X_{i+1} < C(v_i, X_i)$.

For $S_p$ to occur, the infection must spread to $v_k$ before time $t$; therefore, $S_p$ occurs if and only if there is a realization of $p$ with $X_k < t$. There are many possible realizations of $p$, but in our analysis, we will be concerned with a specific realization: the canonical realization. In this realization, given the times of all the cure and spread events, $X_i$ is the maximum over all possible realizations with those cure and spread times. Thus, the canonical realization is the latest possible infection path along $p$.

With $X_i$ as in the canonical realization, we also define an event $S'_p$ which occurs when at least one spread event from $v_i$ to $v_{i+1}$ occurs between $X_i$ and $C(v_i, X_i)$. Thus, if $S'_p$ occurs, then the infection spreads along $p$ to $v_k$, but not necessarily before time $t$. While we are primarily concerned with the event $S_p$, when the spread occurs before time $t$, it is clear that $S_p \subseteq S'_p$, and using the canonical realization allows us to prove the following lemma that indicates that the probability that an infection follows a long path is small:

**Lemma 8.5.** For any path $p$ of length $k$,

$$\Pr[S_p] \leq \Pr[S'_p] \leq \beta^k \prod_{j=0}^{k-1} \frac{1}{c_j}.$$
Proof. Let \( S_j \) denote the event that there is a spread event from \( v_j \) to \( v_{j+1} \) in between times \( X_j \) and \( C(v_j, X_i) \). Due to the Markov property of the contact process, the probability of \( S_j \) occurring is

\[
\Pr[S_j] \leq \frac{\beta}{c_j}.
\]

Since the curing process at every node is independent, we can write

\[
\Pr[S_p] \leq \Pr[S'_p] = \prod_{j=0}^{k-1} \Pr[S_j] = \beta^k \prod_{j=0}^{k-1} \frac{1}{c_j}.
\]

\[\square\]

For a walk \( p \) of length \( k \) with canonical realization \( (X_i)_{i=1}^k \), we define the canonical end time of \( p \) to be

\[
Z_p = \begin{cases} 
X_k & \text{if } S_p \text{ occurs} \\
0 & \text{otherwise.} 
\end{cases}
\]

In other words, \( Z_p \) is the last time that \( v_k \) could become infected via the path \( p \), or 0 if it is never infected via \( p \). The following lemma states that the probability that \( Z_p \) is large is small:

**Lemma 8.6.** Suppose for a path \( p \) of length \( k \), \( Z_p \) is its canonical end time. Then,

\[
\Pr[Z_p > t] \leq \frac{1}{(2k)!} t^{2k-1} e^{-\beta t} \beta^k \prod_{j=0}^{k-1} \frac{1}{c_j}.
\]

Proof. Let \( (X_i)_{i=1}^k \) denote the canonical realization of \( p \). Then,

\[
\Pr[Z_p > t] = \Pr[Z_p > t, S_p] \\
= \Pr[X_k > t, S_p] \\
\leq \Pr[X_k > t, S'_p] \\
= \Pr[X_k > t | S'_p] \Pr[S'_p].
\]

Applying Lemma 8.5, we have

\[
\Pr[Z_p > t] \leq \Pr[X_k > t | S'_p] \beta^k \prod_{j=0}^{k-1} \frac{1}{c_j}.
\]
We further observe that
\[
Pr[X_{k-1} > t \mid S'_p] = Pr \left[ \sum_{i=1}^{k} (X_i - X_{i-1}) > t \mid S'_p \right]
\]
\[
\leq Pr \left( \sum_{i=1}^{k} (C(v_i, X_{i-1}) - X_{i-1}) > t \mid S'_p \right).
\]

We consider the time between \(X_{i-1}\) and the first cure event at \(v_i\) after \(X_{i-1}\) subject to the condition \(S'_p\): at least one spread event occurred before the cure at \(C(v_i, X_{i-1})\). Therefore, the time between \(X_{i-1}\) and \(C(v_i, X_{i-1})\) is at least the time for one spread event, namely, the exponential random variable \(\text{Expo}(\beta)\), plus the time for one cure event, \(\text{Expo}(c_i)\). Thus, we have
\[
\sum_{i=1}^{k} (C(v_i, X_{i-1}) - X_{i-1}) \geq \sum_{i=1}^{k} (\text{Expo}(\beta) + \text{Expo}(c_i)).
\]
Because \(\beta \leq c_i\), \(\text{Expo}(c_i)\) is stochastically dominated by \(\text{Expo}(\beta)\). We can write
\[
\sum_{i=1}^{k} (C(v_i, X_{i-1}) - X_{i-1}) \geq \sum_{i=1}^{k} (\text{Expo}(\beta) + \text{Expo}(\beta)).
\]
The sum of \(2k\) independent exponential random variables has a gamma distribution \(\Gamma(2k, \beta)\). Therefore,
\[
Pr[X_{k-1} > t \mid S'_p] \leq \frac{1}{(2k)!} \int_{t}^{\infty} x^{2k-1} e^{-x} dx
\]
\[
\leq \frac{t^{2k-1} e^{-\beta t}}{(2k)!}.
\]
Putting all of this together, the lemma immediately follows. \(\square\)

The next lemma addresses the question of whether or not a vertex \(v\) is infected at time \(t\). Note that \(S_p\) only addresses whether or not a vertex was infected via \(p\) at some time before \(t\); it could be cured thereafter.

If \((X_i)_{i=1}^{k}\) is the canonical realization of \(p\), then we say that \(v\) is infected at time \(t\) via path \(p\) if \(S_p\) occurs and the first cure event at \(v_k\) after \(X_k\) does not occur until after time \(t\). We denote this event by \(T_{p,t} = S_p \cap \{C(v_k, X_k) > t\}\).
**Lemma 8.7.** Suppose \( p \) is a walk of length \( k \), and the amount of antidote at \( v_k \) is \( c_k \geq \beta \). Then,

\[
\Pr[\mathcal{T}_{p,t}] < e^{-\beta t/2} \left( 1 + \frac{1}{(2k)!} (t/2)^{2k-1} \right) \beta^k \prod_{j=0}^{k-1} \frac{1}{c_j}.
\]

**Proof.** We first note that the elapsed time from \( X_k \) to the cure event \( C(v_k, X_k) \) is an exponential random variable with parameter \( c_k \), independent of \( S_p \). Thus, we can write

\[
\Pr[\mathcal{T}_{p,t}] = \Pr[C(v_k, X_k) > t, S_p]
= \Pr[\text{Expo}(c_k) > t - X_k, S_p]
\leq \Pr[\text{Expo}(c_k) > t/2, S_p, X_k \leq t/2] + \Pr[X_k > t/2, S_p]
\leq \Pr[\text{Expo}(c_k) > t/2] \Pr[S_p] + \Pr[X_k > t/2, S_p].
\]

From the definition of the canonical end time \( Z_p \), if \( S_p \) occurs, then \( X_k = Z_p \). Therefore, we have

\[
\Pr[\mathcal{T}_{p,t}] \leq \Pr[\text{Expo}(c_k) > t/2] \Pr[S_p] + \Pr[Z_p > t/2, S_p].
\]

Using Lemma 8.6, we can write

\[
\Pr[Z_p > t/2, S_p] \leq \frac{1}{(2k)!} (t/2)^{2k-1} e^{-\beta t/2} \beta^k \prod_{j=0}^{k-1} \frac{1}{c_j}.
\]

Meanwhile, from the exponential distribution and Lemma 8.5,

\[
\Pr[\text{Expo}(c_k) > t/2] \Pr[S_p] \leq e^{-\alpha t/2} \beta^k \prod_{j=0}^{k-1} \frac{1}{c_j} \leq e^{-\beta t/2} \beta^k \prod_{j=0}^{k-1} \frac{1}{c_j}.
\]

This implies

\[
\Pr[\mathcal{T}_{p,t}] \leq \Pr[\text{Expo}(c_k) > t/2] \Pr[S_p] + \Pr[Z_p > t/2, S_p]
\leq e^{-\beta t/2} \left( 1 + \frac{1}{(2k)!} (t/2)^{2k-1} \right) \beta^k \prod_{j=0}^{k-1} \frac{1}{c_j}.
\]

\[\square\]
For a path \( p = (v_0, \ldots, v_k) \), we say that \( p \) is safe if \( c_{v_i} \geq d_{v_i} \) for \( 0 \leq i \leq k \).

We denote by \( \mathcal{P}_k \) the set of paths originating in \( S \) of length exactly \( k \), and we define \( \mathcal{P}_k' \) correspondingly for safe paths. We will prove the following lemma which states that the probability that an infection persists within the inoculated nodes can be made arbitrarily small by choosing an appropriate length of time:

**Lemma 8.8.** Suppose \( G \) is a contact graph on \( n \) nodes with seed set \( S \) of size \( s \). Then for any \( h > 0 \) and any infection rate \( \beta \), then

\[
\Pr \left[ \bigcup_{\pi \in \mathcal{P}'} \mathcal{T}_p \right] \leq \sum_{\pi \in \mathcal{P}'} \Pr[\mathcal{T}_p] \leq \frac{sh}{\beta(1 - \beta)}.
\]

if

\[
t \geq 8 \left( \frac{\log(1/h) + \log(2\beta)}{\min(\beta, \beta \log(1/\beta))} \right) = c \log(1/h) + c',
\]

where \( c \) and \( c' \) only depend on \( \beta \) and not on \( n \) or \( s \), and \( \mathcal{T}_p \) can represent any \( \mathcal{T}_{p,t} \) for such \( t \).

**Proof.** Our strategy is to analyze short paths and long paths separately. For long paths, we will use Lemma 8.5, and for short paths, we will use Lemma 8.7.

Building off our definitions of \( \mathcal{P}_k \) and \( \mathcal{P}_k' \), we define \( \mathcal{P}_{\geq k} = \bigcup_{j=k}^{\infty} \mathcal{P}_j \), and \( \mathcal{P}', \mathcal{P}_{< k}, \) and \( \mathcal{P}_{< k}' \) accordingly. Let \( k_0 \) be the cutoff between long and short paths to be determined later. For paths of length at least \( k_0 \), we observe that

\[
\sum_{p \in \mathcal{P}_{\geq k_0}} \Pr[\mathcal{T}_p] \leq \sum_{k=k_0}^{\infty} \sum_{p \in \mathcal{P}_k'} \Pr[S_p]
\]

\[
\leq \sum_{k=k_0}^{\infty} \sum_{v_0 \in S} \sum_{v_1 \sim v_0} \cdots \sum_{v_{k-1} \sim v_{k-2}} \Pr[S_{(v_0, \ldots, v_k)}]
\]

\[
\leq \sum_{k=k_0}^{\infty} \sum_{v_0 \in S} \sum_{v_1 \sim v_0} \cdots \sum_{v_{k-1} \sim v_{k-2}} \beta^k \prod_{j=0}^{k-1} \frac{1}{c_j}
\]

\[
\leq \sum_{k=k_0}^{\infty} \sum_{v_0 \in S} \sum_{v_1 \sim v_0} \cdots \sum_{v_{k-1} \sim v_{k-2}} \beta^k \prod_{j=0}^{k-1} \frac{1}{d_j}
\]

\[
= \frac{s \beta^{k_0}}{1 - \beta'}.
\]
On the other hand, for paths of length less than \( k_0 \), we have

\[
\sum_{p \in P'} \Pr[T_p] \leq \sum_{k=0}^{k_0-1} \sum_{\pi \in P_k'} \Pr[T_{\pi}]
\]

\[
\leq \sum_{k=0}^{k_0-1} \sum_{v_0 \in S} \sum_{v_1 \sim v_0} \ldots \sum_{v_{k-1} \sim v_{k-2}} e^{-\beta t/2} \left( 1 + \frac{1}{(2k)!} \right) \frac{(t/2)^{2k-1}}{(2k)!} \beta^k \prod_{j=0}^{k-1} \frac{1}{c_j}
\]

\[
\leq s \sum_{k=0}^{k_0-1} e^{-\beta t/2} \beta^k \left( 1 + \frac{(t/2)^{2k-1}}{(2k)!} \right)
\]

\[
\leq sk_0 e^{-\beta t/2} \left( 1 + \frac{(t/2)^{2k_0-1}}{(2k_0)!} \right).
\]

Combining the bounds for short and long paths yields

\[
\sum_{\pi \in P'} \Pr[T_{\pi}] \leq s \left( \frac{\beta^{k_0}}{1 - \beta} + k_0 e^{-\beta t/2} \left( 1 + \frac{(t/2)^{2k_0-1}}{(2k_0)!} \right) \right)
\]

We choose \( k_0 = \beta t/8 \). Because \( t \geq 8/\beta \), we can use Stirling’s approximation to derive the bound

\[
e^{\beta t/4} \geq k_0 + \frac{(t/2)^{2k_0}}{(2k_0)!}.
\]

Thus we have

\[
\sum_{p \in P'} \Pr[T_p] \leq s \left( \frac{\beta^{k_0}}{1 - \beta} + e^{-\beta t/4} \right)
\]

\[
\leq s \left( \frac{\beta^{t/8}}{1 - \beta} + e^{-\beta t/4} \right)
\]

\[
\leq \frac{sh}{\beta(1 - \beta)}
\]

by using the assumption \( t \geq 8 \left( \frac{\log(1/h) + \log(2\beta)}{\beta \log(1/\beta)} \right) \), \( \beta^{t/8} \leq \frac{sh}{2\beta(1 - \beta)} \). This completes the proof of Lemma 8.8. \( \square \)

### 8.3 Proofs of the main theorems

We are now ready to prove Theorem 8.1. Suppose that an infection starts in \( S \subseteq H \subseteq V \), and each node \( v \in H \) is inoculated with \( c_v = d_v \). Recall that \( \mathcal{E}_H \)
denotes the event that an infection started in $S$ ever leaves the set $H$. Theorem 8.1 states that $E_H$ satisfies
\[ \Pr[E_H] \leq \frac{s}{\beta} \Pr \left( 1 - \beta, \frac{1_S}{s} \right) 1^\top_{\bar{H}}. \]

**Proof.** (of Theorem 8.1) Let $B_k$ denote the set of all paths of length $k$ from $S$ to $\bar{H}$ such that the first $k - 1$ steps are in $H$. We define $B$ to be the union of all $B_k$. Note that if $u \in \bar{H}$ is ever infected, then $S_p$ occurs for some $p \in B$. We will bound that probability using the union bound:
\[
\sum_{p \in B} \Pr[S_p] \leq \sum_k \sum_{p \in B_k} \Pr[S_p] \\
\leq \sum_k \sum_{v_0 \in S} \sum_{v_k \in H} \sum_{p=(v_0, \ldots, v_k) \in B_k} \Pr[S_p] \\
\leq \sum_k \sum_{v_0 \in S} \sum_{v_k \in H} \sum_{p=(v_0, \ldots, v_k) \in B_k} \beta^k \prod_{j=0}^{k-1} \frac{1}{d_j} \\
= \sum_k 1_S \beta^k (D^{-1}A)^k 1^\top_{\bar{H}} \\
= \sum_k 1_S \beta^k W^k 1^\top_{\bar{H}} \\
= \frac{s}{\beta} \Pr \left( 1 - \beta, \frac{1_S}{s} \right)_{\bar{H}},
\]
where we use Equation (2.2), the assumption on $H$, and Lemma 8.5.

Note that if $p$ is a path that contains vertices in $\bar{H}$, then it has an initial segment $\bar{p} \in B$, and $S_p \subseteq S_{\bar{p}}$. The set of such walks is $\mathcal{P} \setminus \mathcal{P}^\prime$; we have shown that
\[
\Pr \left( \bigcup_{p \in \mathcal{P} \setminus \mathcal{P}^\prime} T_p \right) = \Pr \left( \bigcup_{p \in B} T_p \right) \\
\leq \sum_{p \in B} \Pr[S_p] \\
\leq \frac{s}{\beta} \Pr \left( 1 - \beta, \frac{1_S}{s} \right)_{\bar{H}}.
\]
Thus, we have shown that the probability that the infection leaves $H$ depends on the personalized PageRank on $\bar{H}$. □

We are now ready to prove our next theorem.
Proof. (of Theorem 8.2) By the assumptions, $H$ is a cluster with $S$ contained in its $(1 - \beta)$-core. Thus, for $u \in S$, we have from Equation (8.1):
\[
\Pr(1 - \beta, u)_H \leq \frac{h}{1 - \beta}.
\]
Summing over all $u \in S$ gives
\[
\Pr\left(1 - \beta, \frac{1_0}{s}\right)_H = \frac{1}{s} \sum_{u \in S} \Pr(1 - \beta, u)_H 
\leq \frac{h}{1 - \beta}.
\]
Applying this bound to Theorem 8.1 gives
\[
\Pr[\mathcal{E}_H] \leq \frac{sh}{\beta(1 - \beta)}.
\]
Thus, the probability that the infection escapes $H$ is at most $\frac{sh}{\beta(1 - \beta)}$. Because $c_i = d_i$ for $v_i \in H$, all paths within $H$ are safe paths, and we can apply Lemma 8.8 to bound the probability that the infection persists in $H$. Lemma 8.8 implies that the probability that the infection persists in $H$ for longer than $c \log(1/h) + c'$ time is also at most $\frac{sh}{\beta(1 - \beta)}$. Combining these two results, the probability that the infection persists anywhere on the contact graph for longer than $c \log s + c'$ time is at most $2 \frac{sh}{\beta(1 - \beta)}$.

Theorem 8.4 implies that if $S$ is chosen randomly from an $h$-cluster $H$, then there is a high probability that it is also in the core of $H$. This is important, because if $S$ is in the core of $H$, then we can effectively combat any infection starting from $S$ by only inoculating $H$. The proof is similar to the analysis involved in local partitioning algorithms using PageRank [8].

Proof. (of Theorem 8.4) Suppose we are given an $h$-cluster $H$, and $S$ is formed by selecting $s$ random vertices from $H$, independently with probability proportional to their degrees. Suppose $v$ is one of those $s$ nodes, and let $X$ be a random variable that marks the amount of personalized PageRank contained in $\bar{H}$:
\[
X = \Pr(\alpha, v)_{\bar{H}}.
\]
From [8], we have
\[ E[X] \leq \frac{h}{2\alpha}, \]
where the expectation is over the possible nodes \( v \in H \). Furthermore, since \( X \leq 1 \), we can bound the variance by \( \text{Var}(X) \leq E[X] \).

Here we take \( \alpha = 1 - \beta \). Since we are selecting \( s \) random vertices, we consider \( Y = \sum_{i=1}^{s} X_i \) where \( X_i \) is a copy of \( X \). We are interested in bounding
\[ \Pr \left( Y \geq \frac{sh}{\alpha} \right) \leq \Pr[Y \geq 2E[Y]]. \]

Using a standard Chernoff inequality (see [38]) and the known bound for \( E[X] \), we have
\[ \Pr[Y \geq 2sE[X]] \leq \exp \left( -\frac{sE[X]^2}{2\text{Var}(X)} \right) \leq \exp \left( -\frac{sh}{4\alpha} \right) \leq \epsilon \]
since \( sh/(4\alpha) \geq \log(1/\epsilon) \).

By Theorem 8.1, with probability at most \( 1 - \epsilon \), the event that the infection starting from \( S \) leaves \( H \) satisfies
\[ \Pr[\mathcal{E}_H] \leq \frac{s}{\beta} \Pr \left( 1 - \beta, \frac{1s}{s} \right)_H \leq \frac{sh}{4\alpha\beta} \leq \frac{sh}{4(1 - \beta)\beta} \leq \epsilon \]
by the assumption \( sh \leq \epsilon \). This completes the proof of Theorem 8.4.

8.4 Some questions for further study

There are many questions remaining, several of which we mention here:
1. In this chapter, we show that if $s$ infected nodes are in the core of a $h$-cluster $H$ and the product of $s$ and $h$ is small, then we only need to inoculate nodes in $H$ so that the infection will die out in $O(\log s)$ time with high probability. Is it possible to improve or replace the condition imposed on the product $sh$?

2. In our main theorems, our analysis involves the Cheeger ratio which is one of the parameters concerning the structure of a graph. It will be desirable if other structural parameters can help improve the probabilistic bounds in the statement of Theorem 2, for example.

3. In this chapter, we consider a fixed infection rate $\beta$ and ask how little antidote can be used while still ensuring the contact process dies out quickly. The other natural approach to this problem is to fix an amount of antidote and ask for what range of $\beta$ will the disease necessarily die out quickly.

4. One can also consider alternative models of contact process where cured nodes may or may not susceptible to reinfection. In addition, the type of propagation on networks can be different.

8.5 Acknowledgement

Material in this chapter appears in the following published article:

Chapter 9

Hypergraph Coloring Games and Voter Models

In this chapter, we consider a network coloring game, motivated by human behavioral experiments [85, 87] conducted in a network setting. The network coloring game can be formulated as the following interaction-based voter model:

A set of voters is modeled as the vertex set of a hypergraph $H = (V, E)$. Each hyperedge $g \in E$ represents a small, possibly overlapping group representing social interactions and discussions (such as lunchtime hallway or office discussions, blog commentary, television viewership, and Web forums). At time $t = 0$, each voter has a color representing an initial preference or is undecided. Then at each time step, one hyperedge $g$ is randomly selected according to some prescribed probability distribution on $E$. After the voters in $g$ interact with one another, the voters’ preferences can change probabilistically. This process is repeated for many rounds, and the coloring configuration of the voters evolves.

Many natural questions arise. Will the coloring configurations converge under certain conditions? If the coloring patterns diverge or oscillate, what would be an appropriate time frame to stop the model? When the model is run for some prescribed number of rounds, how does the observed coloring configuration behave? Can the observed coloring configuration be something other than ‘random’? What is the stationary distribution of the observed coloring configuration, if it exists?

The behavioral experiments of Kearns et al. [78, 79, 85, 86] were the original
motivation for our model. In these simulations, actual human agents were given a common graph coloring task with varying monetary rewards for their timely completion. This is a strategic component that is not necessarily built into our interaction-based model, but the probability distributions for color change can potentially be derived using game-theoretic analysis. The experiments allowed agents differing “views” of information about the specific graph problem structure. Thus, agents only had limited information and could only make decisions based on the color configurations of smaller subsets of nodes (but not just pairs). This gives a compelling reason to model the voters as nodes of a hypergraph and not a traditional graph where edges are limited to two endpoints.

Analysis of these more general voter models and coloring games have remained quite elusive in spite of extensive study either by simulation [78, 79, 85, 86] or rigorous analysis of other proposed voter model variations [47, 74, 107, 108] for various types of networks [128]. The results vary widely with the specific set of rules or dynamics and point to evidence of intrinsic difficulties involved in the general voting model.

Although the general voting and coloring problems may be intractable to analyze tightly, we will consider several special cases that will help in evaluating the more general case. The first is a special case of memoryless voter models. In a memoryless voter model, the voters do not consider their current preferences when formulating new ones. This may seem too restrictive at first, but the memoryless model lends itself to tractable mathematical analysis and great insight provides a stepping-stone to rigorous treatment of more general models. Under these assumptions, we can show that the dynamics of coloring configurations can be analyzed quite precisely in several aspects.

Our coloring game is played on a hypergraph where each hyperedge is formed by a group to model multi-party simultaneous interaction. Although the coloring configurations of the nodes in the hypergraph do not necessarily converge in general, we can model the rounds of coloring games by conducting random walks on the associated directed state graph which contains coloring configurations as nodes. It turns out, for memoryless interactions, the spectrum of the state graph
has elegant and well-defined expressions. Using the formula for the eigenvalues, we can determine the rate of convergence for random walks on the state graph, which can then be used for determining a cut-off time for simulating the voting game.

The eigenvectors of the random walk on the state graph contain rich information. In particular, the stationary distribution, which is the eigenvector associated with eigenvalue 1, tells us the distribution of the observed coloring configuration as time goes to infinity. In contrast with undirected graphs, the stationary distribution a random walk on a directed graph does not always exist and is not easy to determine, especially for state graphs with exponentially large size. Even for simple hypergraphs $H$ such as the path or cycle, the stationary distribution for random walks on the state graph have complicated forms quite different from the uniform distribution [36]. Nevertheless, we will show that for events such as “red wins by at least 5% within an error bound $\epsilon$” can be determined by simulating this voter model for $O(\log(1/\epsilon)m \log n)$ rounds, where $m$ is the number of hyperedges in $H$, provided that the hyperedges are always chosen uniformly at random, and the interactions are memoryless.

Next, we proceed to consider a partially memoryless model for which, with some probability $\beta$, the interaction process is memoryless and with probability $1 - \beta$, the process is not required to be memoryless. Specifically, a partially memoryless voter model is one whose associated state-graph random walk can be decomposed into two parts: $P = \beta M + (1 - \beta)P'$, where $M$ is a memoryless random walk and $P'$ is an unrestricted random walk. Using results from the memoryless case and also from general directed Laplacians [32], we can analyze partially memoryless voter games as well.

Even the partially memoryless condition might seem to be too strict to be satisfied by real-world social interactions, but it can still be of interest to serve as a type of benchmark for the sake of comparison. In particular, the partially memoryless case provides a natural framework for evaluating the more general interaction voter process. For a given interaction voter process with its associated arbitrary (memory-dependent) random walk $P$, we can construct a corresponding memoryless process $P' = \beta M + (1 - \beta)P$ where $M$ is a memoryless process. The
determination of $M$ depends on $\beta$ and a simulation of $P$ with an appropriate cut-off time depending on $\beta$. The details for choosing $\beta$ and $M$ will be given later in Section 9.7.

**Related work**

The coordination or consensus game has been extensively studied in evolutionary network game theory [56, 82, 134]. In a set of controlled behavioral experiments with human subjects, a simple voting game [79] as well as a biased version [85] were simulated on a set of small, constructed topologies. The rules varied, but involved small financial rewards for success and differing amounts of information visibility. The results showed that consensus was often reached within a certain timeframe, though there were many cases of failure as well. It is desirable to further explain these social network phenomena with rigorous modeling and analysis.

One method to analyze various consensus games is to use the standard voter model [47, 74]. In this model, there is a given network $G = (V, E)$ where the voters comprise the vertices $V$ and their social connections are represented by the edges $E$. The voter model is a Markov chain in which at each time step, one voter is selected randomly during each round, assuming the vote of a randomly-selected neighbor. Some empirical work has shown that its performance depends on network topology [127]. In more rigorous analysis, several proofs have been given [4, 133] using the dual Markov chain of coalescing random walks to show that the model results in unanimous agreement, and the expected time to reach consensus under the voter model is $O(n^3 \log n)$, where $n$ is the number of voters in the network. For the biased voting game, this model takes exponential time [87] to converge, but it can be used as a subroutine for an algorithm converging in $O(n^8 \log n)$ expected time. Of course, elections in practice rarely end with a unanimous decision, and our model will reflect this observation.

Other recent work consider alternative dynamics for the voting game, including Glauber dynamics [107] and pieces of “advice” [108] given to the voters. These models also result in unanimous consensus, and the time required for conver-
gence is related to complex graph-theoretic quantities such as the tilted cutwidth, diameter, and broadcast time, often difficult to compute or reason about. Some of these models assume that the voters know approximate values for these quantities, which our model avoids. These models again result in unanimous coordination, whereas our proposed model is more likely to result in more realistic voting configurations.

The behavioral experiments of Kearns et al. [79, 85] show that consensus is indeed not always reached, especially for biased voting games. However, not much has been determined thus far about the distribution of voting configurations should a true consensus be unattainable.

Furthermore, nearly all work done on the network coordination game has focused on pairwise relationships between voters. In practice, voters often interact in groups larger than two, and most of the models described so far have not taken this possibility into account. Such interactions as town-hall debate, reader comments on news articles, and dinner-table discussions can all possibly be considered in the framework of our hypergraph model.

9.1 The interaction model on hypergraphs

We consider a variation on the voter model that takes into account multi-voter interactions for the coordination game. Instead of limiting ourselves to pairwise relationships between voters, we model the interaction network as a hypergraph $H = (V, E)$, where each of $n$ vertices is a voter and each of $m$ hyperedges represents a group that can interact.

Our interaction model can be described as follows. At the beginning, all voters have some initial views, voting for one of several candidates, or starting undecided. In each round, one hyperedge $g$ is selected randomly, and an interaction $X_{g, \tau}$ takes place: the randomly selected hyperedge $g$ changes its voting pattern to $\tau$. In the most general case, the probability of $X_{g, \tau}$ occurring can depend on the current coloring configuration $\sigma$ of $H$. We denote by $p(g, \sigma, \tau)$ the probability of $X_{g, \tau}$ occurring when $\sigma$ is the current coloring configuration on $H$, and for all $\sigma$, it
must be the case that \( \sum_{g,\tau} p(g, \sigma, \tau) = 1. \)

If the interactions are memoryless, then the probability of \( X_{g,\tau} \) occurring is constant across all voting configurations \( \sigma \) on \( H \). In this case, we can omit \( \sigma \) and denote by \( p(g, \tau) \) the probability that \( X_{g,\tau} \) occurs. When this probability only has two parameters, it can be assumed that the interaction is memoryless. The model is then simulated for some preset number of rounds.

While the game is taking place, the state can be described as a voting configuration or coloring configuration of the voters among the candidates or undecided. (Note that we can interpret votes and colors and vice versa, so these terms will be used interchangeably throughout this chapter.) There are \( r \) possible votes (including undecided), and thus \( r^n \) possible configurations, and we can construct a state graph \( H^* \), where the vertices are coloring configurations, and a directed edge connects \( u \) to \( v \) if the state \( v \) is reachable from \( u \) in one round of the interaction model. Thus, a simulation of the coloring game using the interaction model can be completely described as a random walk on the state graph \( H^* \).

Let an event \( A \subset V(H^*) \) be a subset of the states in the state graph. These events can represent numerous scenarios: for example, the states where more than half of voters choose red, or the states which have a specific trend over some subsets of the voters, representing a voting district or municipality. In general, \( A \) can be any event of interest. We will show that or memoryless interactions, we can estimate \( \Pr[A] \) within a probabilistic error bound by using a sufficient number of samplings, as long as \( A \) contains enough states.

**Theorem 9.1.** Let \( A \) be an event or subset of all possible coloring configurations on a hypergraph \( H \) on \( n \) vertices and \( m \) edges. Suppose the interactions are memoryless, the probability of selecting a hyperedge \( g \) in any given round is at least \( 1/(\alpha m) \) for some \( \alpha \), and for any set \( S \subseteq V(H) \) there are at least \( |S| \) edges incident to vertices in \( H \). The probability \( \Pr[A] \) that \( A \) occurs at any cut-off time after \( O(\alpha m \log n) \) rounds of simulation can be estimated within an error bound of \( \epsilon \) using \( O(\alpha \log(1/\epsilon)m \log(n)/\Pr[A]) \) rounds of simulation.

Note that for the special case that the hyperedges are chosen uniformly at
random, the number of rounds of simulation for convergence is:

\[ O(\log(1/\epsilon) m \log(n) / \Pr[A]) \]

The main tools that we use to prove the above theorem are sampling and fast mixing of random walks on the state graph \( H^* \) associated with memoryless strategies. In particular, the spectrum of such random walks on the state graph \( H^* \) can be determined by using spectral techniques originating in the analysis of card shuffling [24, 25], self-organizing search [60], hyperplane arrangement [17] and semigroup random walks as well as the recent work on edge flipping games in graphs [36].

For the partially memoryless case, where the random walk \( P \) on the state graph \( H^* \) can be written as \( P = \beta M + (1 - \beta) P' \) with memoryless \( M \), we have the following result showing a trade-off between the convergence time and \( \beta \):

**Theorem 9.2.** Let \( A \) be an event or subset of all possible coloring configurations on a hypergraph \( H \) on \( n \) vertices and \( m \) edges. Suppose the interactions are partially memoryless with parameter \( \beta \), the probability of selecting a hyperedge \( g \) in any given round is at least \( 1/(\alpha m) \) for some \( \alpha \), and for any set \( S \subseteq V(H) \) there are at least \( |S| \) edges incident to vertices in \( H \). The probability \( \Pr[A] \) that \( A \) occurs at any cut-off time after \( O(\alpha m \log n) \) rounds of simulation can be estimated within an error bound of \( \epsilon \) using \( O\left(\frac{\alpha \log \frac{1}{m \log n}}{\beta \Pr[A]}\right) \) rounds of simulation.

Finally, for the most general case, where the interactions are not memoryless at all, it is well known to be very difficult to analyze exactly [36]. This is because the associated random walk can have exponentially small eigenvalues, and convergence time can vary widely based on the specific dynamics. Nevertheless, we will show how to use insight from the memoryless and partially memoryless models to reason about the general voter model. By choosing an appropriate damping constant \( \beta \) and extracting a memoryless version of a given process, we can then construct a partially memoryless process which can be used to approximate the given interactive process.
9.2 The voting game on a hypergraph as a random walk on the associated state graph

As stated in the previous section, the interaction model on a hypergraph $H = (V, E)$ can be analyzed as a random walk on a state graph $H^*$. The state graph $H^*$ is a weighted directed graph whose vertices are coloring configurations of $V(H)$. To distinguish from nodes in $H$, we sometimes call a vertex in $V(H^*)$ a state. There is a directed edge from a state $u$ in $H^*$ to another state $v$ if there is a hyperedge $g$ and a coloring pattern $\tau$ on $g$ such that the interaction $X_{g,\tau}$ moves $u$ to $v$. The weight on the edge $(u, v)$ is determined by the probability that $X_{g,\tau}$ occurs during state $u$. We denote by $p(g, u, \tau)$ the probability that, in state $u$, hyperedge $g$ is selected and changes its color configuration to $\tau$.

Note that for any directed graph with weighted edges, we can define a typical random walk with a transition from $u$ to $v$ occurring with probability $w(u, v)/\sum_z w(u, z)$. Therefore, the typical random walk associated with the state graph $H^*$ simulates the evolving configurations in the voter interaction game.

Starting from a coloring configuration $u$, a sequence of interactions

$$X_{g_1, \tau_1}X_{g_2, \tau_2}\cdots X_{g_t, \tau_t}$$

induces a series of changes in the coloring configuration and can then be viewed as a walk starting from $u$, traversing $t$ directed edges on the state graph $H^*$. This correspondence leads to the following lemma whose proof follows from the Perron-Frobenius Theorem (as shown in [35]).

**Lemma 9.3.** The voter interaction game with interactions as described above does not converge to an equilibrium in general. Instead, from any initial coloring configuration, the resulting configuration after $t$ rounds of simulation, is $s$ with probability approaching $\pi(s)$, where $\pi$ is the stationary distribution of the random walk on the state graph $H^*$, as long as $t$ is sufficiently large and $H^*$ is strongly connected and aperiodic.

An interaction $X_{g,\tau}$ is said to be nontrivial if the associated probability $p(g, \sigma, \tau)$ is nonzero. In the special case that all nontrivial strategies are consistent
with some coloring pattern \( \tau \) (for example, all red), then the coloring configuration will reach an equilibrium. (Note that this is an example of the special case where the interactions are memoryless.) Suppose there is a state \( s \) for which all the nontrivial interactions are of the form \( X_{g,s_g} \), where \( s_g \) is simply the coloring configuration of \( s \) restricted to nodes in \( g \). If this is the case, then starting from any initial configuration, the voting game will converge using standard coupon-collector probabilistic arguments:

**Lemma 9.4.** In the voter interaction game, suppose there exists a coloring configuration \( s \) such that all the nontrivial interactions are of the form \( X_{g,s_g} \), where \( s_g \) denotes the coloring pattern of \( s \) restricted to voters in \( g \). Starting from any initial configuration, the voting game converges to \( s \) after \( t \) rounds of simulation with probability at least \( 1 - e^{-c} \) if

\[
    t \geq \frac{\log n + c}{\min_{v \in V} \sum_{g \ni v} p(g, s_g)},
\]

where \( n \) is the number of voters and \( p(g, s_g) \) is the probability associated with the interaction. In the case that every vertex is incident to exactly \( d \) hyperedges and each hyperedge is chosen with equal probability, the above inequality is just \( t \geq n(\log n + c) \).

For the remainder of this chapter, we will assume that the given hypergraph and interaction dynamics yield a state graph \( H^* \) that is aperiodic and strongly connected. We will refer to the stationary distribution \( \pi \) accordingly.

We remark that Lemma 9.3 reduces the voter interaction game to the study of the associated random walk on the state graph, and the rate of convergence depends on the eigenvalues of the directed state graph. These values can be complex, and in the most general case, the spectral gap can be exponentially small. Nevertheless, we will consider memoryless interactions which allow us to have real eigenvalues for the state graph, and we can use these techniques to derive some bounds for partially memoryless interactions as well.
9.3 Memoryless interactions and semigroup spectral graph theory

For a random walk on the state graph $H^*$, we can describe a path in terms of the interactions $\{X_{g,\tau}\}$ that take place to follow the path. Thus, it is convenient to describe random walks as sequences of interactions. For a sequence $S = X_{g_1,\tau_1}X_{g_2,\tau_2}\ldots X_{g_t,\tau_t}$ and a state $u$, we say $S = u$ if the interaction game leads to state $u$ after following the path described by $S$. For two sequences, we say $S_1 = S_2$ if both paths end at the same state.

We say the nontrivial interactions $\{X_{g,\tau}\}$ are memoryless if the probability of choosing them does not depend on the current state. An equivalent definition is that for memoryless interactions, a repeated interaction $X_{g,\tau}$ means that earlier occurrences have no effect. Namely, for any three sequences of interactions $S_1, S_2, S_3$ of any length, then

$$S_1X_{g,\tau}S_2X_{g,\tau}S_3 = S_1X_{g,\tau}S_2S_3.$$ 

If the interaction strategies are memoryless, we can view them as members of a special type of semigroup known as a left-regular band or LRB, first studied in the 1940’s [91, 121]. An LRB is a semigroup where every element is idempotent, and for any two elements $x, y \in S$, $xyx = xy$. We define the product of two interactions $X_{g,\tau}$ and $X_{g',\tau'}$ to be the two interactions in sequence. If the interactions are memoryless, it is easy to see that the semigroup $S$ generated by all nontrivial interactions $X_{g,\tau}$ is an LRB. This allows us to apply techniques in [17, 24, 25, 36] to the voter interaction game. In particular, the associated random walk on the state graph for memoryless interactions has a clean form:

**Theorem 9.5.** Suppose that the voter interaction game on a hypergraph $H = (V, E)$ in $r$ colors has memoryless interactions $X_{g,\tau}$ for $g \in E$ and coloring patterns $\tau$ on $g$. If $p(g,\tau)$ is the probability of choosing $g$ and coloring voters in $g$ with the coloring pattern $\tau$, then the random walk on the associated state graph $H^*$ has an
eigenvalue $\lambda_T$ for every subset $T \subseteq V$:

$$\lambda_T = \sum_{g, \tau \subseteq T} p(g, \tau)$$

with multiplicity $(r - 1)^{n - |T|}$.

For the specific case where the hyperedges are selected uniformly at random and there are only two colors, we note that the eigenvalues have an even cleaner form: for each subset $T \subseteq V$, there is an eigenvalue:

$$\lambda_T = \frac{|\{g \in E \mid g \subseteq T\}|}{m}$$

with multiplicity 1.

To prove Theorem 9.5, we need to explore further properties of LRB semigroups. The proof of the corollary follows from Theorem A in [36], and the theorem follows by generalizing the techniques to $r$ colors. But in order to use these results, we must first interpret semigroup terminology in terms of the voter interaction game.

For any LRB semigroup $S$, there is a natural partial order defined by: $x \leq y \Leftrightarrow xy = y$. For our LRB where the members are sequences of interactions $X_{g, \tau}$, we define $x \leq y$ if the interaction or sequence of interactions represented by $x$ is irrelevant after performing the sequence $y$. A semilattice $L(S)$ can be defined on $S$ by considering the relation $\preceq$ on $S$ as follows: $y \preceq x \Leftrightarrow xy = x$. In other words, $y \preceq x$ if the for two sequences of interactions $x$ and $y$, the concatenation $xy$ leads to the same state as $x$ by itself. The equivalence class under $\preceq$ which contains $x$ is said to be the support of $x$, denoted by supp($x$), and for $x, y \in S$, supp($xy$) = supp($x$) $\cup$ supp($y$). The support of $x$ can be interpreted as the set of vertices whose colors were affected by the sequence of interactions given by $x$. The various elements of $L(S)$ are called the flats, and an element of $S$ is said to be a chamber if its support is maximal. Therefore, the chambers of $S$ are simply sequences of interactions $x$ that affect the entire set of nodes $V$.

As given in [24, 25, 36], the eigenvalues of a random walk on chambers have an elegant form. For each flat $X \in L(S)$, there is an eigenvalue $\lambda_X =$


\[ \sum_{x \in X} w_x. \] Here, \( w_x \) is the probability of selecting the semigroup member \( x \). In the voter interaction model, the chambers represent coloring configurations, and the probabilities \( w_x \) are simply the probabilities of choosing specific interactions, which are constant in the memoryless case. The flats are simply subsets of \( V \). The theorem statement about the eigenvalues of the random walk on \( H^* \) follow from this interpretation of the flats of the LRB semigroup induced by the memoryless interactions. For the multiplicities, we note from [36] that the multiplicity of \( m_X \) of \( \lambda_X \) satisfies

\[ \sum_{Y \geq X} m_Y = c_X, \]

where \( c_Y \) is the cardinality of \( S_{\geq Y} = S_{\geq y} = \{ z \in S : z \geq y \} \), where \( y \) is any element with support \( Y \). (The cardinality is independent of the choice of \( y \).) Translating the semigroup terminology into the setting of the interaction model shows how the multiplicities given in Theorem 9.5 were derived. We note that these techniques as used in [36] were developed for an edge-flipping game with two colors. The semigroup techniques used generalize to \( r \) colors, and the full proof for Theorem 9.5 can be derived by this simple extension. Further details about LRB semigroups and their terminology can be found in [24].

9.4 The cut-off time for voter interaction games

Our methods also address some of the questions that arise in the recent human network experiments of Kearns et al. [87]. The voters are given a hard deadline, often arbitrarily set by various entities without rigorous justification. Furthermore, for different stopping times, the outcome of the network experiments varied widely from consensus to chaos. Using our interaction model and spectral techniques, we will prove the following theorems about the interaction model’s convergence properties, as well as a mathematical interpretation of the resulting voting configuration after convergence is reached.

For voter interaction games with memoryless interactions, we have the following theorem. The proof follows by using the spectrum of \( H^* \), as derived in the previous section, to bound the total variation distance between the random walk
and its stationary distribution after \( t \) steps:

**Theorem 9.6.** Suppose the interaction model is simulated on a hypergraph \( H = (V, E) \) with \( |V| = n \), and each voter in \( V \) is colored with one of \( r \) colors. If the interactions are memoryless, then the total variation distance between the random walk on the state graph \( H^* \) denoted by the transition probability matrix \( P \) after \( t \) rounds of simulation and its stationary distribution \( \pi \) is given by

\[
\left\| P^t - \pi \right\|_{TV} = \max_{A \subseteq V} \max_y \left| \sum_{x \in A} P^t(y, x) - \pi(x) \right|
\leq \sum_{T \subseteq H} \lambda_T^t (r-1)^{n-|T|}.
\]

We remark that the above bound can be somewhat improved by restricting \( T \) to be the co-maximal subsets, although asymptotically the bound is still the same. Using this bound on the total variation distance, we can derive the convergence time for the interaction model:

**Theorem 9.7.** On a hypergraph \( H = (V, E) \) with \( |V| = n, |E| = m \), suppose the voter interaction model is simulated with memoryless interactions, and the probability of choosing a hyperedge \( g \) in any round is uniform over \( E \). Suppose for any set \( S \) of \( k \) voters, there are at least \( k \) hyperedges involving voters in \( S \). The random walk on the state graph \( H^* \) converges to its stationary distribution in \( O(m \log n) \) steps.

To prove Theorem 9.7, we use the derived spectrum from Theorem 9.5 in our derived bound on the total variation distance from Theorem 9.6.

**Proof.** Using Theorems 9.5 and 9.6:

\[
\left\| P^t - \pi \right\|_{TV} \leq \sum_{T \subseteq H} \lambda_T^t (r-1)^{n-|T|}
\]

\[
= \sum_{T \subseteq V} \left( \frac{|\{g \in E \mid g \subseteq T\}|}{m} \right)^t (r-1)^{n-|T|}
\]

\[
\leq \sum_{k=1}^{n} \left( \frac{\max_{|T|=k} |\{g \in E \mid g \subseteq T\}|}{m} \right)^t \binom{n}{k} (r-1)^{n-k}.
\]
Here, we indexed the subsets $T \subseteq V$ by their sizes.

For any node set $T$ of size $k$, we can upper-bound the number of hyperedges contained within $T$. By using the fact that for any set $T$ with $|T| = k$, there are at least $k$ hyperedges incident to nodes in $T$, we have:

$$\|P^t - \pi\|_{TV} \leq \sum_{k=1}^{n} \left(1 - \frac{k}{m}\right)^t \binom{n}{k} 2^{n-k}$$

$$\leq n^2 \left(1 - \frac{1}{m}\right)^t$$

$$\leq e^{-c}$$

since for $f(k) = \left(1 - \frac{k}{m}\right)^t \binom{n}{k} 2^{n-k}$, we have $f(k) \geq f(k + 1)$ for $k \geq 1$ and $t > 2m \log n + cn$. The theorem is proved.

Theorems 9.6 and 9.7 imply a method for choosing a stopping point for the interaction model: enough time for a desired level of convergence to a stationary distribution $\pi$. This also indicates what the voting configuration among the agents looks like at any time after it has converged: the votes are a random sample from $\pi$ from all the voting configurations in the state graph.

### 9.5 Estimating the probability of a given event

Although estimating individual components of $\pi$ can be computationally intractable, for memoryless interactions, we can effectively use sampling to estimate the probability of an event $A$, as long as the event has enough probability mass. For a general $\pi$, it can be difficult to reason about its components, since it contains exponentially many elements, corresponding to the state graph. Even for simple graphs such as the path of length $k$, the exact stationary distribution on the state graph is quite complex [36]. Additionally, because most of the components of $\pi$ are exponentially small, even estimating $\pi$ can be quite difficult. But in practice, the exact stationary distribution is not of utmost importance. Instead, it is much more revealing and tractable to reason about events that capture a larger portion of $\pi$. We will use the following fact:
**Theorem 9.8.** ([109]). Let $A$ be an event and $\pi(A) = \sum_{s \in A} \pi(s)$ be the probability that the outcome is in $A$. Let $\delta, \epsilon \in (0, 1)$. Suppose that after $N$ samplings, $X$ is the proportion of times the outcome was in $A$. Then

$$\Pr[\{(1 - \delta)\pi(A) \leq X \leq (1 + \delta)\pi(A)\}] \leq 1 - \epsilon,$$

as long as $N \geq O\left(\frac{\log(1/\epsilon)}{\pi(A)c(\delta)}\right)$, where $c(\delta)$ only depends on $\delta$.

We can use this bound to prove Theorem 9.1. Here, we consider the case where each hyperedge is chosen with uniform probability $1/m$, but the same argument will hold using the looser $1/(am)$ bound.

**Proof.** (for Theorem 9.1) For an event $A$, suppose we are given an initial state $f$ which we denote as a row vector indexed by states in $V(H^*)$. For any integer $t$, the coloring configuration we observe after $t$ rounds of the voter interaction model is in $A$ with probability

$$E_t[fA] = \sum_{x \in A} fP^t(x)$$

where $P$ denotes the transition probability matrix of the random walk on $H^*$.

By combining Theorems 9.5 and 9.6, we have

$$|E_t[A] - \pi(A)| \leq \max_y \left| \sum_{x \in A} P^t(y, x) - \pi(x) \right| = \left| \left\| P^t - \pi \right\|_{TV} \right| \leq \sum_{T \subseteq V} \lambda_T^t (r - 1)^{n-|T|}.$$

For the case where each hyperedge is chosen with probability $1/m$, where $m$ is the total number of hyperedges, we have

$$|E_t[A] - \pi(A)| \leq n^2 \left(1 - \frac{1}{m}\right)^t \leq \epsilon$$

if $t > 2m(\log n + \log(1/\epsilon))$. 

Now we use Theorem 9.8, by breaking up the voter interaction game into $N$ phases where $N = O\left(\frac{\log(1/\epsilon)}{\pi(A) \cdot \epsilon^{(\delta)}}\right)$ and each phase consists of $t = O(m \log n)$ rounds. The proportion of phases where the outcome is in $A$ satisfies:

$$\Pr[(1 - \delta)\pi(A) < |X - \pi(A)|] < (1 + \delta)\pi(A) \geq 1 - 2\epsilon. \quad (9.1)$$

9.6 The interaction model with partially memoryless interactions

We say that a set of interactions is partially memoryless if the random walk on the state graph $H^*$ can be decomposed into two parts, one of which is memoryless. Specifically, if the random walk transition matrix is $P$, there is a $\beta \in (0, 1)$ such that we can write

$$P = \beta M + (1 - \beta)P',$$

where the interactions described by $M$ are memoryless and $P'$ is another transition probability matrix without any restrictions.

One way a partially memoryless interaction model can arise is if at each step, the voters in the selected hyperedge interact without memory with some probability $\beta$, allowing their actions to depend on the current state with probability $1 - \beta$. It is also possible that the transition probability matrix $P'$ is not explicitly built this way, but it can nevertheless be expressed as such a partially memoryless model. In this sense, the parameter $\beta$ can be viewed as describing how memoryless the model is, and its properties will depend on $\beta$:

**Theorem 9.9.** On a hypergraph $H = (V, E)$ with $|V| = n, |E| = m$, suppose the voter interaction model is simulated with partially memoryless interactions, and the probability of choosing a hyperedge $g$ in any round is uniform over $E$. Suppose for any set $S$ of $k$ voters, there are at least $k$ hyperedges involving voters in $S$. The random walk on the state graph $H^*$ converges to its stationary distribution in $O\left(\frac{m \log n}{\beta}\right)$ steps.
The partially memoryless structure allows us to use the results from the fully
memoryless case, with an additional factor of $1/\beta$ when analyzing the spectrum.

Proof. Because the random walk on $H^*$ is given by memoryless strategies, we can
write its transition matrix as $\beta P_1 + (1 - \beta)P_2$ for memoryless $P_1$. Theorem 9.5
allows us to analyze the eigenvalues of $P_1$; we can use results from [32] to see that
$P_2$ has all eigenvalues between 0 and 1. Thus, if $\lambda_T$ is an eigenvalue of $P_1$, then
there is a corresponding eigenvalue of $P = \beta P_1 + (1 - \beta)P_2$ satisfying:

$$\lambda \leq \beta \lambda_T + (1 - \beta).$$

Using Theorem 9.6, we have:

$$|E_t[A] - \pi(A)| \leq \max_y \left| \sum_{x \in A} P^t(y, x) - \pi(x) \right|$$

$$= \|P^t - \pi\|_{TV}$$

$$\leq \sum_{T \subseteq V} \lambda_T^t (r - 1)^{n-|T|}$$

$$\leq \sum_{k=1}^{n} \left( 1 - \frac{\beta}{m} \right)^t \binom{n}{k} (r - 1)^{n-k}$$

$$\leq n^2 \left( 1 - \frac{\beta}{m} \right)^t$$

$$\leq e^{-c}$$

if

$$t \geq \frac{2m \log n + cn}{\beta}. \quad (9.2)$$

With this result, we can prove Theorem 9.2.

Proof. (for Theorem 9.2) For an event $A$, suppose we are given an initial state $f$
which we denote as a row vector indexed by states in $V(H^*)$. For any integer $t$,
the coloring configuration we observe after $t$ rounds of the voter interaction model
is in $A$ with probability

$$E_t[fA] = \sum_{x \in A} f P^t(x)$$
where $P$ denotes the transition probability matrix of the random walk on $H^*$.

By combining Theorems 9.5 and 9.6, we have

$$|E_t[A] - \pi(A)| \leq \max_y \left| \sum_{x \in A} P^t(y, x) - \pi(x) \right|$$

$$= \left| P^t - \pi \right|_{TV}$$

$$\leq \sum_{T \subseteq V} \lambda_T^t (r - 1)^{n - |T|}.$$

For the case where each hyperedge is chosen with probability $1/m$, where $m$ is the total number of hyperedges, we have

$$|E_t[A] - \pi(A)| \leq n^2 \left( 1 - \frac{\beta}{m} \right)^t$$

$$\leq \epsilon$$

if $t > \frac{2m}{\beta} (\log n + \log(1/\epsilon))$.

Now we use Theorem 9.8, similar to the proof of Theorem 9.1, by breaking up the voter interaction game into $N$ phases where $N = O \left( \frac{\log(1/\epsilon)}{\pi(A)c(\delta)} \right)$ and each phase consists of $t = O \left( \frac{m \log n}{\beta} \right)$ rounds (from (9.2)). We have the same bound as (9.1); the proportion of phases where the outcome is in $A$ satisfies:

$$\Pr[(1 - \delta)\pi(A) < |X - \pi(A)|] < (1 + \delta)\pi(A)] \geq 1 - 2\epsilon.$$  

9.7 The general interaction model

Thus far, we have written about the interaction model with memoryless and partially memoryless interactions. The general version of the interaction model concerns interactions that all can depend on the current and previous states, where the random walk on the state graph $H^*$ can be quite difficult to analyze. The eigenvalues of the random walk can be real or complex, and exponentially small [36]. With $O(2^n)$ states, computing the spectrum explicitly is too expensive or infeasible. It is even too difficult just to estimate the eigenvalues. Nevertheless, we can
use the previous memoryless and partially memoryless models as a basis for comparison between a specific set of state-dependent interactions and memorylessness in general.

In particular, for any general interaction voter game, we can build a companion game which is partially memoryless with one scalar parameter $\beta$. Let $M$ denote the transition probability matrix of the random walk on $H^*$. We wish to construct a partially memoryless model

$$P' = \beta M + (1 - \beta)P$$

where $M$ is memoryless and can be constructed from $P$ and $\beta$ by simulation as follows.

The main idea is to simulate the general model $M$ for some number of rounds, building a memoryless model using the simulation outcome. To know how long to simulate, we refer to Theorem 9.2 which provides an upper bound on the convergence time to the stationary distribution for partially memoryless interactions. If the companion model serves as an approximation for the actual voter game, it is enough to simulate the more general model $O(\frac{m}{\beta} \log n)$ time. To construct $M$, we collect a sample probability distribution: when a hyperedge $g$ is selected, keep track of the resulting coloring configurations. Then we can use these samples to build a partially memoryless model: first select a hyperedge $g$, then with probability $\beta$, randomly select one of the collected sample coloring configurations on $g$. Note that once the samples have been collected, this step is memoryless. With probability $1 - \beta$, we just use the original memory-dependent dynamics. The problem of determining the appropriate value of $\beta$ can be quite difficult since the required length of time until convergence may be exponential. A feasible heuristic approach is a combination of a series of iteration and simulation. Such process also can be used to get a sense of how “memoryless” the more general model really is. In other words, an alternative interpretation of the damping constant $\beta$ is just the ratio of the rate of convergence of the memoryless random walk and the actual random walk. In general, the value of $\beta$ can range from 1 to some exponentially small values, reflecting the fact that some models have higher extent of memorylessness than others. So even though the general model is notoriously
difficult to reason about, we can still quantify its convergence time empirically by comparing with the partially memoryless model.

To illustrate this comparative process, we consider a traditional voting game on networks. We are given a graph $G$, and each node has a starting color. Then at each time step, a node is randomly selected, and it takes the color of one of its neighbors, selected uniformly at random. In our setting, there is a hyperedge for every node $v$, consisting of $v$ and its neighbors, and whenever it is chosen, $v$ changes color to one of its neighbors’ colors. It should be clear that these dynamics are completely memory-dependent: the coloring configuration at time $t + 1$ always depends on the state at time $t$. But we can build a partially memoryless process using the method described in the previous paragraph.

We demonstrate this method using Zachary’s karate network [135] as an example graph. This graph has 34 nodes, and we initially assign one of 9 colors randomly to each node. Two sample runs of the consensus game are shown in Figures 9.1a and 9.1b. Each row represents one node, and the colors change as time moves from left to right.

We note that the state graph can be as large as $9^{34}$ nodes. The convergence bound for a memoryless game of similar size is about $O(m \log n)$ steps with $n = 34$ and $m = \binom{34}{2}$. This is comparable to the time limit of 1000 steps, if each step is taken to be in the range of a fraction of a second. But in our simulations, consensus may or may not be reached.

In Figure 9.1c, we give an illustration of a partially memoryless version of the consensus game. Using only 100 rounds of simulation, we built the partially memoryless version and simulated it for 1000 rounds. In practice, $\beta$ can be chosen empirically, by using binary search on $(0, 1)$. For Figure 9.1c, we chose $\beta = 0.01$. One way to choose $\beta$ is by iteratively adjusting $\beta$ so that the proportion of cases that achieve consensus reaches the range of what is to be expected.

### 9.8 Acknowledgement

Material in this chapter is published in the following article:
(a) One simulation of the consensus game on Zachary’s karate network for 1000 steps.

(b) Another simulation of the consensus game on Zachary’s karate network for 1000 steps.

(c) Simulation of the partially memoryless version of the consensus game on Zachary’s karate network for 1000 steps, with $\beta = 0.01$ and 100 steps of training.

**Figure 9.1:** Simulations of the consensus game and its partially memoryless approximation.
Bibliography


