Title
Ranking and sparsifying edges of a graph

Permalink
https://escholarship.org/uc/item/32p7m8xh

Author
Zhao, Wenbo

Publication Date
2012

Peer reviewed|Thesis/dissertation
UNIVERSITY OF CALIFORNIA, SAN DIEGO

Ranking and Sparsifying Edges of a Graph

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

in

Computer Science

by

Wenbo Zhao

Committee in charge:

Professor Fan Chung Graham, Chair
Professor Chung-Kuan Cheng
Professor Ronald Graham
Professor Daniele Micciancio
Professor Jason Schweinsberg

2012
The dissertation of Wenbo Zhao 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 my parents, Jinwang Zhao and Youhua Chen
To my lovely wife, Yu Luo
# TABLE OF CONTENTS

Signature Page .................................................. iii

Dedication ......................................................... iv

Table of Contents ................................................ v

Acknowledgements ................................................ vii

Vita ................................................................. ix

Publications ......................................................... ix

Abstract of the Dissertation .................................... xii

Chapter 1  Introduction ......................................... 1
  1.1  History .................................................... 3
  1.2  Summaries of Our Contributions ......................... 6

Chapter 2  Random Walks and PageRank Vectors on a Graph .. 9
  2.1  Random Walks on an Undirected Graph .................. 9
  2.2  PageRank Vectors on an Undirected Graph .............. 11
  2.3  PageRank Approximate Algorithms ....................... 13
  2.4  A Sharp PageRank Approximation Algorithm ............. 15
  2.5  Acknowledgement ......................................... 16

Chapter 3  Ranking and Sparsifying Edges of a Graph by PageRank . . 17
  3.1  The PageRank and Green Values for Edges ............... 19
  3.2  Graph Sparsification Using Green Values ............... 25
  3.3  Graph Sparsification Using Approximate Green Values .. 36
  3.4  Graph Partitioning using Approximate PageRank Vectors 42
  3.5  Acknowledgement ......................................... 44

Chapter 4  Connection Graphs .................................. 45
  4.1  The Definition of a Connection Graph ................. 46
  4.2  The Connection Laplacian of a Connection Graph ...... 47
  4.3  The Consistency of a Connection Graph ................. 50
  4.4  Random walks on a Connection graph ................... 53
  4.5  Acknowledgement ......................................... 54
ACKNOWLEDGEMENTS

First and foremost I express my sincerest gratitude to my supervisor, Prof. Fan Chung Graham, who has supported me throughout my Ph.D study with her patience, enthusiasm and knowledge whilst provided me room to work in my own way. She always inspired me with her brilliant ideas and helped me to explore my potential abilities to solve challenging problems. Her passion in her work and her drive have strongly influenced me. I appreciated her supervision and what I have learned from her will benefit my whole life.

I am particularly thankful to Prof. Ronald Graham and Prof. Ding-Zhu Du who fostered my research interests in attacking hard problems, shared their knowledge and ideas without any reservation, and encouraged me to pursue my Ph.D in UCSD. Their vigor, wisdom and methodology used in their research have shown me how to achieve my Ph.D career successfully. Thanks to Prof. Chung-Kuan Cheng, who has shared with me many interesting combinatorial optimization problems and had discussions with me patiently and enthusiastically. He also made great efforts to give me encouragement and provide support for my work. More thanks go to my committee members Prof. Jason Schweinsberg and Prof. Daniele Micciancio for their valuable advices for my research work and their help to improve the quality of my dissertation.

Moreover, many thanks go to my colleagues and friends Peng Du, Xiang Hu, Alexander Tsiatas, Wensong Xu, Nan Zang, and Keliang Zhao. It has been my great pleasure to work with them.

Finally, I owe my deepest gratitude to my parents. My father Jinwang Zhao and my mother Youhua Chen who have been supporting me with their best efforts in numerous ways for years.

The very last one goes to my lovely wife, Yu Luo. She has relocated from London to be with me during my Ph.D study. She has always been extensively supportive and brightening my life in the stressful time.

This dissertation is dedicated to all of them.

Chapters 2 and 3 contain material that appeared in the following published article:

Chapters 4, 5, 6 and 7 contain material that appears in the following article, accepted for publication:

## VITA

<table>
<thead>
<tr>
<th>Year</th>
<th>Degree</th>
<th>Institution</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000-2004</td>
<td>B. Sc. in Information and Computational Sciences</td>
<td>Nanjing University China</td>
<td>China</td>
</tr>
<tr>
<td>2004-2007</td>
<td>M. Sc. in Fundamental Mathematics</td>
<td>Chinese Academy of Sciences, Beijing</td>
<td>China</td>
</tr>
<tr>
<td>2007-2012</td>
<td>Ph. D. in Computer Science</td>
<td>University of California, San Diego</td>
<td>USA</td>
</tr>
</tbody>
</table>

## PUBLICATIONS


Many problems of practical interest can be represented by graphs. In practice, each edge of a graph is associated with some scalar weight denoting the similarity or the relevance between two vertices. However, these edge weights might not directly reflect the relative “importance” of edges in maintaining the structure of graphs. The study of edge ranking of graphs is to determine the graph edges by the relative importance under various criteria and is therefore essential for our fundamental understanding of graphs. In this thesis, we study various methods to determine the relative importance of the edges under several graphs models. Then we will use the edge ranking to examine two interrelated problems – graph sparsification and graph partitioning.
First, we study the edge ranking problem in the (usual) graph. We use PageRank vectors to define the edge ranking and give an improved algorithm for computing approximate (personalized) PageRank vectors on a graph of $n$ vertices with tight error bounds which can be as small as $O\left(\frac{1}{n^p}\right)$ for any fixed positive integer $p$. The improved PageRank algorithm is crucial for computing the quantitative ranking of edges in a given graph.

Our graph sparsification algorithm samples edges of a given graph with probabilities proportional to the edge ranking defined by PageRank. It can be used as a preprocess for graph partitioning. The combination of the graph sparsification and the partitioning algorithms using PageRank vectors leads an improved partitioning algorithm.

Next, we consider the edge ranking and sparsification in connection graphs. Connection graphs arise in dealing with high-dimensional data sets in which each edge is associated with both a scalar edge weight and a $d$-dimensional linear transformation. We generalized the PageRank and the effective resistance in the usual graph to their vectorized versions in connection graphs. They can be used as basic tools for organizing and analyzing complex data sets. For example, the generalized PageRank and effective resistance can be utilized to derive and modify diffusion distances for vector diffusion maps in data and image processing. Furthermore, the edge ranking of connection graphs determined by the vectorized PageRank and effective resistance are an essential part of sparsification algorithms which simplify and preserve the global structure of connection graphs.

Finally, we further explore the use of the vectorized version of PageRank vector in measuring the (local) consistency or inconsistency of a small area around a vertex in the connection graph. Since a PageRank vector can also be expressed as a geometric sum of random walks with different number of steps, we use the PageRank to define the (local) inconsistency coefficient for measuring the the portion of “probability” vanished in the local random walk process. Furthermore, we develop an algorithm by using vectorized version of PageRank for finding the local consistent area around a vertex.
Chapter 1

Introduction

A popular way to describe the affinities between data points is using a weighted graph, whose vertices correspond to data points, whose edges connect data points with large enough affinities and whose weights quantify the affinities. For example, Facebook’s friends recommendation system is based on users’ contacts in the social graph, in which vertices represent users, edges represent contacts and edge weights represent the frequency of contacts between users. There are all sorts of information networks which can also be built and examined by weighted graphs, such as biological networks built from biological databases and collaboration networks formed by authors writing joint papers, as well as various types of physical networks. However the edge weights in the graph might not directly reflect the relative “importance” of edges in maintaining the global structure of graphs. The study of edge ranking of graphs is to determine graph edges by the relative importance under various criteria and is therefore essential for our fundamental understanding of graphs. In this thesis, we study various methods to determine the relative importance of the edges under several graphs models. Then we will use the edge ranking to examine two interrelated problems – graph sparsification and graph partitioning.

Graph sparsification is an important and well studied problem on graphs. The goal of sparsification is to approximate a given graph $G$ by a sparse graph $\tilde{G}$ on the same set of vertices. If $\tilde{G}$ is close to $G$ in some appropriate metric, then $\tilde{G}$ can be used as a proxy for $G$ in computations within reasonable error bounds.
Since $\tilde{G}$ has fewer edges, computation and storage of $\tilde{G}$ then are cheaper. The appropriate metric usually refers to that every cut in the sparse graph is close to the corresponding cut in the given graph.

Graph partitioning, is another important and well studied problem on graphs. For given massive data sets, it is often desirable to group the data into more manageable segments. The graph partitioning is to organize the vertices 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 application of graph partitioning include clustering people who are closely related in the social network and image segmentation in computer vision.

Edge ranking, graph sparsification and graph partitioning are inherently related: The heart of the graph sparsification algorithms is the sampling techniques for randomly selecting edges according to their edge ranking; graph sparsification can then be used as a preprocess of graph partitioning for the purpose of reducing its complexity. We study these problems not just in the usual graphs but also in the connection graphs.

Connection graphs arise in numerous applications, in particular for data and image processing involving high-dimensional data sets. To quantify the affinities between two data points, it is often not enough to use only a scalar edge weight. For example, if the high-dimensional data set can be represented or approximated by a low-dimensional manifold, the patterns associated with nearby data points are likely to related by certain rotations [StWss]. As a general framework for data analysis and manifold learning, connection graphs are useful tools for measuring alignment between objects such as one-dimensional periodic signals, two-dimensional images and three-dimensional shapes, and then can serve as basic platforms for performing objects partitioning and clustering.
1.1 History

1.1.1 Edge Ranking and PageRank in Graphs

The study of edge ranking of graphs is to determine the graph edges by the relative importance under various criteria. If the criteria is the importance of an edge in preserving cuts of a graph, the value of the minimum cut across that edge is suggested by Benczúr and Karger [BK96, Kar94a, Kar94b, Kar00]; if the criteria is the importance of an edge in preserving the spectrum of a graph, the effective resistance or the commute time between the endpoints of an edge will be more suitable for such a purpose [SS08].

Another main tool for ranking graph edges by relative importance is PageRank. PageRank, which was first introduced by Brin and Page [BP98], is at the heart of Google’s web searching algorithms. Originally, PageRank was defined for the Web graph (which has all webpages as vertices and hyperlinks as edges). For any given graph, PageRank is well-defined and can be used for capturing quantitative correlations between pairs of vertices as well as pairs of subsets of vertices.

The exact computation of PageRank vectors is computationally intractable for extremely large graphs such as the web graph. However, an approximate PageRank is sufficient for most applications and also can be efficiently computed (see [ACL06, AP09, Ber06, Hav03, JW03]). The running time of the approximation algorithm in [ACL06] for computing a PageRank vector within an error bound of $\epsilon$ is basically $O\left(\frac{1}{\epsilon}\right)$. This improves the previous approach proposed by Berkhin [Ber06] by a logarithmic factor, in which a heap or priority queue is replaced by a FIFO queue [ACL06].

1.1.2 Graph Sparsification

To illustrate the usage of edge ranking, we examine a basic problem on graph sparsification. Graph sparsification was first introduced by Benczúr and Karger [BK96, Kar94a, Kar94b, Kar00] for approximately solving various network design problems. The heart of the graph sparsification algorithms is the sampling technique for randomly selecting edges. The goal is to approximate a given graph $G$
with \( m \) edges on \( n \) vertices by a sparse graph \( \tilde{G} \), called sparsifier, with \( O(n \log(n)) \) edges (or fewer) on the same set of vertices in such a way that every cut in the sparsifier \( \tilde{G} \) is within a factor of \( 1 \pm \epsilon \) of the corresponding cut in \( G \) for some constant \( \epsilon \in (0, 1] \). It was shown that

\[
\forall x \in \{0, 1\}^n, (1 - \epsilon)x^T Lx \leq x^T \tilde{L}x \leq (1 + \epsilon)x^T Lx
\]

where \( L \) and \( \tilde{L} \) are the Laplacians of graph \( G \) and its sparsifier \( \tilde{G} \), respectively.

Spielman and Teng [ST04] devised a sampling scheme to construct a spectral sparsifier with \( O(n \log^c(n)) \) edges for some large constant \( c > 0 \) in \( \tilde{O}(m) \) time. A spectral sparsifier \( \tilde{G} \) for graph \( G \) is a sparsifier satisfying

\[
\forall x \in \mathbb{R}^n, (1 - \epsilon)x^T Lx \leq x^T \tilde{L}x \leq (1 + \epsilon)x^T Lx.
\]

In [SS08], Spielman and Srivastava gave a different sampling scheme using the effective resistances of electrical networks to construct an improved spectral sparsifier with only \( O(n \log(n)) \) edges. In the process for constructing such a spectral sparsifier, they need to use the Spielman-Teng solver [ST04, ST06] as subroutines for solving \( O(\log(n)) \) linear systems. The running time of their sparsification algorithm is mainly dominated by the running time of Spielman-Teng solver which is \( O(m \log^c(n)) \) for a very large constant \( c \) [ST04, ST06]. It has been improved to \( O(m \log^2(n) \log^2(n)) \) by Koutis, Miller, and Peng [KMP10] very recently. Later on, Batson, Spielman and Srivastava [BSS09] gave an elegant construction for a spectral sparsifier with a linear number of edges although the running time is \( O(n^3 m) \).

### 1.1.3 Graph Partitioning

For graph partitioning algorithms, previously widely used approach is the recursive spectral method which finds a balanced cut in a graph on \( n \) vertices with running time \( \tilde{O}\left(\frac{n^2}{\lambda}\right) \) (see [Spi96]), together with an approximation guarantee within a quadratic root of the optimal conductance where \( \lambda \) denotes the spectral gap of the normalized Laplacian. The running time can be further improved to \( \tilde{O}(n^2) \) by using Spielman-Teng solver for linear systems [ST04, ST06, KMP10].
Another approach for the balanced cut problem is by using commodity flows [AK07, OSVV08]. In [AK07], the approximation guarantee is within a factor of $O\left(\log(n)\right)$ of the optimal, which was further reduced to $O\left(\sqrt{\log(n)}\right)$ in [AHK04] but the running time is still $\tilde{O}(n^2)$. In another direction, Spielman and Teng [ST04, ST06, ST08] introduced local graph partitioning which yields a cut near the specified seeds with running time only depending on the volume of the output. Their local partitioning algorithm has an approximation guarantee similar to the spectral method by using a mixing result on random walks [LS90]. Andersen, Chung and Lang [ACL06] used PageRank vectors to give a local partitioning algorithm with improved approximation guarantee and running time. Recently, Andersen and Peres use involving sets instead of PageRank to further improved the running time [AP09].

1.1.4 The Connection Graph and Its Consistency

In the past decade many non-linear dimensionality reduction methods have been well studied. These methods utilize the local affinities in the weighted graph to infer its global structure, including locally linear embedding (LLE) [HZX+09], Hessian LLE [DG03], Local Tangent Space Alignment (LTSA) [ZZ05], Laplacian eigenmaps [BN03] and diffusion maps [NLCK06]. Inspired by recent developments in the mathematical theory of cryo-electron microscopy [HS11, Sin11, SZSH11], Singer and Wu [StWss] introduced the connection graph as an invaluable tool for organizing complex networks and in particular for data and image processing involving high-dimensional data sets [StWss]. They demonstrated that in many applications, connection graphs significantly enhance the representation of data sets by attaching to every edge of the graph not only a scaler weight but also a linear orthogonal transformation, and thus can be used as an improved platform for performing objects partitioning and clustering in the related research fileds e.g. cryo-electron microscopy [HS11, Sin11, SZSH11], angular synchronization of eigenvectors [CLSss, Sin11, SZSH11] and vector diffusion maps [StWss].

However, the rotations of a connection graph are normally contaminated by “noise” which comes from the sampling procedure of high-dimensional data points.
and the way to determine the local structure of a data point via its neighbors [StWss]. The presence of noise would significantly flaw the quality of clustering base on the underlying connection graphs. Usually the assumption is that data points lie exactly on a manifold, without any noise contamination, i.e. the corresponding connection graph is consistent [StWss]. This assumption needs to be examined before the further use of corresponding connection graphs. Also, even though the underlying assumption doesn’t hold for the whole graph, we would still like to identify subgraphs which are relatively local consistent as many information networks arising from massive data sets exhibit the small world phenomenon [WS98]. Consequently exploring the local inconsistency or local consistency of small portions of a connection graph also attracts lots interests.

1.2 Summaries of Our Contributions

1.2.1 Edge Ranking, Graph Sparsification and Partitioning in the Usual Graphs

PageRank and Edge Ranking. In Chapter 2, we give an improved algorithm for computing (personalized) PageRank vectors with tight error bounds which can be as small as $O\left(\frac{1}{n^p}\right)$ for any fixed positive integer $p$. The improved PageRank algorithm is crucial for computing a quantitative ranking of edges in a given graph.

The PageRank is originally meant for determining the importance of vertices in the Web graph. It is also essential to identify the importance of edges in dealing with various problems. For a graph, we will use PageRank vectors to define a quantitative ranking that we call Green values for edges because of its connection with discrete Green functions. The Green values for edges can also be viewed as a generalized version of effective resistances in electrical network theory. The detailed definition of Green value for an edge will be given in Chapter 3 Section 3.1. We then use the sharp approximate PageRank algorithm to compute Green values within sharp error bounds in Chapter 3 Section 3.3.
Graph Sparsification and Partitioning. In Chapter 3, armed with the edge ranking defined by Green values, for a given graph, we use Green values to sample edges of $G$ in order to form a sparsifier $\tilde{G}$ with $O(n \log(n))$ edges. We analyze the sample algorithms for graph sparsification using the exact Green values in Section 3.2 and the sharply approximate Green values in Section 3.4. There are two advantages of sampling using PageRank and Green values. The running time of our sparsification algorithms is significantly simpler than those in [BSS09, ST04, ST06, SS08] since we avoid using Spielman-Teng solver [ST04, ST06] for solving linear system. In addition, the graph sparsification problem is closely related to graph partitioning algorithms.

Then we will combine the graph sparsification and the partitioning algorithms using PageRank vectors to derive an improved partitioning algorithm in Section 3.4. Our balanced-cut algorithm consists of two parts. First we use PageRank vectors to sparsify the graph. Then we use the known PageRank partitioning algorithm to find a balanced cut [ACL06] with target Cheeger ratio $\phi$. Consequently, the complexity for our PageRank balanced-cut algorithm is $O\left(\frac{m \log^2(n)}{\phi} + \frac{n \log^6(n)}{\phi^2}\right)$ for any input graph on $n$ vertices and $m$ edges.

1.2.2 Edge Ranking, Graph Sparsification and Consistency in Connection Graphs

Connection Laplacians and Connection Resistances. In Chapter 4, we give definitions for the connection graph and the connection Laplacian in Section 4.1. In particular, we discuss the notion of “consistency” in a connection graph (which is considered to be the ideal situation for various applications). We give a characterization for a consistent connection graph by using the eigenvalues of the connection Laplacian.

In Chapter 5, inspired by the effective resistance defined via the discrete Laplacian of usual graphs, we define the connection resistance via the connection Laplacian which can be viewed as a generalization version of effective resistance. We then examine various properties of the connection resistance, e.g. the connection resistances of pairs of vertices in a path or a tree.
Edge Ranking and Sparsification. In Chapter 6, we study the graph sparsification in connection graphs. The goal of the connection graph sparsification is to approximate a given connection graph $G$ with $m$ edges on $n$ vertices by a sparse connection graph $\tilde{G}$ with $O(n \log(n))$ edges (or fewer) on the same set of vertices such that the spectrums of the connection Laplacian are preserved. With the edge ranking defined by connection resistances, we are able to sample edges of $G$ in order to form a sparsifier $\tilde{G}$ with $O(n \log(n))$ edges. In the applications to cryo-electron microscopy, the edge ranking can also help eliminate the superfluous or erroneous edges that appear because of various “noises”.

PageRank and Consistency of a Connection Graph. In Chapter 7, we introduce the connection PageRank which is a generalization of PageRank in the usual graph. We give two efficient approximation algorithms for computing the connection PageRank vectors. The two algorithms provide somewhat different approximation guarantees. In some applications, we might require the approximation to be sharp with error bounds of order $\log \left( \frac{1}{\epsilon} \right)$ so that the bounds will still be effective when the $\epsilon$ is taken to be in the range of $O\left( \frac{1}{n^2} \right)$, for example.

Then, we give definitions for the inconsistency coefficient to measure the inconsistency for a connection graph or the local inconsistency of its subgraphs by using connection PageRank vectors in Section 7.2 and Section 7.3. We generalize the sweep cuts technique (which has been widely used in spectral partitioning and clustering) by using connection PageRank vectors in Section 7.4. We combine the inconsistency coefficient determined by the connection PageRank vectors and the sweep cuts produced by the connection PageRank vectors to derive an algorithm in Section 7.5 to find the local consistent area around a given vertex.
Chapter 2

Random Walks and PageRank
Vectors on a Graph

2.1 Random Walks on an Undirected Graph

Given a graph and a starting vertex, a random walker randomly selects one of its neighbors and move to this neighbor; then it select a neighbor of this vertex at random and move to it etc. The (random) sequence of vertices selected this way is a random walk on the graph.

A random walk is a finite Markov chain that is time reversible (see [Lov96]). However, there is little difference between the theory of random walks on graphs and the theory of finite Markov chains. Every Markov chain can be viewed as a random walk on a directed graph if we allow weighted edges and the probability that we randomly choose the neighbor is proportional to the edge weight linked to it. Similarly, time reversible Markov chains can be viewed as random walks on undirected graphs and symmetric Markov chains as random walks on regular symmetric graphs. In this paper we will formulate the results in terms of random walks and mostly restrict our attention to the undirected case.

We consider an undirected, weighted graph $G = (V, E, w)$ with $n$ vertices and $m$ edges where the edge weights satisfying $w(u, v) = w(v, u) \geq 0$ and the edge set $E$ consists of all pairs $(u, v)$ with $w(u, v) > 0$. The weighted degree $d(u)$ of
vertex $u$ is the sum of $w(u,v)$ over all $v$, i.e.,
\[ d(u) = \sum_v w(u,v). \]

A typical random walk is defined by its transition probability matrix $P$ satisfying
\[ P(u,v) = \frac{w(u,v)}{d(v)}. \]

We may also write
\[ P = AD^{-1}, \]
where $A$ is the weighted adjacency matrix satisfying $A(u,v) = w(u,v)$ for all pairs $(u,v) \in E$ and $D$ is the diagonal matrix of weighted degree, i.e.
\[ D(v,v) = d(v). \]

We here consider the lazy walk $Z$ on $G$, defined by
\[ Z = \frac{1}{2} (I + P). \]

In several recent applications of random walks, the most important parameter is the mixing rate. The mixing rate is a parameter to determine how fast the distribution defined by a random walker on an undirected graph $G$, starting from any specified vertex and walking a number of steps, converges to the stationary distribution $\pi$ defined by
\[ \pi(v) = \frac{d(v)}{\sum_{u \in V} d(u)}. \]

Using eigenvalues it is an easy task to determine the mixing rate in polynomial time. One of the well known mixing results for a random walk starting at vertex $u$ [Lov96] is that
\[ |p_t(v) - \pi(v)| \leq \sqrt{\frac{d(v)}{d(u)}} \chi^t \]
where $p_t(v)$ is the probability of staying at vertex $v$ after randomly walking $t$ steps starting from vertex $u$, i.e.
\[ p_t = P^t \chi_u. \]
\[ \lambda = \min\{\lambda_2, \lambda_n\}, \ \lambda_2 \text{ and } \lambda_n \text{ are the second largest and the smallest eigenvalues of } P \text{ respectively. More generally if for a subset of vertices } S \subseteq V \text{ we let } \pi(S) = \sum_{v \in S} \pi(v) \text{ and } p_t(S) = \sum_{v \in S} p_t(v), \text{ we will have} \]

\[ |p_t(S) - \pi(S)| \leq \sqrt{\frac{\pi(S)}{\pi(u)}} \lambda^t. \]

For more details about random walks on an undirected graph, please see the survey [Lov96].

### 2.2 PageRank Vectors on an Undirected Graph

PageRank was introduced by Brin and Page [BP98], is at the heart of Google’s web searching algorithms. Originally, PageRank was defined for the webgraph (which has all webpages as vertices and hyperlinks as edges). For any given graph, PageRank is well-defined and can be used for capturing quantitative correlations between pairs of vertices as well as pairs of subsets of vertices. For example, PageRank is used to quantitatively study the probability of a person staying at a website when he is browsing the internet. The behavior of an internet surfer is modeled as follows: he pick a random website as a starting point according to an initial probability distribution \( s \), and then at each step, with probability \( \alpha \) he either restarts to browse the internet from a new website as a starting point according to the initial probability distribution \( s \), or stays at the current website with probability \( 1 - \alpha \), or goes to one of its links with uniform possibilities.

Formally, a PageRank vector \( \text{pr}_{\alpha,s} \) [ACL06], is defined by a recurrence relation involving a seed vector \( s \) (as a probability distribution or preference distribution) and a positive jumping constant \( \alpha \in (0, 1] \) (or transportation constant), i.e.

\[ \text{pr}_{\alpha,s} = \alpha s + (1 - \alpha) Z \text{pr}_{\alpha,s}. \tag{2.1} \]

Another (equivalent) way to express the recurrence of PageRank in terms of a parameter \( \beta \) and a seed vector \( s \) is that: for a positive value \( \beta > 0 \), the (personalized)
PageRank vector $p_{r,\beta,s}$ with a seed vector $s$ is the unique solution of equation

$$p_{r,\beta,s} = \frac{\beta}{2+\beta} s + \frac{2}{2+\beta} Z p_{r,\beta,s}. \quad (2.2)$$

That is, we replace $\alpha$ with $\alpha = \frac{\beta}{2+\beta}$.

PageRank whose preference distribution equals to the uniform distribution is usually used to provided searching ranking [Hav03]. In order to provide more specified search service for different people with different favorites, the preference distribution may have distribution concentrated on some specified vertices. PageRank vectors with such kinds of preference vector are often call personalized PageRank vectors and have been extensively studied in [Ber06, FRCS05, JW03]. If the seed vector $s$ is a characteristic function $\chi_u$ of a vertex $u$, we may write $p_{r,\beta,\chi_u} = p_{r,\beta,u}$. It is easy to check that $\sum_{v \in V} p_{r,\beta,s}(v) = 1$, i.e. $\|p_{r,\beta,s}\|_1 = 1$ since $s$ is a distribution over vertices set $V$, i.e. $\|s\|_1 = 1$.

Here are some useful properties of PageRank vectors (also see [ACL06, AC07])

**Proposition 2.2.1 ([ACL06])** Let $r_{p,\alpha,s}$ be the standard PageRank vector which is the unique solution to the linear system

$$r_{p,\alpha,s} = \alpha s + (1-\alpha) P r_{p,\alpha,s}. $$

Then $p_{\alpha,s} = r_{p,\frac{\alpha}{1+\alpha},s}$.

The standard PageRank vector are more widely used than the one defined here, but the above proposition implies that if we can compute the approximate PageRank vector efficiently here, then we can also compute approximate standard PageRank vector efficiently too for other applications. They are actually equivalent to each other.

**Proposition 2.2.2 ([ACL06])** For any preference distribution $s$, any constant $\alpha$, there is only a unique solution to the linear system

$$p_{\alpha,s} = \alpha s + (1-\alpha) Z p_{r,\alpha,s}. $$
Proposition 2.2.3 ([ACL06]) For any fixed $\alpha \in (0, 1]$, there is a linear transformation $T_\alpha$ such that $pr_{\alpha,s} = T_\alpha s$ where
\[
T_\alpha = \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t Z^t.
\]

2.3 PageRank Approximate Algorithms

One of the fundamental problems for PageRank is how to compute a PageRank vector efficiently. Instead of computing the PageRank vector $pr_{\beta,s}$ exactly, we will approximate it by another PageRank vector with a slightly different starting vector, $pr_{\beta,s-r}$, where $r$ is a vector with nonnegative entries. If
\[
 r(v) \leq \epsilon d(v)
\]
for every vertex $v \in V$ in the graph, then we say $pr_{\beta,s-r}$ is an $\epsilon$-approximate PageRank vector for $pr_{\beta,s}$.

For any $\epsilon > 0$, such an $\epsilon$-approximate PageRank vector can be efficiently computed (see [ACL06, AP09, Ber06, Hav03, JW03]). The running time of the approximation algorithm in [ACL06] for computing a PageRank vector within an error bound of $\epsilon$ is basically $O(\frac{1}{\epsilon})$. For the problems that we will examine in the chapter 3, it is quite crucial to have a sharper error bound. In the next section, we will give an improved approximation algorithm with running time $O(m \log (\frac{1}{\epsilon}))$ to compute a PageRank vector within an error bound of $\epsilon$ for a graph of $m$ edges.

The PageRank approximation algorithm in [ACL06] contains two routines, called Push and ApproximatePR for approximating PageRank vectors defined by Equation (2.1). For the purpose of manipulating the PageRank vectors defined in Equation (2.2) and the completeness of our thesis, we will give routines generalized from those in [ACL06], also called Push and ApproximatePR, for approximating the PageRank vectors defined in Equation (2.2). They will serve as subroutines later in our sharp approximate PageRank algorithm SharpApproximatePR in the next section.

Given a vertex $u$, an approximate PageRank vector $p$ and a residual vector $r$, the Push operation maintains a pair of vectors $p$ and $r$, starting with the some
given approximation vectors, and applies a series of \textbf{Push} operations which move probability from $r$ to $p$ while maintaining the invariant $p = pr_{\beta,s-r}$. Formally, the \textbf{Push} operation is as follows.

**Push**($u$)

Let $p' = p$ and $r' = r$, except for these changes:

1. Let $p'(u) = p(u) + \frac{\beta}{2+\beta} r(u)$ and $r'(u) = \frac{1}{2+\beta} r(u)$.
2. For each vertex $v$ such that $(u,v) \in E$: $r'(v) = r(v) + \frac{r(u)}{(2+\beta)d(u)}$.

Although our definition of PageRank vectors is slightly different from that of [ACL06], one may still have the following Lemma 2.3.1 and Theorem 2.3.2 derived from [ACL06] which will be useful in the proof of Theorem 2.4.1. For the completeness of this thesis, we state them as follows.

**Lemma 2.3.1 ([ACL06])** Let $p'$ and $r'$ denote the resulting vectors after performing operation \textbf{Push} with vectors $p$ and $r$ on some vertex $u$. Then $p = pr_{\beta,s-r}$ implies $p' = pr_{\beta,s-r'}$.

\[
(p, r) = \text{ApproximatePR}(s, \beta, \epsilon) \\
1. \text{Let } p = \vec{0} \text{ and } r = s. \\
2. \text{While } r(u) \geq \epsilon d(u) \text{ for some vertex } u: \\
   \quad \text{Pick any vertex } u \text{ where } r(u) \geq \epsilon d(u) \text{ and apply operation } \textbf{Push}(u). \\
3. \text{Return } p \text{ and } r.
\]

In the algorithm \text{ApproximatePR}, during each \textbf{Push} operation, some probability is moved from $r$ to $p$. It performs the \textbf{Push} operations only on vertices where $r(u) \geq \epsilon d(u)$, which ensures that a significant amount of probability is moved at each step, and allows us to bound the number of \textbf{Push} operations required to compute an $\epsilon$-approximate PageRank vector.
Theorem 2.3.2 ([ACL06]) For any vector $s$ with $\|s\|_1 \leq 1$, and any constant $\epsilon \in (0, 1]$, the algorithm $\text{ApproximatePR}(s, \beta, \epsilon)$ computes an approximate PageRank vector $p = pr_{\beta, s} - r$ such that the residual vector $r$ satisfies

$$\forall v \in V, r(v) \leq \epsilon d(v).$$

Furthermore, the running time of algorithm is $O\left(\frac{2+\beta}{\beta \epsilon}\right)$.

2.4 A Sharp PageRank Approximation Algorithm

Now, we are ready to show how to improve the estimate error bound for the algorithm $\text{ApproximatePR}$ by the following iterative process $\text{SharpApproximatePR}$.

\[
(p, r) = \text{SharpApproximatePR}(s, \beta, \epsilon)
\]

1. Let $\epsilon' = 1$, $r = s$ and $p = \vec{0}$.

2. While $\epsilon' > \epsilon$:

2.a Set $\epsilon' = \frac{\epsilon'}{2}$.

2.b Let $p'$ and $r'$ be the output of $\text{ApproximatePR}(r, \beta, \epsilon')$.

2.c Let $p = p + p'$ and $r = r'$.

3. Return $p$ and $r$.

Theorem 2.4.1 Given constants $\epsilon, \beta \in (0, 1]$, and any seed vector $s$ with $\|s\|_1 \leq 1$, to approximate a PageRank vector $pr_{\beta, s}$, the algorithm $\text{SharpApproximatePR}(s, \beta, \epsilon)$ computes an approximate PageRank vector $p = pr_{\beta, s} - r$ such that the residual vector $r$ satisfies

$$\forall v \in V, r(v) \leq \epsilon d(v).$$
and the running time is $O\left(\frac{m\log(1/\epsilon)}{\beta}\right)$. In particular, if $\epsilon$ is an inverse of a polynomial on $n$ with degree $p$, i.e. $\Omega\left(\frac{1}{n^p}\right)$, the running time can be bounded by $O\left(\frac{m\log(n)}{\beta}\right)$.

**Proof.** Clearly, by Lemma 2.3.1 and Theorem 2.3.2 the algorithm returns an approximate PageRank vector $p = pr_{\beta,s-r}$ where the residual vector satisfies that $r(u) \leq \epsilon d(u)$ for all $u \in V$.

To bound the running time, we consider one fixed round of while-loop in the second step. Let $T$ denote the total number of Push operations performed by ApproximatePR and let $d_i$ be the degree of vertex involved in the $i$th Push operation. When the $i$th Push($u$) operation was performed, the PageRank vector at this vertex was at least $\epsilon'd_i$, so $\|r\|_1$ decreased by at least $\epsilon'\beta d_i$. Since $\|r\|_1$ is at most $2\epsilon' \sum_{v \in V} d(v) = 2m\epsilon'$, we have

$$\frac{\epsilon'\beta}{2 + \beta} \sum_{i=1}^{T} d_i \leq 2m\epsilon'$$

which implies that

$$\sum_{i=1}^{T} d_i \leq \frac{2(2 + \beta)m}{\beta}.$$

Note that there are at most $\log\left(\frac{1}{\epsilon}\right)$ rounds in the second step. Thus, the total running time is bounded by $O\left(\frac{m\log(1/\epsilon)}{\beta}\right)$.

### 2.5 Acknowledgement

Material in this chapter has appeared in the following article:

Chapter 3

Ranking and Sparsifying Edges of a Graph by PageRank

PageRank, which was first introduced by Brin and Page [BP98], is at the heart of Google’s web searching algorithms. PageRank was originally defined for the web graph (which has all webpages as vertices and hyperlinks as edges) and was meant for determining the “importance” of vertices in the Web graph. It is also essential to identify the “importance” of edges in dealing with various problems. We will use PageRank vectors to define a quantitative ranking that we call Green values for edges because of its connection with discrete Green functions. In addition, PageRank vectors can be efficiently computed and approximated (see [ACL06, AP09, Ber06, Hav03, JW03]). The running time of the approximation algorithm in [ACL06] for computing a PageRank vector within an error bound of $\epsilon$ is basically $O(\frac{1}{\epsilon})$. For the problems that we will examine in this chapter, it is quite crucial to have a sharper error bound. In the previous section, we presented an improved approximation algorithm with running time $O(m \log \left( \frac{1}{\epsilon} \right))$ to compute a PageRank vector within an error bound of $\epsilon$ for a graph of $m$ edges. We then use this sharp approximate PageRank algorithm to compute approximate Green values within sharp error bounds.

The Green values for edges can also be viewed as a generalized version of effective resistances in electrical network theory. The detailed definition of Green value for an edge will be given in Section 3.1. To illustrate the usage of Green
values, we examine a basic problem on graph sparsification. Graph sparsification was first introduced by Benczúr and Karger [BK96, Kar94a, Kar94b, Kar00] for approximately solving various network design problems. The heart of the graph sparsification algorithms is the sampling techniques for randomly selecting edges. The goal is to approximate a given graph $G$ with $m$ edges on $n$ vertices by a sparse graph $\tilde{G}$, called sparsifier, with $O(n \log(n))$ edges (or fewer) on the same set of vertices in such a way that every cut in sparsifier $\tilde{G}$ is within a factor of $1 \pm \epsilon$ of the corresponding cut in $G$ for some constant $\epsilon \in (0, 1]$. It was shown that

$$\forall x \in \{0, 1\}^n, (1 - \epsilon)x^T Lx \leq x^T \tilde{L}x \leq (1 + \epsilon)x^T Lx$$

where $L$ and $\tilde{L}$ are the Laplacians of graph $G$ and its sparsifier $\tilde{G}$, respectively.

Spielman and Teng [ST04] devised a sampling scheme to construct a spectral sparsifier with $O(n \log^c(n))$ edges for some large constant $c > 0$ in $\tilde{O}(m)$ time. A spectral sparsifier $\tilde{G}$ for graph $G$ is a sparsifier satisfying

$$\forall x \in \mathbb{R}^n, (1 - \epsilon)x^T Lx \leq x^T \tilde{L}x \leq (1 + \epsilon)x^T Lx.$$ 

In [SS08], Spielman and Srivastava gave a different sampling scheme using the effective resistances of electrical networks to construct an improved spectral sparsifier with only $O(n \log(n))$ edges. In the process of constructing such a spectral sparsifier, they need to use the Spielman-Teng solver [ST04] as subroutines for solving $O(\log(n))$ linear systems. The running time of their sparsification algorithm is mainly dominated by the running time of Spielman-Teng solver which is $O(m \log^c(n))$ for a very large constant $c$ [ST04]. It has been improved to $O(m \log^2(n) \log^2 \log(n))$ by Koutis, Miller, and Peng [KMP10] very recently. Later on, Batson, Spielman and Srivastava [BSS09] gave an elegant construction for a spectral sparsifier with a linear number of edges although the running time is $O(n^3m)$.

For graph partitioning algorithms, previously widely used approach is the recursive spectral method which finds a balanced cut in a graph on $n$ vertices with running time $\tilde{O}\left(\frac{n^2}{\lambda}\right)$ (see [Spi96]), together with an approximation guarantee within a quadratic root of the optimal conductance where $\lambda$ denotes the spectral gap of the normalized Laplacian. The running time can be further improved to
by using Spielman-Teng solver for linear systems [ST04, ST06, KMP10]. Another approach for the balanced cut problem is by using commodity flows [AK07, OSVV08]. In [AK07], the approximation guarantee is within a factor of \( O \left( \log(n) \right) \) of the optimal, which was further reduced to \( O \left( \sqrt{\log(n)} \right) \) in [AHK04] but the running time is still \( \tilde{O}(n^2) \). In another direction, Spielman and Teng [ST04, ST06, ST08] introduced local graph partitioning which yields a cut near the specified seeds with running time only depending on the volume of the output. Their local partitioning algorithm has an approximation guarantee similar to the spectral method by using a mixing result on random walks [LS90]. Andersen, Chung and Lang [ACL06] used PageRank vectors to give a local partitioning algorithm with improved approximation guarantee and running time. Recently, Andersen and Peres use involving sets instead of PageRank to further improved the running time [AP09].

Our balanced-cut algorithm in Section 3.4 consists of two parts. First we use PageRank vectors to sparsify the graph. Then we use the known PageRank partitioning algorithm to find a balanced cut [ACL06] with target Cheeger ratio \( \phi \). Both parts have the same complexity as approximately computing the PageRank vectors. Consequently, the complexity for our PageRank balanced-cut algorithm is \( O \left( \frac{m \log^2(n)}{\phi} + \frac{n \log^6(n)}{\phi^2} \right) \) for any input graph on \( n \) vertices and \( m \) edges. The balanced-cut algorithm here can be viewed as an application of graph sparsification.

### 3.1 The PageRank and Green Values for Edges

Recall that the discrete Laplacian of \( G \) is defined by

\[
L = D - A.
\]

If we orient the edges of \( G \) in an arbitrary but fixed way, we can write its discrete Laplacian as

\[
L = B^TWB,
\]
where $B_{m\times n}$ is the signed edge-vertex incidence matrix, given by

$$B(e, v) = \begin{cases} 
1 & \text{if } v \text{ is } e\text{'s head}, \\
-1 & \text{if } v \text{ is } e\text{'s tail}, \\
0 & \text{otherwise,}
\end{cases}$$

and $W_{m\times m}$ is the diagonal matrix with $W(e, e) = w(e)$. The normalized Laplacian of $G$ is defined to be

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

and we can write

$$\mathcal{L} = S^TWS$$

where $S_{m\times n} = BD^{-1/2}$. Since $\mathcal{L}$ is symmetric and we have

$$\mathcal{L} = \sum_{i=0}^{n-1} \lambda_i \phi_i \phi_i^T,$$

where $\lambda_0 = 0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{n-1} \leq 2$ are eigenvalues of $\mathcal{L}$ and $\phi_0, \ldots, \phi_{n-1}$ are a corresponding orthonormal basis of eigenvectors. Various properties concerning eigenvalues of the normalized Laplacian can be found in [Chu97]. Denote the $\beta$-normalized Laplacian $\mathcal{L}_\beta$ by

$$\mathcal{L}_\beta = \beta I + \mathcal{L},$$

and we may write

$$\mathcal{L}_\beta = S'^T W_\beta S'$$

where we define $S'$ and $W_\beta$ as follows:

$$S' = \begin{bmatrix} I \\ S \end{bmatrix}_{(n+m)\times n} \quad \text{and} \quad W_\beta = \begin{bmatrix} \beta I & 0 \\ 0 & W \end{bmatrix}_{(n+m)\times(n+m)}.$$

Here the index sets for the columns of $S'$ and columns (rows) of $W_\beta$ are $V \cup E$ where the first $n$ columns (rows) are indexed by $V$ and the last $m$ columns (rows) are indexed by $E$.

Green’s functions were first introduced in a celebrated essay by George Green [Gre28] in 1828. Since then, the concept of Green’s functions has been
used in numerous areas, especially in the study of partial differential equations and quantum field theory. The discrete analog of Green’s function which are associated with the normalized Laplacian of graphs were considered in [CY00] in connection with the study of Dirichlet eigenvalues with boundary conditions. In particular, the following modified Green’s function $G_\beta$ was used in [CY00]. For $\beta \in \mathbb{R}^+$, let Green’s function $G_\beta$ denote the symmetric matrix satisfying

$$\mathcal{L}_\beta G_\beta = I.$$  

Clearly, we have

$$G_\beta = \sum_{i=0}^{n-1} \frac{1}{\lambda_i + \beta} \phi_i \phi_i^T. \quad (3.1)$$

We remark that the discrete Green’s function is basically a symmetric form of PageRank. Namely, it is straightforward to check that

$$\frac{pr_{\beta,s}}{\beta} = sD^{-1/2}G_\beta D^{1/2}. \quad (3.2)$$

For every edge $e = (u, v) \in E$, we define the Green value $g_\beta(u, v)$ of $e = (u, v)$ to be a combination of four terms in PageRank vectors as follows:

$$g_\beta(u, v) = \frac{pr_{\beta,u}(u)}{d(u)} - \frac{pr_{\beta,u}(v)}{d(v)} + \frac{pr_{\beta,v}(v)}{d(v)} - \frac{pr_{\beta,v}(u)}{d(u)}. \quad (3.3)$$

Note that it is easy to verify that

$$g_\beta(u, v) = \beta(\chi_u - \chi_v)^T D^{-1/2}G_\beta D^{-1/2}(\chi_u - \chi_v) \quad (3.4)$$

by Equation (3.2). By using Equation (3.1) and Equation (3.3), we can show the following facts which will be useful later.

**Lemma 3.1.1** For any constant $\beta \in (0, 1]$ and every edge $e = (u, v) \in E$, Green value $g_\beta(u, v)$ can be expressed as the following form

$$\sum_{i=0}^{n-1} \frac{\beta}{\lambda_i + \beta} \left( \frac{\phi_i(u)}{\sqrt{d(u)}} - \frac{\phi_i(v)}{\sqrt{d(v)}} \right)^2.$$  

In particular, Green value $g_\beta(u, v)$ satisfies the following inequalities

$$\frac{\beta}{2 + \beta \left( \frac{1}{d(u)} + \frac{1}{d(v)} \right)} \leq g_\beta(u, v) \leq \frac{1}{d(u)} + \frac{1}{d(v)}.$$
Proof. By plugging Equation (3.1) into Equation (3.4), we have
\[ g_\beta(u,v) = \sum_{i=0}^{n-1} \frac{\beta}{\lambda_i + \beta} \left( \frac{\phi_i(u)}{\sqrt{d(u)}} - \frac{\phi_i(v)}{\sqrt{d(v)}} \right)^2 \]
where \( \lambda_0 = 0 \) and \( 0 < \lambda_1 \leq \lambda_2 \ldots \leq \lambda_{n-1} \leq 2 \) are the nonzero eigenvalues of \( \mathcal{L} \) and \( \phi_0, \phi_1, \ldots, \phi_{n-1} \) are the corresponding orthonormal eigenvectors. Since \( \beta \in (0, 1] \), we have \( \frac{\beta}{2+\beta} \leq \frac{\beta}{\lambda_i + \beta} \leq 1 \), and thus
\[ \frac{\beta}{2+\beta} \left( \frac{\phi_i(u)}{\sqrt{d(u)}} - \frac{\phi_i(v)}{\sqrt{d(v)}} \right)^2 \leq g_\beta(u,v) \leq \sum_{i=0}^{n-1} \left( \frac{\phi_i(u)}{\sqrt{d(u)}} - \frac{\phi_i(v)}{\sqrt{d(v)}} \right)^2. \]
Note that for two fixed vertices \( u \) and \( v \), the vector \( f_u \), defined by \( f_u(i) = \phi_i(u) \), for \( i = 0, 1, \ldots, n-1 \), is orthogonal to \( f_v \). This implies
\[ \sum_{i=0}^{n-1} \frac{\phi_i(u)\phi_i(v)}{\sqrt{d(u)d(v)}} = 0, \]
and
\[ \sum_{i=0}^{n-1} \left( \frac{\phi_i(u)}{\sqrt{d(u)}} - \frac{\phi_i(v)}{\sqrt{d(v)}} \right)^2 = \sum_{i=0}^{n-1} \left( \frac{\phi_i^2(u)}{d(u)} + \frac{\phi_i^2(v)}{d(v)} \right) = \frac{1}{d(u)} + \frac{1}{d(v)}. \]
Thus, the lemma follows. \( \blacksquare \)

Since the Green values can be relatively small (e.g., of order \( \Omega \left( \frac{1}{n^p} \right) \) for some large but fixed positive integer \( p \)), we may need a very sharply approximate PageRank to be within a factor of \( 1 \pm o \left( \frac{1}{n^p} \right) \) of the exact values in the analysis of performance bound for our graph sparsification algorithms that we will examine later. For every edge \((u,v) \in E\), we define the approximate Green value \( \tilde{g}_\beta(u,v) \) by
\[ \tilde{g}_\beta(u,v) = \frac{\text{pr}_{\beta,Xu - Y_u}(u)}{d(u)} - \frac{\text{pr}_{\beta,Xu - Y_u}(v)}{d(v)} + \frac{\text{pr}_{\beta,Yv - Y_v}(v)}{d(v)} - \frac{\text{pr}_{\beta,Xv - Y_v}(u)}{d(u)}. \]
Here, \( pr_{\beta,u} - r_{\chi u} \) and \( pr_{\beta,v} - r_{\chi v} \) are the approximate PageRank vectors as outputs of \texttt{ApproximatePR} for exact PageRank vectors \( pr_{\beta,u} \) and \( pr_{\beta,v} \) respectively; \( r_{\chi u} \) and \( r_{\chi v} \) are the corresponding residual vectors satisfying
\[
\| r_{\chi u}^T D^{-1} \|_1 \leq \frac{\epsilon}{4} \quad \text{and} \quad \| r_{\chi v}^T D^{-1} \|_1 \leq \frac{\epsilon}{4}.
\]

With this definition, we can prove the following lemma.

**Lemma 3.1.2** For every edge \((u, v) \in E\), we have
\[
|g_{\beta}(u, v) - \tilde{g}_{\beta}(u, v)| \leq \epsilon.
\]

**Proof.** Let \( T_\beta = \beta D^{-1/2} G_{\beta} D^{1/2} \), and we may express \( \tilde{g}_{\beta}(u, v) \) as the following form
\[
\tilde{g}_{\beta}(u, v) = (\chi_u - r_{\chi u})^T T_{\beta} D^{-1} \chi_u - (\chi_u - r_{\chi u})^T T_{\beta} D^{-1} \chi_v + (\chi_v - r_{\chi v})^T T_{\beta} D^{-1} \chi_v - (\chi_v - r_{\chi v})^T T_{\beta} D^{-1} \chi_u,
\]
which implies that
\[
g_{\beta}(u, v) - \tilde{g}_{\beta}(u, v)
= r_{\chi u}^T T_{\beta} D^{-1} \chi_u + r_{\chi u}^T T_{\beta} D^{-1} \chi_u + r_{\chi v}^T T_{\beta} D^{-1} \chi_v + r_{\chi v}^T T_{\beta} D^{-1} \chi_v.
\]
Now, the key observation is that \( \mathbf{1}^T D T_{\beta} D^{-1} = \mathbf{1}^T \) and thus \( \| D T_{\beta} D^{-1} \|_1 = 1 \).
Therefore
\[
\| r_{\chi u}^T T_{\beta} D^{-1} \|_1 = \| r_{\chi u}^T D^{-1} DT_{\beta} D^{-1} \|_1 \\
\leq \| r_{\chi u}^T D^{-1} \|_1 \| DT_{\beta} D^{-1} \|_1 \\
\leq \| r_{\chi u}^T D^{-1} \|_1 \\
\leq \frac{\epsilon}{4}.
\]
Thus, we have
\[
|g_{\beta}(u, v) - \tilde{g}_{\beta}(u, v)|
\leq |r_{\chi u}^T T_{\beta} D^{-1} \chi_u| + |r_{\chi u}^T T_{\beta} D^{-1} \chi_u| + |r_{\chi v}^T T_{\beta} D^{-1} \chi_v| + |r_{\chi v}^T T_{\beta} D^{-1} \chi_v|
\leq \epsilon
\]
and thus the Lemma is proved. \( \blacksquare \)
Here we will give a rough estimation for computing Green’s values by directly using Lemma 3.1.2 and Theorem 2.3.2. Note that by invoking Theorem 2.3.2 without using further techniques the running time does not seem to be improved.

**Theorem 3.1.3** Given any constant \( \epsilon > 0 \) and every edge \((u, v) \in E, \) the approximate Green value \( \tilde{g}_\beta(u, v) \) can be computed in \( O \left( \frac{(2+\beta)n}{\beta \epsilon} \right) \) time such that

\[
|g_\beta(u, v) - \tilde{g}_\beta(u, v)| \leq \epsilon.
\]

In particular, after \( O \left( \frac{(2+\beta)n}{\beta \epsilon} \right) \) preprocessing time, for every edge \((u, v) \in E, \) we can compute such \( \tilde{g}_\beta(u, v) \) by using 4 times of queries.

**Proof.** By Theorem 2.3.2, we can compute approximate PageRank \( pr_{\beta,\chi u - \chi v} \) for the exact PageRank vector \( pr_{\beta,v} \) such that the residual vector \( r_{\chi v} \) satisfies

\[
\left| \frac{r_{\chi v}(v)}{d(v)} \right| \leq \frac{\epsilon}{4}, \quad \forall v \in V
\]

in \( O \left( \frac{2+\beta}{\beta \epsilon} \right) \) time, respectively. By the definition of Green (approximate) value, we need to compute \( n \) such approximate PageRank vectors for preprocessing, thus the theorem follows. \( \blacksquare \)

Recall that in Lemma 3.1.1, we established a lower bound for \( g_\beta(u, v) \). We denote as \( \Delta \) the maximum degree of graph \( G. \) Then, a direct consequence of the above theorem is the following.

**Corollary 3.1.4** Given any constant \( \epsilon > 0 \) and every edge \((u, v) \in E, \) the approximate Green value \( \tilde{g}_\beta(u, v) \) can be computed in \( O(\frac{\Delta}{\beta \epsilon}) \) time such that

\[
|g_\beta(u, v) - \tilde{g}_\beta(u, v)| \leq \epsilon g_\beta(u, v).
\]

In particular, after \( O(\frac{\Delta n}{\beta \epsilon}) \) preprocessing time, for every edge \((u, v) \in E, \) we can compute such \( \tilde{g}_\beta(u, v) \) by using 4 times of queries.

**Proof.** In Lemma 3.1.1, we establish a lower bound for \( g_\beta(u, v) \) as

\[
g_\beta(u, v) \geq \frac{\beta}{(2+\beta)\Delta}.
\]
If we set $\epsilon' = \frac{\beta \epsilon}{(2 + \beta) \Delta}$ for a target error bound $\epsilon$ and apply Theorem 3.1.3 with parameter $\epsilon'$ instead of $\epsilon$, we have

$$|g_\beta(u, v) - \tilde{g}_\beta(u, v)| \leq \epsilon' = \frac{\beta}{(2 + \beta) \Delta} \leq \epsilon g_\beta(u, v).$$

Thus, the corollary follows.

We will improve both Theorem 3.1.3 and Corollary 3.1.4 in the Section 3.3 by using sharp approximate PageRank algorithm $\text{SharpApproximatePR}$ and the dimension reduction techniques used in [SS08].

### 3.2 Graph Sparsification Using Green Values

To construct our sparsifier, we use a method quite similar to the scheme used by Spielman and Srivastava [SS08] except that PageRank vector and Green value are used here instead of effective resistance. We will give several graph sparsification algorithms, some of which involve approximate Green values (which will be examined in section 3.3). In this section, we only use exact Green values for edges.

Without loss of generality, we consider a graph $G = (V, E, w)$ which is an undirected unweighted graph. For a subset $S$ of vertices in $G$, the edge boundary $\partial(S)$ of $S$ consists of all edges with exactly one endpoint in $S$. The weight of $\partial(S)$, denoted by $w(S, \bar{S})$, is the sum of all edge weights of edges in $\partial(S)$ (i.e., the number of edges as $G$ is unweighted graph). The volume of $S$, denoted by $\text{Vol}(S)$, is defined to be the sum of degrees $d(v)$ over all $v$ in $S$. When $S = V$, we write $\text{Vol}(S) = \text{Vol}(G)$. The Cheeger ratio (or conductance) $h_G(S)$ of $S$ in $G$ is defined by

$$h_G(S) = \frac{w(S, \bar{S})}{\min\{\text{Vol}(S), \text{Vol}(\bar{S})\}}.$$  

The conductance $h_G$ of $G$ is defined to be the minimum Cheeger ratio among all subsets $S$ with $\text{Vol}(S) \leq \frac{\text{Vol}(G)}{2}$.

The goal of sparsification is to approximate a given graph $G$ by a sparse graph $\tilde{G}$ on the same set of vertices while the sparse graph $\tilde{G}$ preserves the Cheeger
ratios of every subset of vertices to be within a factor of $1 \pm \epsilon$. The main step in any sparsification algorithm [BK96, Kar94a, Kar94b, ST04, ST08] is to choose an appropriate probability distribution for random sampling the edges in a way that Cheeger ratios of subsets change little. Our sparsification algorithm is a sampling process using probabilities proportional to the Green values as follows.

$$(\tilde{G}) = \text{SparsifyExactGreen}(G, q, \beta)$$

1. For each $e = (u, v) \in E$, set probability $p_e \propto w(e)g_{\beta}(e)$ and repeat the following steps for $q$ times:
   2.a Choose an edge $e \in G$ randomly with probability $p_e$.
   2.b Add $e$ to $\tilde{G}$ with weight $\frac{w(e)}{qp_e}$.
   2.c Sum the weights if an edge is chosen more than once.
2. Return $\tilde{G}$.

The analysis of the above algorithm will be examined later. Our main theorem is the following

**Theorem 3.2.1** Given an unweighted graph $G$ on $n$ vertices with $m$ edges, for any constant $\epsilon \in (0, 1]$, let $\tilde{G}$ be the output of the algorithm $\text{SparsifyExactGreen}(G, q, \beta)$, where $q = \frac{256C^2n\log(n)}{\epsilon^2}$, $\beta = \frac{1}{2}$, $C \geq 1$ is a absolute constant. Then with probability at least $\frac{1}{2}$, we have

$$|h_{\tilde{G}}(S) - h_G(S)| \leq \epsilon, \forall S \subset V.$$

Our analysis follows the general scheme as that of [SS08]. In our analysis of sparsifier, we consider the matrix

$$\Lambda_{\beta} = W_{\beta}^{1/2}S'G_{\beta}S^{\beta T}W_{\beta}^{1/2}.$$  

Note that $\Lambda_{\beta}$ is a $(n + m) \times (n + m)$ matrix and we index its the first $n$ columns (rows) by $V$ and its last $m$ columns (rows) by $E$. From the definition and properties
of Green values in Section 3.1, one may verify that

\[ \Lambda_\beta(e,e) = \frac{1}{\beta} \sqrt{W_\beta(e,e) g_\beta(e)} \sqrt{W_\beta(e,e)} = \frac{1}{\beta} w(e) g_\beta(e). \]

Here are several useful properties for matrix \( \Lambda_\beta \).

**Lemma 3.2.2** Matrix \( \Lambda_\beta \) has the following properties.

(i) \( \Lambda_\beta^2 = \Lambda_\beta \).

(ii) The dimension (or rank) of \( \Lambda_\beta \), denoted by \( \text{dim}(\Lambda_\beta) \) is \( n \).

(iii) The eigenvalues of \( \Lambda_\beta \) are 1 with multiplicity \( n \) and 0 with multiplicity \( m \).

(iv) \( \Lambda_\beta(e,e) = \| \Lambda_\beta(\cdot,e) \|_2^2. \)

**Proof.** For (i), we note that

\[
\Lambda_\beta \Lambda_\beta = (W_{\beta}^{1/2} S' \mathcal{G}_\beta S'^T W_{\beta}^{1/2}) (W_{\beta}^{1/2} S' \mathcal{G}_\beta S'^T W_{\beta}^{1/2}) = W_{\beta}^{1/2} S' \mathcal{G}_\beta (S'^T W_{\beta} S') \mathcal{G}_\beta S'^T W_{\beta}^{1/2} = W_{\beta}^{1/2} S' \mathcal{G}_\beta \mathcal{L}_\beta \mathcal{G}_\beta S'^T W_{\beta}^{1/2} = W_{\beta}^{1/2} S' \mathcal{G}_\beta S'^T W_{\beta}^{1/2} = \Lambda_\beta.
\]

For (ii), it is easy to verify that \( \dim(W_{\beta}^{1/2} S' \mathcal{G}_\beta S'^T W_{\beta}^{1/2}) = \dim(S' \mathcal{G}_\beta S'^T) = \dim(\mathcal{G}_\beta) = n. \)

For (iii), Since \( \Lambda_\beta^2 = \Lambda_\beta \), the eigenvalue of \( \Lambda_\beta \) are all 0 or 1. Since \( \dim(\Lambda_\beta) = n \), there must be \( n \) nonzero eigenvalues.

(iv) follows from the fact that \( \Lambda_\beta \) is symmetric. \( \blacksquare \)

Next, we introduce some notations and several lemmas that the theorems later rely on. Let \( \bar{w}(e) \) be the edge weight of edge \( e \) in the sparsifier \( \bar{G} \). Recall that \( q \) is the number of times of sampling and \( p_e \) is the sampling probability for \( e \in E \). Denote \( I_\beta \) as a nonnegative diagonal matrix

\[
I_\beta = \begin{bmatrix}
I_{n \times n} & 0 \\
0 & R
\end{bmatrix}_{(n+m) \times (n+m)}
\]
where
\[ R(e, e) = \frac{\tilde{w}(e)}{w(e)} = \frac{\# \text{ of times } e \text{ is sampled}}{qp_e}. \]

**Lemma 3.2.3** Suppose \( I_\beta \) is a nonnegative diagonal matrix such that
\[ \| \Lambda_\beta I_\beta \Lambda_\beta - \Lambda_\beta \Lambda_\beta \|_2 \leq \epsilon. \]
Then we have
\[ \forall x \in \mathbb{R}^n, \quad \left| x^T \tilde{L}_\beta x - x^T L_\beta x \right| \leq \epsilon x^T L_\beta x \]
where \( L_\beta = S'^T W_\beta S' \) and \( \tilde{L}_\beta = S'^T W_\beta^{1/2} I_\beta W_\beta^{1/2} S' \).

**Proof.** The assumption of this lemma is equivalent to
\[ \sup_{y^T y \neq 0} \frac{|y^T \Lambda_\beta (I_\beta - I) \Lambda_\beta y|}{y^T y} \leq \epsilon, \]
which implies
\[ \sup_{y \in \text{im}(W_\beta^{1/2} S'), y \neq 0} \frac{|y^T \Lambda_\beta (I_\beta - I) \Lambda_\beta y|}{y^T y} \leq \epsilon. \]
Here \( y \in \text{im}(M) \) means \( y = xM \) for some \( x \). We have
\[
\sup_{y \in \text{im}(W_\beta^{1/2} S'), y \neq 0} \frac{|y^T \Lambda_\beta (I_\beta - I) \Lambda_\beta y|}{y^T y} = \sup_{x \in \mathbb{R}^n, W_\beta^{1/2} S' x \neq 0} \frac{|x^T S'^T W_\beta S' \Lambda_\beta G_\beta S'^T W_\beta^{1/2} (I_\beta - I) W_\beta^{1/2} S' \Lambda_\beta x|}{x^T S'^T W_\beta S' x} \\
= \sup_{x \in \mathbb{R}^n, W_\beta^{1/2} S' x \neq 0} \frac{|x^T L_\beta \Lambda_\beta G_\beta S'^T W_\beta^{1/2} (I_\beta - I) W_\beta^{1/2} S' \Lambda_\beta L_\beta x|}{x^T S'^T W_\beta S' x} \\
= \sup_{x \in \mathbb{R}^n, W_\beta^{1/2} S' x \neq 0} \frac{|x^T S'^T W_\beta^{1/2} (I_\beta - I) W_\beta^{1/2} S' x|}{x^T S'^T W_\beta S' x} \\
\leq \epsilon \]
The lemma follows from the fact that
\[ \dim \left( \text{im} \left( W_\beta^{1/2} S' \right) \right) = n \]
and \( W_\beta^{1/2} S' x = 0 \) if and only if \( x = 0 \). \( \blacksquare \)
Also, we need the following lemma in [RV07] to bound the 2-norm of a random matrix.

Lemma 3.2.4 ([RV07]) Let $p$ be a probability distribution over a subset $\Omega \subseteq \mathbb{R}^d$ of $d$-dimensional vectors such that $\sup_{y \in \Omega} \|y\|_2 \leq M$ and $E_p[yy^T]_2 \leq 1$. Let $y_1 \ldots y_q$ be independently samples drawn from $p$. Then for any constant $\epsilon \in (0,1]$,

$$\Pr \left[ \left\| \frac{1}{q} \sum_{i=1}^q y_i y_i^T - E[yy^T] \right\|_2 > \epsilon \right] \leq 2 \exp \left( -\frac{\epsilon^2}{a^2} \right),$$

where $a = \min \left( CM \sqrt{\frac{\log(q)}{q}}, 1 \right)$ and $C$ is an absolute constant.

Now, we are ready to prove Theorem 3.2.1. We will first prove the following theorem which leads to the proof of Theorem 3.2.1.

Theorem 3.2.5 Let $L$ be the normalized Laplacian of $G$ and $\tilde{G}$ be the output of the algorithm $\text{SparsifyExactGreen}(G,q,\beta)$, where $q = \frac{4C^2n \log(n)}{\epsilon^2}$, $\epsilon \in (0,1]$ and $C \geq 1$ is an absolute constant. Then with probability at least $\frac{1}{2}$, we have

$$\left| x^T \tilde{L}_\beta x - x^T L_\beta x \right| \leq \epsilon x^T L_\beta x, \forall x \in \mathbb{R}^n,$$

where $L_\beta = \beta I + L = S^T W_\beta S'$ and $\tilde{L}_\beta = S^T W_\beta^{1/2} I_\beta W_\beta^{1/2} S'$.

Proof. Before applying Lemma 3.2.4, we observe that

$$\Lambda_\beta I_\beta \Lambda_\beta = \sum_{e \in E} R(e,e) \Lambda_\beta(e,\cdot)^T \Lambda_\beta(e,\cdot) + \sum_{v \in V} \Lambda_\beta(v,\cdot)^T \Lambda_\beta(v,\cdot),$$

$$\Lambda_\beta \Lambda_\beta = \sum_{e \in E} \Lambda_\beta(e,\cdot)^T \Lambda_\beta(e,\cdot) + \sum_{v \in V} \Lambda_\beta(v,\cdot)^T \Lambda_\beta(v,\cdot),$$

which implies that

$$\Lambda_\beta I_\beta \Lambda_\beta - \Lambda_\beta \Lambda_\beta = \sum_{e \in E} (R(e,e) - 1) \Lambda_\beta(e,\cdot)^T \Lambda_\beta(e,\cdot).$$
Now, let us consider the summation $\sum_{e \in E} R(e, e) \Lambda_\beta(e, \cdot)^T \Lambda_\beta(e, \cdot)$ and we have

$$\sum_{e \in E} R(e, e) \Lambda_\beta(e, \cdot)^T \Lambda_\beta(e, \cdot) = \sum_{e \in E} \left(\frac{\text{# of times } e \text{ is sampled}}{q p_e}\right) \Lambda_\beta(e, \cdot)^T \Lambda_\beta(e, \cdot)$$

$$= \frac{1}{q} \sum_{e \in E} \left(\frac{\text{# of times } e \text{ is sampled}}{\sqrt{p_e}}\right) \frac{\Lambda_\beta(e, \cdot)^T \Lambda_\beta(e, \cdot)}{\sqrt{p_e}}$$

$$= \frac{1}{q} \sum_{i=1}^{q} y_i y_i^T$$

where $y_1, \ldots, y_q$ are random vectors drawn independently with replacement from the distribution $p$ defined by setting

$$y = \frac{1}{\sqrt{p_e}} \Lambda_\beta(\cdot, e)$$

with probability $p_e$. We also need to bound the norm of the expectation of $yy^T$ as follows.

**Claim 3.2.6** $\|E_p[yy^T]\|_2 \leq 1$.

By definition, we have

**Proof.**

$$\|E_p[yy^T]\|_2 = \left\|\sum_{e \in E} \frac{1}{p_e} \Lambda_\beta(e, \cdot)^T \Lambda_\beta(e, \cdot)\right\|_2 = \left\|\sum_{e \in E} \Lambda_\beta(e, \cdot)^T \Lambda_\beta(e, \cdot)\right\|_2$$

and thus

$$\|E_p[yy^T]\|_2 = \sup_{x \in \mathbb{R}^{m+n}, x \neq 0} \sum_{e \in E} x^T \Lambda_\beta(e, \cdot)^T \Lambda_\beta(e, \cdot) x / x^T x.$$ 

Since for any $x \in \mathbb{R}^{m+n}$

$$\sum_{v \in V} x^T \Lambda_\beta(v, \cdot)^T \Lambda_\beta(v, \cdot) x \geq 0,$$
we have

\[
\left\| \mathbf{E}_p y^T y \right\|_2 = \sup_{x \in \mathbb{R}^{m+n}, x \neq 0} \frac{\sum_{e \in E} x^T \Lambda_{\beta}(e, \cdot)^T \Lambda_{\beta}(e, \cdot) x}{x^T x} \leq \sup_{x \in \mathbb{R}^{m+n}, x \neq 0} \frac{x^T (\sum_{e \in E} \Lambda_{\beta}(e, \cdot)^T \Lambda_{\beta}(e, \cdot) + \sum_{v \in V} \Lambda_{\beta}(v, \cdot)^T \Lambda_{\beta}(v, \cdot)) x}{x^T x} = \sup_{x \in \mathbb{R}^{m+n}, x \neq 0} \frac{x^T \Lambda_{\beta} x}{x^T x} = \sup_{x \in \mathbb{R}^{m+n}, x \neq 0} \frac{x^T \Lambda_{\beta} x}{x^T x} = \| \Lambda_{\beta} \|_2 = 1.
\]

The last equality above follows from the fact that the eigenvalues of \( \Lambda_{\beta} \) are all 1 or 0, and the claim follows.

An upper bound for the norm of \( y \) can be established as follows:

\[
\frac{1}{\sqrt{p_e}} \| \Lambda_{\beta}(e, \cdot) \|_2 = \frac{1}{\sqrt{p_e}} \sqrt{\Lambda_{\beta}(e, e)} = \sqrt{\sum_{e' \in E} w(e') g_{\beta}(e') \frac{w_e g_{\beta}(e)}{\beta}} = \sqrt{\sum_{e' \in E} \frac{w(e') g_{\beta}(e')}{\beta}}.
\]

Note that \( \Lambda_{\beta} = W_{\beta}^{1/2} S \mathcal{G}_{\beta} S^T W_{\beta}^{1/2} \) can be expressed as the following form

\[
\Lambda_{\beta} \triangleq \begin{bmatrix} \beta \mathcal{G}_{\beta} & \beta^{1/2} \mathcal{G}_{\beta} S^T W_{\beta}^{1/2} \\ \beta^{1/2} W_{\beta}^{1/2} S \mathcal{G}_{\beta} W_{\beta}^{1/2} & W_{\beta}^{1/2} S \mathcal{G}_{\beta} S^T W_{\beta}^{1/2} \end{bmatrix}_{(n+m) \times (n+m)}.
\]

This implies that

\[
\sum_{e \in E} \frac{w_e g_{\beta}(e)}{\beta} = \sum_{e \in E} \Lambda_{\beta}(e, e) = \text{Tr}(W_{\beta}^{1/2} S \mathcal{G}_{\beta} S^T W_{\beta}^{1/2}) = \text{Tr}(\Lambda_{\beta}) - \text{Tr}(\beta \mathcal{G}_{\beta}).
\]
Since $\text{Tr}(\Lambda_{\beta}) = n$, we have
\[
\frac{1}{\sqrt{p_e}} \|\Lambda_{\beta}(e, \cdot)\|_2 \leq \sqrt{n}.
\]
By setting $q = \frac{4C^2n\log(n)}{\epsilon^2}$, we get
\[
\min \left( CM \sqrt{\frac{\log(q)}{q}}, 1 \right) \leq C \sqrt{\frac{\epsilon^2 n \log(4C^2n \log(n))}{4C^2n \log(n)}} \leq \frac{\epsilon}{2} \tag{3.5}
\]
for some absolute constant $C > 0$. By applying the Rudelson and Vershynin’s lemma in [RV07] (Lemma 3.2.4), we completes the proof of the theorem.

Before applying Theorem 3.2.5 to prove Theorem 3.2.1, we still need the following two lemmas. We here consider $G$ as an unweighted graph first, although this can be easily extended to the general weighted graphs.

**Lemma 3.2.7** For any constant $\epsilon \in (0, 1]$, let sparsifier $\tilde{G}$ be the output of algorithm $\text{SparsifyExactGreen}(G, q, \beta)$, where $q = \frac{4C^2n(\beta + 2)\log(n)}{\epsilon^2}$ and $C \geq 1$ is an absolute constant. Then, with probability $1 - \frac{1}{n}$, we have
\[
|\text{Vol}_{\tilde{G}}(S) - \text{Vol}_{G}(S)| \leq \epsilon \text{Vol}_{G}(S), \forall S \subset V.
\]

**Proof.** Without loss of generality, we only consider $G$ as an unweighted graph, i.e., $w(e) = 1, \forall e \in E$. For a fixed subset $S \subset V$, let $E(S)$ be the set of edges with at least one endpoint in $S$. Let us consider the i.i.d. random variables $X_1, X_2, \ldots, X_q$ defined as
\[
X_i = \begin{cases} 
\frac{1}{p_e} & \text{with probability } p_e \text{ for } e \in E(S), \\
0 & \text{otherwise}, 
\end{cases}
\]
and a random variable
\[
V_S = \frac{1}{q} \sum_{i=1}^{q} X_i.
\]
We note that
\[
V_S = \sum_{e \in E(S)} \left( \frac{\# \text{ of times } e \text{ is sampled}}{q p_e} \right) = \sum_{e \in E(S)} \frac{\tilde{w}(e)}{w(e)} = \text{Vol}_{\tilde{G}}(S).
\]
Clearly, by the definition of random variable \( X_i \), we have

\[
E[X_i] = \sum_{e \in E(S)} p_e \frac{1}{p_e} = \text{Vol}_G(S),
\]

and

\[
E[X_i^2] = \sum_{e \in E(S)} p_e \frac{1}{p_e^2}
\]

\[
= \sum_{e \in E(S)} \frac{1}{p_e}
\]

\[
= \sum_{e \in E(S)} \frac{\sum_{e' \in E} w(e') g_\beta(e')}{w(e) g_\beta(e)}.
\]

By using the fact that \( \sum_{e \in E} w_e g_\beta(e) \leq \beta \text{Tr}(\Lambda_\beta) \leq \beta n \) and Lemma 3.1.1, we have

\[
E[X_i^2] = \sum_{e \in E(S)} \frac{\sum_{e' \in E} w(e') g_\beta(e')}{w(e) g_\beta(e)}
\]

\[
\leq \sum_{e \in E(S)} \frac{\beta n}{g_\beta(e)}
\]

\[
\leq \sum_{e=(u,v) \in E(S)} \frac{n \beta (\beta + 2) \min\{d(u), d(v)\}}{\beta}
\]

\[
\leq n(\beta + 2)\text{Vol}^2(S).
\]

By the Bernstein inequality, if i.i.d. random variables \( X_1, \ldots, X_q \) satisfy \( |X_i| \leq M \) for some constant \( M \), then for any \( \epsilon > 0 \)

\[
\Pr \left[ \left| \frac{1}{q} \sum_{i=1}^{q} X_i - \frac{1}{q} \sum_{i=1}^{q} E[X_i] \right| > \epsilon \frac{1}{q} \sum_{i=1}^{q} E[X_i] \right] \leq \exp \left( -\frac{\epsilon^2 (\sum_{i=1}^{q} E[X_i])^2}{\sum_{i=1}^{q} E[X_i^2] + \epsilon M} \right).
\]

In our setting, we observe that \( M = n(\beta + 2)\Delta_S \) because Lemma 3.1.1 and the fact that \( \sum_{e \in E} w_e g_\beta(e) \leq \beta n \). Here \( \Delta_S \) is the maximum degree of vertices in \( S \), i.e.

\[
\Delta_S = \max_{v \in S} d(v).
\]
Thus, we have
\[
\exp \left( -\frac{\epsilon^2}{\sum_{i=1}^q E[X_i]^2} \right) \leq \exp \left( -\frac{q^2 \epsilon^2 \text{Vol}^2(S)}{qn(\beta + 2) \text{Vol}^2(S) + \frac{ne(\beta + 2)\Delta_S}{3}} \right)
\]
\[
\leq \exp \left( -\frac{q \epsilon^2}{2(\beta + 2)n} \right)
\]
\[
= \exp \left( -\frac{2 \epsilon^2 C^2 \log(n)}{\epsilon^2} \right)
\]
\[
\leq \frac{1}{n^2}.
\]

Therefore, the probability of the event that there exists a vertex \( v \) such that
\[
|\text{Vol}_{\tilde{G}}(v) - \text{Vol}_G(v)| > \epsilon
\]
is less than \( \frac{1}{n} \) which implies that
\[
|\text{Vol}_{\tilde{G}}(S) - \text{Vol}_G(S)| \leq \sum_{v \in V} |\text{Vol}_{\tilde{G}}(v) - \text{Vol}_G(v)|
\]
\[
\leq \epsilon \sum_{v \in V} \text{Vol}_G(v)
\]
\[
\leq \epsilon \text{Vol}_G(S).
\]

The proof of the lemma is complete.

**Lemma 3.2.8** If the sparse graph \( \tilde{G} \) corresponding to a graph \( G \) satisfies two conditions:

(a) \( |x^T \tilde{L}_\beta x - x^T L_\beta x| \leq \epsilon x^T L_\beta x, \forall x \in \mathbb{R}^n; \)

(b) \( |\text{Vol}_{\tilde{G}}(S) - \text{Vol}_G(S)| \leq \epsilon \text{Vol}_G(S), \forall S \subset V. \)

Then, we have
\[
|h_{\tilde{G}}(S) - h_G(S)| \leq 2 \epsilon h_G(S) + \epsilon \beta, \forall S \subset V.
\]

**Proof.** Let \( L \) and \( \tilde{L} \) denote the Laplacians of graphs \( G \) and \( \tilde{G} \) respectively. Similarly, let \( D \) and \( \tilde{D} \) be the weighted degree matrices of the graphs \( G \) and \( \tilde{G} \) respectively. For a constant \( \beta \in (0, 1] \), let us consider \( L_\beta = \beta D + L \) and \( \tilde{L}_\beta = \beta \tilde{D} + \tilde{L} \).
From the definition of Cheeger ratio, we have

\[ |h_G(S) - h_G(S)| \]

\[ = \left| \frac{\chi_S^T \tilde{L}_\beta S \chi_S}{\chi_S^T D} \chi_S \chi_S^T D \chi_S \chi_S^T D \chi_S \right| \]

\[ \leq \left| \frac{\chi_S^T \tilde{L}_\beta S \chi_S}{\chi_S^T D} \chi_S \chi_S^T (D - \tilde{D}) \chi_S \chi_S^T D \chi_S \right| + \left| \frac{\chi_S^T \tilde{L}_\beta S \chi_S}{\chi_S^T D} \chi_S \chi_S^T D \chi_S \right| \]

where \( \chi_S \) denote the characteristic function on \( S \subset V \), satisfying \( \chi_S(v) = 1 \) if \( v \in S \) and \( \chi_S(v) = 0 \) otherwise.

Recall that \( L_\beta = S^T W_\beta S' \) and \( \tilde{L}_\beta = S^T W_\beta^{1/2} I_\beta W_\beta^{1/2} S' \). It is straightforward to verify that

\[ \frac{\chi_S^T L_\beta S \chi_S}{\chi_S^T D} = \frac{\chi_S^T D^{1/2} L_\beta D^{1/2} \chi_S}{\chi_S^T D \chi_S} \]

and

\[ \frac{\chi_S^T \tilde{L}_\beta S \chi_S}{\chi_S^T D} = \frac{\chi_S^T D^{1/2} \tilde{L}_\beta D^{1/2} \chi_S}{\chi_S^T D \chi_S} . \]

Thus, the first condition implies that

\[ \left| \frac{\chi_S^T \tilde{L}_\beta S \chi_S}{\chi_S^T D} \chi_S \chi_S^T D \chi_S \right| \leq \epsilon \left| \frac{\chi_S^T L_\beta S \chi_S}{\chi_S^T D \chi_S} \right| . \]

Now, by the second condition, we have

\[ |h_G(S) - h_G(S)| \]

\[ \leq (1 + \epsilon) \left| \frac{\chi_S^T L_\beta S \chi_S}{\chi_S^T D} \chi_S \chi_S^T (D - \tilde{D}) \chi_S \chi_S^T D \chi_S \right| + \epsilon \left| \frac{\chi_S^T L_\beta S \chi_S}{\chi_S^T D} \chi_S \chi_S^T D \chi_S \right| \]

\[ \leq (1 + \epsilon) \epsilon \left| \frac{\chi_S^T L_\beta S \chi_S}{\chi_S^T D} \chi_S \chi_S^T D \chi_S \right| + \epsilon \left| \frac{\chi_S^T L_\beta S \chi_S}{\chi_S^T D} \chi_S \chi_S^T D \chi_S \right| \]

\[ \leq 2\epsilon \left| \frac{\chi_S^T L_\beta S \chi_S}{\chi_S^T D} \chi_S \chi_S^T D \chi_S \right| \]

\[ \leq 2\epsilon \left| \frac{\chi_S^T D \chi_S}{\chi_S^T D} \chi_S \chi_S^T S \chi_S \chi_S^T D \chi_S \right| + \epsilon \beta \]

\[ = 2\epsilon h_G(S) + \epsilon \beta . \]

The lemma follows.

Now, we get all pieces ready for proving Theorem 3.2.1.
Proof. [of Theorem 3.2.1] To prove Theorem 3.2.1, we need to combine Lemma 3.2.7, Lemma 3.2.8 and Theorem 3.2.5. For any constant $\epsilon \in (0, 1]$, let $\tilde{G}$ be the output of the algorithm $\text{SparsifyExactGreen}(G, q, \beta)$, where $q = \frac{256C^2n \log(n)}{\epsilon^2}$, $\beta = \frac{1}{2}$, and $C \geq 1$ is an absolute constant. By Theorem 3.2.5 and Lemma 3.2.7, the conditions of Lemma 3.2.8 are satisfied with probability at least $\frac{1}{2}$. Note that $h_G(S) \leq 1$ and we have chosen $\beta$ to be $\frac{1}{2}$, thus algorithm $\text{SparsifyExactGreen}$ can be applied by using $O\left(\frac{n \log(n)}{\epsilon^2}\right)$ sampling. Furthermore, we have

$$|h_{\tilde{G}}(S) - h_{G}(S)| \leq \epsilon, \forall S \subset V.$$

This completes the proof of of Theorem 3.2.1. □

By choosing a different $\beta$, namely, $\beta = \frac{\phi}{2}$, we have the following theorem whose proof is quite similar to that of Theorem 3.2.1. We omit the proof here.

**Theorem 3.2.9** Given any constants $\epsilon, \phi \in (0, 1]$, let $\tilde{G}$ be the output of the algorithm $\text{SparsifyExactGreen}(G, q, \beta)$, where $q = \frac{256C^2n \log(n)}{\epsilon^2}$, $\beta = \frac{\phi}{2}$, and $C \geq 1$ is an absolute constant. Then with probability at least $\frac{1}{2}$, we have

$$|h_{\tilde{G}}(S) - h_{G}(S)| \leq \epsilon h_{G}(S)$$

for all $S \subset V$ satisfying $h_{G}(S) \geq \phi$.

### 3.3 Graph Sparsification Using Approximate Green Values

In Corollary 3.1.4, we show that how to compute approximate Green values $\tilde{g}_\beta(u, v)$ satisfying for all $(u, v) \in E$

$$(1 - \kappa)g_\beta(u, v) \leq \tilde{g}_\beta(u, v) \leq (1 + \kappa)g_\beta(u, v)$$

in $O\left(\frac{\Delta n}{\beta^3}\right)$ time, where $\kappa \in (0, 1]$ is any absolute constant (e.g., $\kappa = 0.001$). Instead of using exact Green values, we may use approximate Green values in algorithm $\text{SparsifyExactGreen}$. 
Recall that the approximate Green value \( \tilde{g}_\beta \) is a combination of four terms of approximate PageRank vectors \( \text{pr}'_{\beta,v} \)'s. If we choose the parameters for algorithm \text{ApproximatePR} as

\[
\text{pr}'_{\beta,v} = \text{ApproximatePR}(\chi_v, \beta, \frac{\beta_\kappa}{(2 + \beta)\Delta}).
\]

It is not difficult to verify that all results in Section 3.2 will change at most by a constant factor if we run the algorithm \text{SparsifyExactGreen} by using approximate Green values. The performance guarantee and the number of sampled edges in the Theorems 3.2.1 differ by at most a constant factor, although the computational complexity will increase to \( O(\frac{\Delta n}{\beta^2}) \). In order to further improve the running time, we will use several methods in the rest of this section.

In order to have better error estimate of approximate Green values, we need to improve the error estimate for approximate PageRank vectors in Theorem 2.3.2. We will use the strengthened approximate PageRank algorithm \text{SharpApproximatePR} and the dimension reduction technique in [SS08] to approximate the Green values by using these sharply approximate PageRank vectors produced by \text{SharpApproximatePR}.

First, we recall that

\[
g_{\beta}(u, v) = \beta(\chi_u - \chi_v)^T D^{-1/2} G_{\beta} D^{-1/2} (\chi_u - \chi_v)
\]

and thus

\[
g_{\beta}(u, v) = \beta(\chi_u - \chi_v)^T D^{-1/2} G_{\beta} L_{\beta} G_{\beta} D^{-1/2} (\chi_u - \chi_v)
\]

\[
= \beta \| W_{\beta} S' G_{\beta} D^{-1/2} (\chi_u - \chi_v) \|_2^2
\]

\[
= \frac{1}{\beta} \| W_{\beta} S' D^{1/2} [\beta D^{-1/2} G_{\beta} D^{-1/2}] (\chi_u - \chi_v) \|_2^2.
\]

Therefore, \( g_{\beta}(u, v) \)'s are just pairwise distances between vectors \( \{\Pi \chi_v^T\}_{v \in V} \) where

\[
\Pi = W_{\beta} S' D^{1/2} [\beta D^{-1/2} G_{\beta} D^{-1/2}].
\]

However, the dimension of the vectors in \( \{\Pi \chi_v^T\}_{v \in V} \) is \( (m + n) \). In order to reduce the computational complexity for computing these vectors, we project these vectors
into a lower dimensional space while preserving their pairwise distances by the following lemma in [Ach01].

**Lemma 3.3.1 ([Ach01])** Given vectors $x_1, \ldots, x_n \in \mathbb{R}^d$ and constants $\epsilon, \gamma > 0$, let $k_0 = \frac{c_\gamma \log(n)}{\epsilon^2}$ where $c_\gamma$ is a constant depending on $\gamma$. For integer $k \geq k_0$, let $R_{k \times d}$ be a random matrix where $\{R_{ij}\}$ are independent random variables with values $\pm \frac{1}{\sqrt{k}}$. Then with probability at least $1 - \frac{1}{n^\gamma}$, we have that for every pair $i, j$

$$(1 - \epsilon) \|x_i - x_j\|_2^2 \leq \|Rx_i - Rx_j\|_2^2 \leq (1 + \epsilon) \|x_i - x_j\|_2^2.$$

Now, we are ready to state our algorithm to approximate the Green values. Later, in order to analyze our algorithm $\text{ApproximateGreen}$, we will give a bound for $y_i$’s by Lemma 3.3.2.

$$(\tilde{g}_\beta) = \text{ApproximateGreen}(\beta, \epsilon, k)$$

1. Let $R_{k \times (n+m)} = [R_1, R_2]$ be a random matrix whose entries are independent random variables with values $\pm \frac{1}{\sqrt{k}}$, where $R_1$ is a $k \times n$ matrix and $R_2$ is a $k \times m$ matrix.

2. Let $Y = RW_\beta^{1/2}S'D^{1/2}$ and $\tilde{\Pi} = R\Pi$.

3. For $i = 1, \ldots, k$, do the following

   3.a Let $y_i$ be the $i$th row of $Y$ and $\tilde{z}_i$ be the $i$th row of $\tilde{\Pi}$.

   3.b Approximate $\tilde{z}_i$ by $\tilde{z}_i' = \text{SharpApproximatePR}(y_i, \beta, \frac{c_r}{n^r})$ where $r$ is a sufficient large constant.

4. Let $\tilde{\Pi}'$ be the approximated matrix for $\tilde{\Pi}$ whose rows are $\tilde{z}_1', \ldots, \tilde{z}_k'$. For all $(u, v) \in E$, return

   $$\tilde{g}_\beta(u, v) = \left\|\tilde{\Pi}'(\chi_u - \chi_v)\right\|_2^2.$$
Lemma 3.3.2 Given an integer $k$ and a random matrix whose entries are independent random variables with values $\pm \frac{1}{\sqrt{k}}$, with probability $1 - \frac{1}{n^2}$, we have for $1 \leq i \leq k$,

$$\|y_i\|_1 \leq c \sum_{v \in V} \sqrt{\log(n)d(v)}$$

where $c > 4$ is an absolute constant.

Proof. To simplify the argument, we consider a graph $G = (V, E, w)$ as an unweighted graph, i.e. $w(e) = 1, \forall e \in E$. Denote the $v$th element of vector $y_i$ to be $y_{i,v}$. From the definition of matrix $Y$, we have

$$Y = \beta^{1/2} R_1 D^{1/2} + R_2 W^{1/2} B.$$

We now fix a vertex $v \in V$ and let $X_0, X_1, \ldots, X_l$ be independent random variables with values $\pm \frac{1}{\sqrt{k}}$. We can rewrite $y_{i,v}$ as

$$y_{i,v} = \sqrt{\beta d(v)} X_0 + \sum_{l=1}^{d(v)} X_l.$$

By Chernoff’s inequality, we have

$$\Pr \left[ \left| \sum_{l=1}^{d(v)} X_l \right| > \sqrt{ka} \right] < 2 \exp \left( - \frac{a^2 k}{2d(v)} \right),$$

which is equivalent to

$$\Pr \left[ \left| \sum_{l=1}^{d(v)} X_l \right| > c \sqrt{\log(n)d(v)} \right] < 2 \exp \left( - \frac{c^2 \log(n)}{2} \right)$$

where $c \geq 4$ is a constant. Notice that $\beta \in (0,1]$ and $|X_0| = \frac{1}{\sqrt{k}}$. The lemma follows by applying the above inequality on the following event

$$\bigcup_{v \in V} \{ |y_{i,v}| > c \sqrt{\log(n)d(v)} \}.$$

By combining the above lemmas and Theorem 2.4.1, we have the follow theorem.
Theorem 3.3.3 Given any constant $\epsilon \in (0, 1]$, set $k = \frac{c \log(n)}{\epsilon^2}$ where $c > 4$ is an absolute constant. If $\beta = \Omega\left(\frac{1}{n^p}\right)$ for any fixed nonnegative integer $p$, algorithm ApproximateGreen$(\beta, \epsilon, k)$ will output approximate Green values $\tilde{g}_\beta(u, v)$ satisfying
\[|g_\beta(u, v) - \tilde{g}_\beta(u, v)| \leq \epsilon g_\beta(u, v), \forall (u, v) \in E\]
in $O\left(\frac{m \log^2(n)}{\beta \epsilon^2}\right)$ time.

Proof. To bound the running time, note that the running time of step 1 is $O\left(\frac{m \log(n)}{\epsilon}\right)$ since $R$ is a $k \times (n + m)$ random matrix. In step 2, we set
\[Y = RW_\beta^{1/2} S' D^{1/2}\]
and it only takes $O\left(\frac{m \log(n)}{\epsilon^2}\right)$ time since $S'$ has $O(m)$ nonzero entries and $W_\beta^{1/2}$ is a diagonal matrix.

In step 3, let $y_i$ be the $i$th row of $Y$ and $\tilde{z}_i$ be the $i$th row of $\tilde{\Pi}$ which is the matrix
\[\left[Y[\beta D^{-1/2} G_\beta D^{-1/2}]\right]_{k \times n}.
\]
Therefore, we have
\[\tilde{z}_i = y_i[\beta D^{-1/2} G_\beta D^{-1/2}]\]
and we can view $\tilde{z}_i$ as a scaled PageRank vector with seed vector $y_i$.

In Lemma 3.3.2, we proved that with probability at least $1 - \frac{1}{n}$, for $1 \leq i \leq k$,
\[\|y_i\|_1 \leq c \sum_{v \in V} \sqrt{\log(n) d(v)}.
\]
Without loss of generality, we may also assume that
\[c \sum_{v \in V} \sqrt{\log(n) d(v)} = O(m),\]
otherwise the graph is sufficient sparse. Thus, $\tilde{z}_i$ can be approximated by using algorithm SharpApproximatePR with arbitrary small absolute error, say, $\epsilon'$ and each call of SharpApproximatePR just takes $O\left(\frac{m \log(1/\epsilon')}{\beta}\right)$ time. By Lemma 3.1.1, $g_\beta(u, v) = \Omega\left(\frac{\beta}{n}\right)$ which implies that we only need to set $\epsilon' = \frac{\epsilon}{n^r}$ for some large enough but fixed constant $r$. This implies that each call of SharpApproximatePR.
will actually take $O\left(\frac{m \log(n)}{\beta}\right)$ time. Since there are $k = O\left(\frac{\log(n)}{\epsilon^2}\right)$ calls by Lemma 3.2.4, the total running time of step 3 is $O\left(\frac{m \log^2(n)}{\beta \epsilon^2}\right)$.

In step 4, since each column of $\tilde{\Pi}'$ has $k = O\left(\frac{\log(n)}{\epsilon^2}\right)$ entries and there are $m$ edges in the graph, the running time of step 4 is $O\left(\frac{m \log(n)}{\epsilon^2}\right)$. The lemma then follows.

Now, we summarize our results for sparsifying via approximate Green values in the following algorithm $\text{SparsifyApprGreen}$.

\begin{align*}
(\tilde{G}) &= \text{SparsifyApprGreen}(G, q, \beta, \epsilon) \\
1. & \text{ For all } e \in E, \text{ compute approximate Green values } \tilde{g}_\beta(e) \text{ by calling } \\
& \text{ApproximateGreen}(\beta, \kappa, k) \text{ where } \kappa = \frac{1}{2} \text{ and } k = \frac{c \log(n)}{\epsilon^2} \text{ for some } \\
& \text{absolute constant } c. \\
2. & \text{ Let } \tilde{G} \text{ be the output of algorithm } \text{SparsifyExactGreen}(G, q, \beta) \text{ with } \\
& \text{approximate Green values } \tilde{g}_\beta(e). \\
3. & \text{ Return } \tilde{G}.
\end{align*}

**Theorem 3.3.4** Given any constants $\epsilon, \phi \in (0, 1]$ and a graph $G$ on $n$ vertices with $m$ edges, set $q = \frac{256 C^2 n \log(n)}{\epsilon^2}$ where $C$ is an absolute constant, $\beta = \frac{\phi}{2}$, and let $\tilde{G}$ be the output of the algorithm $\text{SparsifyApprGreen}(G, q, \beta, \epsilon)$. Then with probability at least $\frac{1}{2}$, we have

(i) For all $S \subset V$ satisfying $h_G(S) \geq \phi$,

$$|h_{\tilde{G}}(S) - h_G(S)| \leq \epsilon h_G(S).$$

(ii) Algorithm $\text{SparsifyApprGreen}$ can be performed by using $O\left(\frac{m \log^2(n)}{\phi}\right)$ preprocessing time and $O\left(\frac{n \log(n)}{\epsilon^2}\right)$ sampling.

**Proof.** The proof is quite similar to the analysis of the Theorem 3.2.9 except that we use the sharp approximate Green values instead of exact Green values. Since
the approximate Green values used in algorithm $\text{SparsifyApprGreen}$ satisfying
\[(1 - \kappa)g_\beta(u, v) \leq \tilde{g}_\beta(u, v) \leq (1 + \kappa)g_\beta(u, v)\]
for some absolute constant $\kappa$, one can verify that all the results in Theorem 3.2.9 still hold (may change in a constant factor). Note that the only additional computational complexity is to compute approximate Green values which takes $O\left(\frac{m \log^2(n)}{\phi}\right)$ time. The theorem follows.

\section{Graph Partitioning using Approximate PageRank Vectors}

In this section, we combine the graph sparsification and partitioning algorithms using PageRank vectors to derive an improved partitioning algorithm. An application of our sparsification algorithm by PageRank is the balanced cut problem. For a given graph $G$, we first use our sparsification algorithm to preprocess the graph. Then we apply the local partitioning algorithm using PageRank vectors [ACL06, AP09] on the sparsifier. Since the local partitioning algorithm is a subroutine for the balance cut problem, we obtain a balanced cut algorithm with an improved running time. Spielman and Teng [ST04] gave a local partitioning algorithm which, for a fixed value of $\phi$, gives a cut with approximation ratio $O\left(\frac{1}{\phi^{1/2}} \log^{3/2} n\right)$ and of volume $v_\phi$ in $\tilde{O}\left(\frac{v_\phi \log^6(n)}{\phi^{5/4}}\right)$ time where $v_\phi$ is the largest volume of the set with Cheeger ratio $\phi$. Note that the constant $c$ above is quite large [ST04]. In [ACL06], PageRank vectors were used to derive a local partitioning algorithm with an improved running time $\tilde{O}\left(m + \frac{n}{\phi}\right)$. In [AP09], the running time was further reduced to $\tilde{O}\left(m + \frac{n}{\phi^{1/2}}\right)$ by preprocessing using sparsification algorithms in [BK96].

\textbf{Theorem 3.4.1 ([ACL06])} There exists an algorithm called $\text{PageRank-Partition}$ takes as input a parameter $\phi$ and a graph $G = (V, E, w)$ with $m$ edges and $n$ vertices, and has expected running time $O\left(\frac{m \log^4(n)}{\phi}\right)$. If there exists a set $C \subset V$ with Cheeger ratio $h_G(C) = O\left(\frac{\phi^2 \log^2(m)}{\log^3(n)}\right)$, then with high probability $\text{PageRank-Partition}$ finds a set $S$ such that $\text{Vol}(S) \geq \frac{\text{Vol}(C)}{2}$ and $h_G(S) \leq \phi$. 
Given an undirected, weighted graph $G = (V, E, w)$ with $n$ vertices and $m$ edges, we can apply algorithm $\text{SparsifyApprGreen}$ as a preprocess procedure on graph $G$ to get a sparsifier $\tilde{G}$ with only $O\left(\frac{n \log(n)}{\epsilon^2}\right)$ edges in time $O\left(\frac{m \log^2(n)}{\phi}\right)$ such that for all $S \subset V$ with $h_G(S) \geq \phi$,

$$|h_{\tilde{G}}(S) - h_G(S)| \leq \epsilon h_G(S).$$

Then, we use the algorithm $\text{PageRank-Partition}$ [ACL06] on graph $\tilde{G}$ instead of $G$ for balanced cut problem. The algorithm $\text{PageRank-Partition}$ has two inputs including a parameter $\phi$ and a graph with $m$ edges. As stated in Theorem 3.4.1 from [ACL06], the $\text{PageRank-Partition}$ algorithm has expected running time $O\left(\frac{m \log^4(m)}{\phi^2}\right)$. Furthermore, with high probability the $\text{PageRank-Partition}$ algorithms was shown to be able to find a set $S$, if exists, such that $\text{Vol}(S) \geq \frac{\text{Vol}(C)}{2}$ and $h_G(S) \leq \phi$. This can be summarized in the following algorithm $\text{Partition}$ and Theorem 3.4.2.

\[
(S) = \text{Partition}(G, \phi, \epsilon)
\]

1. Set $\beta = \frac{\phi}{2}, q = \frac{Cn \log(n)}{\epsilon^2}$ for some constant $C$.

2. Let $\tilde{G}$ be the output of $\text{SparsifyApprGreen}(G, q, \beta, \epsilon)$.

3. Let $S$ be the output of $\text{PageRank-Partition}(\tilde{G}, \phi)$.

4. Return $S$.

**Theorem 3.4.2** Given an undirected, weighted graph $G = (V, E, w)$ with $n$ vertices and $m$ edges, constant $\phi, \epsilon \in (0, 1]$. Algorithm $\text{Partition}(G, \phi, \epsilon)$, with probability $\frac{1}{2}$, can preprocess graph $G$ in $O\left(\frac{m \log^2(n)}{\phi}\right)$ time to obtain a sparse graph $\tilde{G}$ with $O\left(\frac{n \log(n)}{\epsilon^2}\right)$ edges such that for all $S \subset V$ satisfying $h_G(S) \geq \phi$,

$$|h_{\tilde{G}}(S) - h_G(S)| \leq \epsilon h_G(S).$$

Furthermore, it calls algorithm $\text{PageRank-Partition}$ with input parameter $\phi$ and the graph $\tilde{G}$ and runs in expected running time $O\left(\frac{n \log^4(n)}{\phi^2 \epsilon^2}\right)$. If there exists a set $C$ with $h_G(C) = O\left(\frac{\phi^2}{\log^2 n}\right)$, then with high probability the $\text{PageRank-Partition}$ algorithm finds a set $S \subset V$ such that $\text{Vol}(S) \geq \frac{\text{Vol}(C)}{2}$ and $h_G(S) \leq \phi$. 
3.5 Acknowledgement

Material in this chapter has appeared in the following article:

Chapter 4

Connection Graphs

In this chapter, we consider a generalization of graphs, called connection graphs, in which each edge of the graph is associated with a weight and also a rotation (which is a linear orthogonal transformation acting on a \( d \)-dimensional vector space for some positive integer \( d \)). The adjacency matrix and the discrete Laplace operator are acting on the space of vector-valued functions (instead of the usual real-valued functions) and therefore can be represented by matrices of size \( dn \times dn \) where \( n \) is the number of vertices in the graph.

Connection graphs arise in numerous applications, in particular for data and image processing involving high-dimensional data sets. To quantify the affinities between two data points, it is often not enough to use only a scalar edge weight. For example, if the high-dimensional data set can be represented or approximated by a low-dimensional manifold, the patterns associated with nearby data points are likely to be related by certain rotations [StWss]. There are many recent developments of related research in cryo-electron microscopy [HS11, Sin11, SZSH11], angular synchronization of eigenvectors [CLSss, Sin11, SZSH11] and vector diffusion maps [StWss]. In many areas of machine learning, high-dimensional data points in general can be treated by various methods, such as the Principle Component Analysis [Jol05], to reduce vectors into some low-dimensional space and then use the connection graph with rotations on edges to provide the additional information for proximity. In computer vision, there has been a great deal of recent work dealing with trillions of photos that are now available on the web [ASS+09].
Feature matching techniques [Low99] can be used to derive vectors associated with the images. Then information networks of photos can be built which are exactly connection graphs with rotations corresponding to the angles and positions of the cameras in use. The use of connection graphs can be further traced to earlier work in graph gauge theory for computing the vibrational spectra of molecules and examining the spins associated with vibrations [CS92].

4.1 The Definition of a Connection Graph

For positive integers \( m, n \) and \( d \), we consider a family of matrices, denoted by \( \mathcal{F}(m, n, d; \mathbb{R}) \) consisting of all \( md \times nd \) matrices with real-valued entries. A matrix \( A \in \mathcal{F}(m, n, d; \mathbb{R}) \) can also be viewed as a \( m \times n \) matrix whose entries are represented by \( d \times d \) matrices (or blocks) denoted by \( A(i, j) \) for \( 1 \leq i \leq m, 1 \leq j \leq n \). A rotation matrix is a matrix that is used to perform a rotation in Euclidean space. Namely, a rotation \( O \) is a square matrix, with real entries, which can be characterized as orthogonal matrix satisfying

\[
O^T = O^{-1}
\]

and

\[
\det(O) = 1.
\]

All rotation matrices of dimension \( d \) is known as the special orthogonal group \( \text{SO}(d) \). It is easy to check that all eigenvalues of a rotation \( O \) are of norm 1. Furthermore, the following property of rotation matrix will be useful throughout this section.

**Proposition 4.1.1 ([GVL96])** If \( d \) is a positive odd number and rotation \( O \in \text{SO}(d) \) (e.g. \( \text{SO}(3) \)), then one of eigenvalues of \( O \) is 1.

Throughout this paper, we will use \( M^+ \) to denote the Moore-Penrose pseudoinverse of a matrix \( M \). For a matrix \( M \), the pseudoinverse of \( M \) is defined as the unique matrix \( M^+ \) satisfying all of the following four criteria [GVL96, Pen55].

(i) \( MM^+M = M \);
(ii) $M^+M^+ = M^+$;
(iii) $(MM^+)^* = (MM^+)$;
(vi) and $(M^+M)^* = M^+M$.

4.2 The Connection Laplacian of a Connection Graph

Recall that $G = (V, E, w)$ is an undirected graph with vertex set $V$, edge set $E$ and edge weights $w(u, v) = w(v, u) > 0$ for edges $(u, v)$ in $E$. Suppose each oriented edge $(u, v)$ is associated with a rotation matrix $O_{uv} \in SO(d)$ satisfying

$$O_{uv}O_{vu} = I_{d \times d},$$

i.e. $O_{vu} = O_{uv}^T$. Let $O$ denote the set of rotations associated with all oriented edges in $G$. The connection graph, denoted by $\mathbb{G} = (V, E, O, w)$, has $G$ as the underlying graph. The connection matrix $A$ of $\mathbb{G}$ is defined by:

$$A(u, v) = \begin{cases} w(u, v)O_{uv} & \text{if } (u, v) \in E, \\ 0_{d \times d} & \text{if } (u, v) \notin E \end{cases}$$

where $0_{d \times d}$ is the zero matrix of size $d \times d$. In other words, for $|V| = n$, we view $A \in \mathcal{F}(n, n, d; \mathbb{R})$ as a block matrix where each block is either a $d \times d$ rotation matrix $O_{uv}$ multiplied by a scalar weight $w(u, v)$, or a $d \times d$ zero matrix. The matrix $A$ is symmetric as $O_{uv}^T = O_{vu}$ and $w(u, v) = w(v, u)$. The diagonal matrix $D \in \mathcal{F}(n, n, d; \mathbb{R})$ is defined by the diagonal blocks

$$D(u, u) = d(u)I_{d \times d}$$

for $u \in V$. Here $d(u)$ is the weighted degree of $u$ in $G$, i.e., $d(u) = \sum_{(u, v) \in E} w(u, v)$.

The connection Laplacian $L \in \mathcal{F}(n, n, d; \mathbb{R})$ of a graph $\mathbb{G}$ is the block matrix

$$L = D - A.$$
Recall that for any orientation of edges of the underlying graph $G$ on $n$ vertices and $m$ edges, the combinatorial Laplacian $L$ can be written as

$$L = B^T W B$$

where $W$ is a $m \times m$ diagonal matrix with $W(e,e) = w(e)$, and $B$ is the edge-vertex incident matrix of size $m \times n$ such that

$$B(e,v) = \begin{cases} 
1 & \text{if } v \text{ is } e \text{'s head}, \\
-1 & \text{if } v \text{ is } e \text{'s tail}, \\
0 & \text{otherwise}.
\end{cases}$$

A useful observation for the connection Laplacian is the fact that it can be written in a similar form. Let $B \in \mathcal{F}(m,n,d; \mathbb{R})$ be the block matrix given by

$$B(e,v) = \begin{cases} 
O_{uv} & \text{if } v \text{ is } e \text{'s head}, \\
-I_{d \times d} & \text{if } v \text{ is } e \text{'s tail}, \\
0_{d \times d} & \text{otherwise}.
\end{cases}$$

Also, let the block matrix $W \in \mathcal{F}(m,m,d; \mathbb{R})$ denote a diagonal block matrix given by

$$W(e,e) = w(e)I_{d \times d}.$$ 

Then, we have the following useful lemma.

**Lemma 4.2.1** Suppose $G = (V,E,O,w)$ is a connection graph. Then

(i) For any orientation of edges on graph $G$, the connection Laplacian $L$ of $G$ can be written as

$$L = B^T W B.$$ 

(ii) For any function $f : V \rightarrow \mathbb{R}^d$, we have

$$f^T L f = \sum_{(u,v) \in E} w(u,v) \|O_{uv} f(u) - f(v)\|^2_2. \quad (4.1)$$

where $f$ here can be also regarded as a vector of dimension $nd$.  

(iii) \( L \) has a complete set of real eigenfunctions \( \phi_1, \phi_2, \ldots, \phi_n \) and corresponding real eigenvalues \( 0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \). Furthermore, \( \lambda_i = 0 \) if and only if \( O_{uv} \phi_i(u) = \phi_i(v) \) for all \( (u,v) \in E \).

**Proof.** To verify (i), for an edge \( e = (u,v) \in E \), we may assume the orientation of \( e \) is from \( u \) to \( v \) without loss of generality and we note that for the off-diagonal blocks of \( L \)

\[
\mathbb{B}(e,u)^T \mathbb{W}(e,e) \mathbb{B}(e,v) = -\mathbb{W}(e,e) O_{uv}
\]

\[
= -w(u,v) O_{uv}
\]

\[
= L_G(u,v),
\]

and similarly

\[
\mathbb{B}(e,v)^T \mathbb{W}(e,e) \mathbb{B}(e,u) = O_{uv}^T \mathbb{W}(e,e)(-I)
\]

\[
= -O_{vu} \mathbb{W}(e,e)
\]

\[
= -w(v,u) O_{vu}
\]

\[
= L_G(v,u).
\]

In the meanwhile, for the diagonal blocks of \( L \), we observe that for a fixed \( v \in V \)

\[
\mathbb{B}(\cdot, v)^T \mathbb{W} \mathbb{B}(\cdot, v) = \sum_{(u,v) \in E : v \text{ is head}} w(u,v) O_{uv}^T O_{uv} + \sum_{(u,v) \in E : v \text{ is tail}} w(u,v)(-I)(-I)
\]

\[
= \sum_{(u,v) \in E : v \text{ is head}} w(u,v) I + \sum_{(u,v) \in E : v \text{ is tail}} w(u,v) I
\]

\[
= \mathbb{B}(\cdot, v)^T \mathbb{W} \mathbb{B}(\cdot, v)
\]

\[
= \mathbb{D}(v, v)
\]

\[
= L(v, v).
\]

Thus, for the diagonal block of \( L \), we have \( \mathbb{B}(\cdot, v)^T \mathbb{W} \mathbb{B}(\cdot, v) = L(v, v) \).

(ii) is a direct consequence of (i) as one may observe that for a fixed edge \( e = (u,v) \),

\[
\mathbb{B}f = O_{uv}f(u) - f(v).
\]
Thus,

\[
f^T \mathbb{L} f = f^T \mathbb{L} f \\
= (fB)^T W(Bf) \\
= \sum_{(u,v) \in E} w(u,v) \|O_{uv} f(u) - f(v)\|^2_2.
\]

For (iii), \( \mathbb{L} \) is symmetric and therefore has real eigenfunctions and real eigenvalues. The spectral decompositions of \( \mathbb{L} \) is given by

\[
\mathbb{L}_G(u,v) = \sum_{i=1}^{nd} \lambda_i \phi_i(u) \phi_i(v)^T.
\]

By Equation (4.1), \( \lambda_i \geq 0 \) and \( \lambda_i = 0 \) if and only if \( O_{uv} \phi_i(u) = \phi_i(v) \) for all \( \{u,v\} \in E \) and the lemma follows.

### 4.3 The Consistency of a Connection Graph

For a connection graph \( G = (V, E, O, w) \), we say \( G \) is consistent if for any cycle \( c = (v_1, v_2, \ldots, v_k, v_1) \) the product of rotations along this cycle is the identity rotation, i.e.

\[
O_{v_kv_1} \prod_{i=1}^{k-1} O_{v_iv_{i+1}} = I_{d \times d}.
\]

In other words, for any two vertices \( u \) and \( v \), the products of rotations along different paths from \( u \) to \( v \) are the same. In the following theorem, we give a characterization for a consistent connection graph by using the eigenvalues of the connection Laplacian.

**Theorem 4.3.1** Suppose connected graph \( G \) is the underlying graph of a connection graph \( G = (V, E, O, w) \). Then \( G \) is consistent if and only if the eigenvalues of \( \mathbb{L} \) are the \( d \) copies of eigenvalues of \( L \) where \( \mathbb{L} \) is the connection Laplacian of \( G \), \( L \) is the combinatorial Laplacian of \( G \) and \( d \) is the dimension of rotations.

**Proof.** (\( \implies \)). For a fixed vertex \( u \in V \) and an arbitrary \( d \)-dimensional vector \( \hat{x} \), we can define a function \( \hat{f} : V \to \mathbb{R}^d \), by defining \( \hat{f}(u) = \hat{x} \) initially. Then
we assign \( \widehat{f}(v) = O_{uv} \widehat{f}(u) \) for all the neighbors \( v \) of \( u \). Since \( G \) is connected and \( G \) is consistent, we can continue the assigning process to all vertices without any confliction until all vertices are assigned. The resulting function \( \widehat{f} : V \to \mathbb{R}^d \) satisfies
\[
\widehat{f}^T \mathbb{L} \widehat{f} = \sum_{(u,v) \in E} w(u,v) \left\| O_{uv} \widehat{f}(u) - \widehat{f}(v) \right\|_2^2 = 0.
\]
Therefore \( \mathbb{L} \) has an eigenspace of dimension \( d \) for the eigenvalue 0 as the initial vector \( \widehat{f}(u) = \hat{x} \) can be arbitrary, and it has \( d \) orthogonal eigenfunctions \( \widehat{f}_1, \ldots, \widehat{f}_d \) corresponding to eigenvalues 0.

Now, let us consider the underlying graph \( G \) of the consistent graph \( G \). Let \( f_i : V \to \mathbb{R} \) denote the eigenfunctions of \( L \) corresponding to the eigenvalue \( \lambda_i \) for \( 1 \leq i \leq n \) respectively. A key observation is the following claim. Our proof of this direction will follow directly from the following claim.

**Claim 4.3.2** Functions \( f_i \otimes \widehat{f}_k : V \to \mathbb{R}^d \) for \( i \in [n], k \in [d] \) are the orthogonal eigenfunctions of \( \mathbb{L} \) corresponding to eigenvalue \( \lambda_i \) where \( f_i \otimes \widehat{f}_k(v) = f_i(v) \widehat{f}_k(v) \).

**Proof.** First, we need to verify that functions \( f_i \otimes \widehat{f}_k \) are eigenfunctions of \( \mathbb{L} \). We note that
\[
[\mathbb{L} f_i \otimes \widehat{f}_k](u) = d(u) f_i \otimes \widehat{f}_k(u) - \sum_{(u,v) \in E} w(u,v) O_{uv} f_i \otimes \widehat{f}_k(v)
\]
\[
= d(u) f_i(u) \widehat{f}_k(u) - \sum_{(u,v) \in E} w(u,v) O_{uv} f_i(v) \widehat{f}_k(v)
\]
\[
= \widehat{f}_k(u) \left( d(u) f_i(u) - \sum_{(u,v) \in E} w(u,v) f_i(v) \right).
\]
Since \( f_i \) is an eigenfunction of \( L \) corresponding to the eigenvalue \( \lambda_i \), we have \( L f_i = \lambda_i f_i \), i.e.
\[
\left( d(u) f_i(u) - \sum_{(u,v) \in E} w(u,v) f_i(v) \right) = \lambda_i f_i(u).
\]
Thus,
\[
[\mathbb{L} f_i \otimes \widehat{f}_k](u) = \lambda_i \widehat{f}_k(u) f_i(u) = \lambda_i f_i \otimes \widehat{f}_k(u)
\]
and \( f_i \otimes \hat{f}_k, 1 \leq i \leq n, 1 \leq k \leq d \) are the eigenfunctions of \( \mathbb{L} \).

To prove the orthogonality of \( f_i \otimes \hat{f}_k \)'s, we note that if \( k \neq l \),
\[
\langle f_i \otimes \hat{f}_k, f_j \otimes \hat{f}_l \rangle = \sum_v f_i(v) f_j(v) \langle \hat{f}_k(v), \hat{f}_l(v) \rangle = 0
\]
since \( \langle \hat{f}_k(v), \hat{f}_l(v) \rangle = 0 \) for \( k \neq l \). For the case of \( k = l \) but \( i \neq j \), we have
\[
\langle f_i \otimes \hat{f}_k, f_j \otimes \hat{f}_l \rangle = \sum_v f_i(v) f_j(v) \langle \hat{f}_k(v), \hat{f}_k(v) \rangle = 0
\]
because of \( \langle f_i, f_j \rangle = 0 \) for \( i \neq j \). The claim is proved. \( \blacksquare \)

\((\Leftarrow)\). Suppose \( \hat{f}_1, \ldots, \hat{f}_d \) are \( d \) orthogonal eigenfunctions of \( \mathbb{L} \) corresponding to the eigenvalue 0, i.e.
\[
\hat{f}_i^T \mathbb{L} \hat{f}_i = \sum_{(u,v) \in E} w(u,v) \left\| O_{uv} \hat{f}_i(u) - \hat{f}_i(v) \right\|^2 = 0.
\]
for all \( 1 \leq i \leq d \). Thus, one may verify that \( \hat{f}_1(v), \ldots, \hat{f}_d(v) \) must be a linear orthogonal basis of \( \mathbb{R}^d \) for all \( v \in V \) satisfying for all \( (u,v) \in E \)
\[
O_{uv} \hat{f}_i(u) = \hat{f}_i(v).
\]
This implies that for any pair of vertices \( u, v \in V \),
\[
\prod_{(v_i, v_{i+1}) \in p} O_{v_i,v_{i+1}} \hat{f}_i(u) = \hat{f}_i(v)
\]
where \( p \) is any path from \( u \) to \( v \). Now, suppose there are two vertices \( u \) and \( v \) and two different path from \( u \) to \( v \) such that the products of rotations along different paths, say \( O_1 \) and \( O_2 \), are different. There must be a vector \( g(u) \in \mathbb{R}^d \) such that \( O_1 g(u) \neq O_2 g(u) \). However, as \( \hat{f}_1(v), \ldots, \hat{f}_d(v) \) form a linear orthogonal basis of \( \mathbb{R}^d \) for all \( v \) and for \( 1 \leq i \leq d \)
\[
O_1 \hat{f}_i(u) = \hat{f}_i(v) = O_2 \hat{f}_i(u),
\]
which is a contradiction. Thus, the theorem is proved. \( \blacksquare \)
4.4 Random walks on a Connection graph

Consider the underlying graph $G$ of a connection graph $G = (V, E, O, w)$ and a starting vertex. A random walker randomly selects one of its neighbors and move to this neighbor; then it select a neighbor of this vertex at random and move to it etc. The (random) sequence of vertices selected this way is a random walk on the graph. The random walk on a graph $G$ is therefore defined by the transition probability matrix $P$ where

$$P(u, v) = \frac{w(u, v)}{d(v)}$$

denotes the probability of moving to a neighbor $v$ at a vertex $u$. We may rewrite the transition matrix as

$$P = AD^{-1},$$

where $A$ is the weighted adjacency matrix of $G$ and $D$ is the diagonal matrix of weighted degree.

In a similar way, we can define a random walk on the connection graph $G$ by setting the transition probability matrix

$$P = AD^{-1}.$$

While $P$ acts on the space of real-valued functions, $P$ acts on the space of vector-valued functions $f : V \rightarrow \mathbb{R}^d$.

Given a connection graph $G$, a $d$-dimensional vector and a starting vertex, the random walk on $G$ is that a random walker randomly selects one of its neighbors and move to this neighbor while rotate the $d$-dimensional vector according the rotation on that edge; then it select a neighbor of this vertex at random and move to it while rotate the $d$-dimensional vector according the rotation on that edge etc. The (random) sequence of vertices selected this way is a random walk on the graph.

**Theorem 4.4.1** Suppose $G$ is consistent. Then for any positive integer $t$, any vertex $u \in V$ and any function $\tilde{s} : V \rightarrow \mathbb{R}^d$ satisfying $\|\tilde{s}(v)\|_2 = 0$ for all $v \in V \setminus \{u\}$, we have

$$\|\tilde{s}(u)\|_2 = \sum_v \| [P^t \tilde{s}] (v) \|_2.$$
Proof. The proof of this theorem is straightforward from the assumption that \( G \) is consistent. For \( \hat{p} = \mathbb{P}^t \hat{s} \), note that \( \hat{p}(v) \) is the summation of all \( d \) dimensional vectors resulted from rotating \( \hat{s}(u) \) via rotations along all possible paths of length \( t \) from \( u \) to \( v \). Since \( G \) is consistent, the rotated vectors arrive at \( v \) via different paths are positive multiples of the same vector. Also the rotations maintain the 2-norm of vectors. Thus, \( \frac{\|\hat{p}(v)\|_2}{\|\hat{s}(u)\|_2} \) is simply the probability that a random walk in \( G \) arriving at \( v \) from \( u \) after \( t \) steps, i.e.

\[
\frac{\|\hat{p}(v)\|_2}{\|\hat{s}(u)\|_2} = P^t(u, v)
\]

The theorem follows. \( \blacksquare \)

4.5 Acknowledgement

Material in this chapter has appeared in the following article, accepted for publication:

Chapter 5

Connection Resistances

Many information networks arising from massive data sets exhibit the small world phenomenon. Consequently the usual graph distance is no longer very useful. It is crucial to have the appropriate metric for expressing the proximity between two data points. Previously, various notions of diffusion distances have been defined [StWss] and used for manifold learning and dimension reduction. Here we consider a basic notion, the connection resistance which is a generalization of the usual effective resistance. The effective resistance plays a major role in electrical network theory and can be traced back to the classical work of Kirchhoff [Kir47].

The connection resistance can then be used to derive edge ranking in the connection graph. In the applications to cryo-electron microscopy, the edge ranking can help eliminate the superfluous or erroneous edges that appear because of various “noises”.

In this chapter, we will first describe electrical flows and effective resistances in electrical network in Section 5.1, which will serve as a starting point of our intuition for connection resistances. In the section 5.2, we will define the connection resistance and exam connection resistances for several examples, e.g. paths and trees.
5.1 The Effective Resistance in an Electrical Network

If we ignore rotations on edges, we may consider the underlying graph $G$ of a connection graph $\mathcal{G} = (V, E, O, w)$ as an electrical network. That is, each edge $e \in E$ represents an electrical resistance of $\frac{1}{w(e)}$ or conductance of $w(e)$. We will adopt the notations for the description of the electrical network from [SS08] to describe electrical flows on graphs, which will serve as a starting point of our intuition for connection resistance.

There are two classical laws in the electrical network theory which are the Kirchoff’s current law and the Ohm’s law [Kir47]. The Kirchoff’s current law states that the sum of the currents entering into a vertex is equal to the total amount of currents injected at this vertex. The Ohm’s law [Kir47] states that the current flow in an edge is equal to the potential difference across its ends times its conductance.

We begin by orienting the edges of $G$ arbitrarily. Let $i_V$ be a vector of currents injected at the vertices $V$, let $i_E(e)$ be the currents induced in the edges (where positive sign represents that current direction coincides the edge direction), and let $v(u)$ be the potentials induced at the vertices. By the Kirchoff’s current law we have

$$B^T i_E = i_V. \tag{5.1}$$

In the meanwhile, by the Ohm’s law we obtain

$$i_E = W B v, \tag{5.2}$$

where $W$ is a $m \times m$ diagonal matrix satisfying $W(e, e) = w(e)$. Combining the above two equations gives us

$$i_V = B^T (WBv) = Lv.$$

Since $\ker(L) = \{\lambda \mathbf{1} : \lambda \in \mathbb{R}\}$, this implies that if the total amount of current injected is equal to the total amount extracted i.e. $i_V \perp \ker(L)$, then we can write

$$v = L^+ i_V.$$
In the electrical network theory, the effective resistance \( R_{\text{eff}}(v, u) \) between two vertices \( u \) and \( v \) is defined as the potential difference induced between them when a unit current is injected at one and extracted at the other. To inject and extract a unit current across the endpoints of an edge \( e = (u, v) \), we need to set

\[
i_V = (\chi_u - \chi_v) = B^T \chi_e
\]

which also satisfies that \( i_V \perp \ker(L) \). Thus, we have \( v = L^+ B^T \chi_e \) and by the definition of effective resistance \( R_{\text{eff}}(v, u) \), we get

\[
R_{\text{eff}}(v, u) = v(v) - v(u) = (\chi_v - \chi_u)^T v = \chi_e^T B L^+ B^T \chi_e = L^+_{u,u} + L^+_{v,v} - 2L^+_{u,v}.
\]

(5.3)

### 5.2 The Connection Resistance in a Connection Graph

Motivated by the definition of effective resistances in electrical network theory and its equivalent form in Equation (5.3), we consider the following block matrix

\[
\Pi = BL^+ B^T \in \mathcal{F}(m, m; \mathbb{R})
\]

for a connection graph \( G \) which plays a similar role as matrix

\[
BL^+ B^T
\]

in the third line of Equation (5.3). Here, \( L^+ \) is the pseudo-inverse of \( L \) and \( L \) is the connection Laplacian of \( G \). Note that for a matrix \( M \), the pseudo-inverse of \( M \) is defined as the unique matrix \( M^+ \) satisfying the following four criteria [GVL96, Pen55].

(i) \( MM^+ M = M \);

(ii) \( M^+ M^+ = M^+ \);
(iii) \((MM^+)^* = (MM^+)\);

(vi) and \((M^+M)^* = M^+M\).

Inspired by Equation (5.3), with the block matrix \(\Pi\), we are ready to define the connection resistance \(R_{\text{eff}}(e)\) for an edge \(e \in E\) as

\[R_{\text{eff}}(v, u) = \|\Pi(e, e)\|_2.\]  

(5.4)

Note that block \(\Pi(e, e)\) is a \(d \times d\) matrix. If \(d = 1\) or all orthogonal transformations are identity transformation, i.e.

\[O_e = I_{d \times d}, \forall e \in E,\]

then it can be shown that the connection resistance \(R_{\text{eff}}(u, v)\) between vertices \(u\) and \(v\) is reduced to the effective resistance \(R_{\text{eff}}(u, v)\) between vertices \(u\) and \(v\) in the underlying graph \(G\). In general, the connection resistance between the endpoints of an edge \(e = (u, v)\) is not necessarily equal to its effective resistance in the underlying graph \(G\). We will investigate the relation between effective resistances in the underlying graphs and connection resistance for some family of connection graphs.

Let us first look into a very simple case which is just a simple edge with two vertices \(u\) and \(v\).

**Proposition 5.2.1** Suppose \(G_{(u,v)}\) is a simple edge \((u, v)\) with edge weight \(w_{uv}\) and a rotation matrix \(O_{uv} \in SO(d)\) for some odd number \(d\). Then

\[R_{\text{eff}}(u, v) = R_{\text{eff}}(u, v).\]

**Proof.** Without loss of generality, we may assume \(w_{uv} = 1\). Then, we can write the connection Laplacian of \(G_{(u,v)}\) as

\[
L_{G_{(u,v)}} = \begin{bmatrix}
I_{d \times d} & -O_{uv} \\
-O_{uv}^T & I_{d \times d}
\end{bmatrix} = \begin{bmatrix}
U_{uv} & 0_{d \times d} \\
0_{d \times d} & V_{uv}
\end{bmatrix} \begin{bmatrix}
I_{d \times d} & -\Sigma_{uv} \\
-\Sigma_{uv}^T & I_{d \times d}
\end{bmatrix} \begin{bmatrix}
U_{uv}^T & 0_{d \times d} \\
0_{d \times d} & V_{uv}^T
\end{bmatrix}.
\]
where
\[ O_{uv} = V_{uv} \Sigma_{uv}^T U_{uv} \]
is the singular value decomposition of \( O_{uv} \). Thus, one can verify that the eigenvalues of \( \mathbb{L}_{G(u,v)} \) are 0 of multiplicities \( d \) and 2 of multiplicities \( d \), respectively. Since \( \mathbb{L}_{G(u,v)} \) is symmetric, this implies that the pseudoinverse of \( \mathbb{L}_{G(u,v)} \) is simply
\[
\mathbb{L}_{G(u,v)}^+ = \frac{1}{4} \left[ \begin{array}{cc}
I_{d \times d} & -O_{uv} \\
-O_{uv}^T & I_{d \times d}
\end{array} \right]
\]
Thus, \( R_{\text{eff}}(u, v) \) in graph \( G_w(u,v) \) is the maximum eigenvalue of the following matrix
\[
\frac{1}{4} \left[ \begin{array}{cc}
O_{uv} & -I_{d \times d} \\
-I_{d \times d} & O_{uv}
\end{array} \right] \left[ \begin{array}{cc}
I_{d \times d} & -O_{uv} \\
-O_{uv}^T & I_{d \times d}
\end{array} \right] \left[ \begin{array}{cc}
O_{uv}^T \\
-O_{uv}^T
\end{array} \right]
\]
\[
= \frac{1}{4} (2I_{d \times d} + O_{uv}^2 + O_{vu}^2)
\]
\[
= \frac{1}{4} (O_{uv} + O_{vu}) (O_{uv} + O_{vu}).
\]
Since \( d \) is odd, \( O_{uv} \) has eigenvalue 1 whose corresponding eigenfunction is also the eigenfunction of \( O_{uv} + O_{vu} \). Thus, we have \( \|O_{uv} + O_{vu}\|_2 = 2 \) and
\[
\|(O_{uv} + O_{vu})(O_{uv} + O_{vu})\|_2 = 4.
\]
This implies that
\[ R_{\text{eff}}(u, v) = \frac{1}{4} \times 4 = 1 = R_{\text{eff}}(u, v) \]
and the proposition follows.

**Lemma 5.2.2** Suppose \( G_p \) is a connection graph whose underlying graph \( G_p \) is a simple path. The rotation matrices between \( v_i \) and \( v_{i+1} \) are \( O_{v_i,v_{i+1}} \in O(d) \) for \( 1 \leq i \leq n-1 \). Let \( \mathbb{L} \) be the connection Laplacian of \( G_p \) and \( L \) be the discrete Laplacian of \( G_p \) respectively. Then
\[
\mathbb{L}^+(v_i, v_j) = \begin{cases} 
L^+(v_i, v_j) \prod_{k=i}^{j-1} O_{v_k v_{k+1}} & i \neq j, \\
L^+(v_i, v_j) I_{d \times d} & i = j.
\end{cases}
\]
Proof. To verify \( L^+(v_i, v_j) \) is the pseudoinverse of \( L \), we just need to verify that \( L^+(v_i, v_j) \) satisfies all of the four criteria [Pen55, GVL96].

To see (i) \( PLL^+L = L \), we consider two vertices \( v_i \) and \( v_j \) and without loss of generality we may assume that \( i \leq j \). Note that

\[
(LLL^+)_{vi,vj} = \sum_{k,l} L(v_i,v_k)L^+(v_k,v_l)L(v_l,v_j)
\]

\[
= \sum_{k,l} L(v_i,v_k)L^+(v_k,v_l)L(v_l,v_j)O_{v_kv_l}O_{v_tv_l}
\]

\[
= \sum_{k,l} L(v_i,v_k)L^+(v_k,v_l)L(v_l,v_j)O_{v_tv_j}
\]

where the last equality is by the definition of \( O_{v_tv_j} \). Since \( L^+ \) is the pseudoinverse of \( L \), we also have \( LL^+L = L \) which implies that

\[
L(v_l,v_j) = \sum_{k,l} L(v_i,v_k)L^+(v_k,v_l)L(v_l,v_j).
\]

Thus,

\[
(LLL^+)_{vi,vj} = L(v_i,v_j)O_{v_tv_j} = L_{vi,vj}
\]

and the verification of (i) is completed.

To see (ii) \( L^+LLL^+ = L^+ \), we also consider two vertices \( v_i \) and \( v_j \) and without loss of generality we may assume that \( i \leq j \) and note that

\[
(L^+LLL^+)_{vi,vj} = \sum_{k,l} L^+(v_i,v_k)L(v_k,v_l)L^+(v_l,v_j)O_{v_tv_j}.
\]

The rest of argument is quite similar to that of (i), we omit here.

To see (iii) \( (LLL^+)^* = (LLL^+) \), we also consider two fixed vertices \( v_i \) and \( v_j \). Note that

\[
(LLL^+)_{vi,vj} = \sum_k L(v_i,v_k)L^+(v_k,v_j)
\]

\[
= \sum_k L(v_i,v_k)L^+(v_k,v_j)O_{v_kv_l}O_{v_tv_l}
\]

\[
= \sum_k L(v_i,v_k)L^+(v_k,v_j)O_{v_tv_j}.
\]
On the other side,

\[(LL^+)(v_j, v_i) = \sum_k L_{v_j,v_k} L_{v_k,v_i}^+ O_{v_j,v_i} \]

\[= \sum_k L_{v_j,v_k} L_{v_k,v_i}^+ O_{v_j,v_i}^T.\]

Since \(L^+\) is the pseudoinverse of \(L\), we also have \((LL^+)^* = LL^+\) which implies that

\[\sum_k L(v_i, v_k) L^+(v_k, v_j) = \sum_k L(v_j, v_k) L^+(v_k, v_i)\]

and thus \((LL^+)^* = (LL^+)\).

The verification of \((iv) (L^+L)^* = L^+L\) is also similar to \((iii)\), we omit here.

For all above, the lemma follows.

The intuition behind Lemma 5.2.2 is by the observation that there exits only a unique path between any pair of vertices. In fact, with some mirror changes of the proof in Lemma 5.2.2, one can prove the following lemma.

**Lemma 5.2.3** Suppose \(G\) is a connection graph whose underlying graph \(G\) is a tree. The rotation matrices between \(u\) and \(v\) are \(O_{uv} \in O(d)\) for \((u,v) \in E\). Let \(L\) be the connection Laplacian of \(G\) and \(L\) be the Laplacian of \(G\) respectively. Then for two vertices \(u\) and \(v\) joined by a path, denoted by \((v_1 = u, v_1, \ldots, v_k = v)\), we have

\[L^+(u, v) = \begin{cases} 
L^+(u, v) \prod_{i=1}^{k-1} O_{v_i,v_{i+1}} & u \neq v, \\
L^+(u, v) I_{d \times d} & u = v.
\end{cases}\]

By using the above lemma, we examine the relation between the connection resistance and the effective resistance by the following theorem and give one of its application by Corollary 5.2.5.

**Theorem 5.2.4** Suppose \(G = (V, E, O, w)\) is a connection graph and \(T\) is a spanning tree of \(G\) satisfies

1. The edge weights of \(G\) satisfies \(w(uv) > 0\) if and only if \((u, v) \in T\),
2. The rotation matrices \(O_{uv} \in SO(d)\) for some odd number \(d > 0\).
Then for any edge \((u, v) \in G\) satisfying \(L^+(u, v) \leq 0\), we have

\[
R_{\text{eff}}(u, v) = R_{\text{eff}}(u, v).
\]

**Proof.** Let \(L_T\) be the connection Laplacian of \(T\) and \(L_G\) be the connection Laplacian of \(G\) respectively. Also, let \(T\) be the underlying tree of \(T\), \(L_T\) be the Laplacian of \(T\), and \(L_G\) be the Laplacian of \(G\) respectively. Clearly, \(L \triangleq L_G = L_T\). Let us fix an edge \(e = (u, v) \in G\). By the definition of effective resistance, \(R_{\text{eff}}(u, v)\) is the maximum eigenvalue of the following matrix

\[
\Pi(e, e) = \begin{bmatrix}
O_{uv} & -I_{d \times d}
\end{bmatrix}
\begin{bmatrix}
L^+(u, u) & L^+(u, v) \\
L^+(v, u) & L^+(v, v)
\end{bmatrix}
\begin{bmatrix}
O^T_{uv} \\
-I_{d \times d}
\end{bmatrix}
\]

where \(O_{uv}\) is the rotation from \(u\) to \(v\). By Lemma 5.2.3, we have

\[
L^+(u, u) = L^+(u, u)I_{d \times d},
\]
\[
L^+(u, v) = L^+(u, v)O_{puv},
\]
\[
L^+(v, v) = L^+(v, v)I_{d \times d},
\]
\[
L^+(v, u) = L^+(v, u)O_{puv} = L^+(u, v)O_{puv},
\]

where \(p_{uv} = (v_1 = u, v_1, \ldots, v_k = v)\) is the unique path from \(u\) to \(v\) in \(T\) and

\[
O_{puv} = \prod_{(v_i, v_{i+1}) \in p_{uv}} O_{v_i, v_{i+1}}.
\]

Thus, by the definition of matrix \(\Pi\),

\[
\Pi(e, e) = (L^+_{u,u} + L^+_{v,v}) I_{d \times d} - L^+_{u,v} (O_{puv}O_{vu} + O_{puv}O_{uv}).
\]

Note that \(O_{puv}O_{vu} \in \text{SO}(d)\) for some odd number \(d > 0\),

\[
\|O_{puv}O_{vu} + O_{puv}O_{uv}\|_2 = 2.
\]

By the assumption that \(L^+(u, v) \leq 0\), we obtain

\[
\|\Pi(e, e)\|_2 = L^+(u, u) + L^+(v, v) - 2L^+(u, v) = R_{\text{eff}}(u, v).
\]

Thus, the theorem is proved. \(\blacksquare\)
**Corollary 5.2.5** For any uniform weighted path on vertices \( v_1, v_2, \ldots, v_n \) with rotation matrices \( O_{v_i,v_{i+1}} \in SO(d) \) for \( 1 \leq i < n \) and some odd number \( d > 0 \),

\[
\mathbb{R}_{\text{eff}}(v_1, v_n) = R_{\text{eff}}(v_1, v_n).
\]

**Proof.** Without loss of generality, we may assume that all edge weights are one. Let \( \mathbb{L} \) be the connection Laplacian of the path and \( L \) be the discrete Laplacian of underlying path. By Theorem 5.2.4, it is sufficient to verify that

\[
2L^+(v_1, v_n) = L^+(v_1, v_1) + L^+(v_n, v_n) - R_{\text{eff}}(v_1, v_n)
\]

\[
\leq 0.
\]

One of noteworthy representations for the diagonal entry of \( L^+(v_i, v_i) \) is the following equation [HPG08] which holds for an arbitrary graph rather than a simple path

\[
L^+(v_i, v_i) = \frac{R(i)}{n} - \frac{R_{\text{tot}}}{n^2}
\]

where \( R(i) = \sum_{j=1}^{n} R_{\text{eff}}(v_i, v_j) \) and \( R_{\text{tot}} = \sum_{i<j} R_{\text{eff}}(v_i, v_j) \). As all the edge weights are just one, one may verify that for any \( 1 \leq i, j \leq n \) we have \( R_{\text{eff}}(v_i, v_j) = |i - j| \).

Thus,

\[
L^+(v_1, v_1) = \frac{R(1)}{n} - \frac{R_{\text{tot}}}{n^2}
\]

\[
= \frac{\sum_{j=1}^{n} (j - 1)}{n} - \frac{\sum_{i<j} (j - i)}{n^2}
\]

\[
= \frac{(2n - 1)(n - 1)}{6n}
\]

\[
= L^+(v_n, v_n).
\]

Since \( R_{\text{eff}}(v_1, v_n) = n - 1 \), we have the corollary follows by the fact that

\[
2L^+(v_1, v_n) = L^+(v_1, v_1) + L^+(v_n, v_n) - R_{\text{eff}}(v_1, v_n)
\]

\[
= \frac{(2n - 1)(n - 1)}{6n} + \frac{(2n - 1)(n - 1)}{6n} - (n - 1)
\]

\[
= \frac{-n^2 - 6n + 1}{6n}
\]

\[
\leq 0.
\]
5.3 Acknowledgement

Material in this chapter has appeared in the following article, accepted for publication:

Chapter 6

Ranking and Sparsifying Edges of a Connection Graph

In this chapter, we examine a basic problem on connection graph sparsification, to illustrate the usage of connection resistances in edge ranking. Graph sparsification was first introduced by Benczúr and Karger [BK96, Kar94a, Kar94b, Kar00] for approximately solving various network design problems. The heart of the graph sparsification algorithms is the sampling techniques for randomly selecting edges. The goal is to approximate a given graph $G$ with $m$ edges on $n$ vertices by a sparse graph $\tilde{G}$, called sparsifier, with $O(n \log(n))$ edges (or fewer) on the same set of vertices in such a way that every cut in sparsifier $\tilde{G}$ is within a factor of $1 \pm \epsilon$ of the corresponding cut in $G$ for some constant $\epsilon \in (0, 1]$.

For a connection graph, in which each edge of the graph is associated with a weight and also a rotation (which is a linear orthogonal transformation acting on a $d$-dimensional vector space for some positive integer $d$). The adjacency matrix and the discrete Laplace operator $\mathbb{L}$ are acting on the space of vector-valued functions (instead of the usual real-valued functions) and therefore can be represented by matrices of size $dn \times dn$ where $n$ is the number of vertices in the graph. The goal of sparsification is to approximate a given connection graph $\mathbb{G}$ with $m$ edges on $n$ vertices by a sparse connection graph $\tilde{\mathbb{G}}$ with $O(n \log(n))$ edges (or fewer) on the
same set of vertices such that

\[ \forall f : V \to \mathbb{R}^d, (1 - \epsilon) f^T L_G f \leq f^T L_{\tilde{G}} f \leq (1 + \epsilon) f^T L_G f \]

where \( L_G \) and \( L_{\tilde{G}} \) are the discrete connection Laplacian of connection graphs \( G \) and \( \tilde{G} \) respectively.

In Section 6.1, we will use connection resistances to sample edges of \( G \) in order to form the sparsifier \( \tilde{G} \) with \( O(n \log(n)) \) edges. In Section 6.2, we will discuss some variants of connection resistances.

6.1 Ranking Edges Using the Connection Resistance

A central part of a graph sparsification algorithm is the sampling technique for selecting edges. It is crucial to choose the appropriate probabilistic distribution which can lead to a sparsifier preserving every cut in the original graph or its spectrum. The following algorithm Sample is a generic sampling algorithm for a usual graph sparsification problem. We will show that Sample can also be applied to a connection graph where we will use the distribution proportional to the weighted connection resistances.

\[(\tilde{G} = (V, \tilde{E}, O, \tilde{w})) = \text{Sample}(G = (V, E, O, w), p', q)\]

1. For every edge \( e \in E \), set \( p_e \) proportional to \( p'_e \).

2. Choose a random edge \( e \) of \( G \) with probability \( p_e \), and add \( e \) to \( \tilde{G} \) with edge weight \( \tilde{w}_e = \frac{w_e}{q p_e} \). Take \( q \) samples independently with replacement, summing weights if an edge is chosen more than once.

3. Return \( \tilde{G} \).

Our main result is the following theorem.
Theorem 6.1.1 For a given connection graph $G$ and some positive $\xi > 0$, we consider $\tilde{G} = \text{Sample}(G, p', q)$, where $p'_e = w_e \mathcal{R}_{\text{eff}}(e)$ and $q = \frac{4nd \log(nd) \log(1/\xi)}{\epsilon^2}$. Suppose $G$ and $\tilde{G}$ have connection Laplacians $L_G$ and $L_{\tilde{G}}$ respectively. Then with probability at least $\xi$, for any function $\forall f : V \to \mathbb{R}^d$, we have

$$(1 - \epsilon)f^T L_G f \leq f^T L_{\tilde{G}} f \leq (1 + \epsilon)f^T L_G f.$$ (6.1)

We will apply the similar proof framework as in Section 3.2 as well as that [SS08] to prove Theorem 6.1.1. Before proving Theorem 6.1.1, we need the following two lemmas, in particular concerning the matrix $\Lambda$ defined by

$$\Lambda = W^{1/2}B_L^+B^T W^{1/2} \in \mathcal{F}(m, m, d; \mathbb{R}).$$

Lemma 6.1.2 The matrix $\Lambda$ has the following properties.

(i) $\Lambda$ is a projection matrix, i.e. $\Lambda^2 = \Lambda$.

(ii) The eigenvalues of $\Lambda$ are 1 with multiplicity at most $nd$ and 0 otherwise.

(iii) $\Lambda(e, e) = \Lambda(\cdot, e)^T \Lambda(\cdot, e)$.

Proof. To see (i), observe that

$$\Lambda^2 = (W^{1/2}B_L^+B^T W^{1/2})(W^{1/2}B_L^+B^T W^{1/2})$$

$$= W^{1/2}B_L^+L_G L_G^+B^T W^{1/2}$$

$$= W^{1/2}B_L^+B^T W^{1/2}$$

$$= \Lambda.$$

For (ii), we first note that $\dim(\text{im}(\Lambda)) = \dim(\text{im}(W^{1/2}B)) \leq nd$. However, as $\Lambda^2 = \Lambda$, the eigenvalues of $\Lambda$ are all 1 or 0. Thus, the eigenvalues of $\Lambda$ are 1 with multiplicity at most $nd$ and 0 otherwise.

For (iii), it is simply by the facts that $\Lambda = \Lambda^2$ and $\Lambda$ is symmetric. Thus, the lemma follows.

To show that the output of the algorithm $\text{Sample} \tilde{G} = (V, \tilde{E}, O, \tilde{w})$ is a good sparsifier for $G$, we need to show that the quadratic forms $f^T L_{\tilde{G}} f$ and $f^T L_G f$
are close. Similar to the techniques of [SS08], we reduce the problem of preserving \( f^T \mathbb{L}_G f \) to that of preserving \( g^T \Lambda g \). Let us consider a diagonal matrix \( S \in \mathcal{F}(m, m, d; \mathbb{R}) \). The diagonal blocks of \( S \) are scalar matrices given by

\[
S(e, e) = \frac{\tilde{w}_e}{w_e} I_{d \times d} = \frac{N_e}{qP_e} I_{d \times d}
\]

where \( N_e \) is the number of times \( e \) being sampled. The following lemma shows how to reduce the problem of preserving \( f^T \mathbb{L}_G f \) to that of preserving \( g^T \Lambda g \) by using matrix \( \Lambda \).

**Lemma 6.1.3** Suppose \( S \) is a nonnegative diagonal matrix and \( \epsilon < 1 \) is a positive constant such that

\[
\| \Lambda S \Lambda - \Lambda \Lambda \|_2 \leq \epsilon.
\]

Then we have

\[
\forall f : V \to \mathbb{R}^d, (1 - \epsilon) f^T \mathbb{L}_G f \leq f^T \tilde{\mathbb{L}}_G f \leq (1 + \epsilon) f^T \mathbb{L}_G f.
\]

**Proof.** The assumption is equivalent to

\[
\sup_{f \in \mathbb{R}^{nd}, f \neq 0} \frac{|f^T \Lambda (S - I) \Lambda f|}{f^T f} \leq \epsilon
\]

Restricting our attention to vectors in \( \text{im} \left( \mathbb{W}^{1/2} \mathbb{B} \right) \),

\[
\sup_{f \in \text{im} \left( \mathbb{W}^{1/2} \mathbb{B} \right), f \neq 0} \frac{|f^T \Lambda (S - I) \Lambda f|}{f^T f} \leq \epsilon
\]

Since \( \Lambda \) is the identity on \( \text{im} \left( \mathbb{W}^{1/2} \mathbb{B} \right) \) so \( \Lambda f = f \) for all \( f \in \text{im} \left( \mathbb{W}^{1/2} \mathbb{B} \right) \). Also, every such \( f \) can be written as \( f = \mathbb{W}^{1/2} \mathbb{B} g \) for \( g \in \mathbb{R}^{nd} \). Thus,

\[
\sup_{f \in \text{im} \left( \mathbb{W}^{1/2} \mathbb{B} \right), f \neq 0} \frac{|f^T \Lambda (S - I) \Lambda f|}{f^T f} = \sup_{f \in \text{im} \left( \mathbb{W}^{1/2} \mathbb{B} \right), f \neq 0} \frac{|f^T (S - I) f|}{f^T f} = \sup_{g \in \mathbb{R}^{nd}, \mathbb{W}^{1/2} \mathbb{B} g \neq 0} \frac{|g^T \mathbb{B}^T \mathbb{W}^{1/2} \mathbb{S} \mathbb{W}^{1/2} \mathbb{B} g - g^T \mathbb{B}^T \mathbb{W} \mathbb{B} g|}{g^T \mathbb{B}^T \mathbb{W} \mathbb{B} g} = \sup_{g \in \mathbb{R}^{nd}, \mathbb{W}^{1/2} \mathbb{B} g \neq 0} \frac{|g^T \mathbb{L}_G g - g^T \tilde{\mathbb{L}}_G g|}{g^T \tilde{\mathbb{L}}_G g} \leq \epsilon
\]

Rearranging yields the desired conclusion for all \( g \in \mathbb{R}^{nd} \). \( \blacksquare \)
We also require the following concentration inequality in order to prove our Theorem 6.1.1. Previously, various matrix concentration inequalities have been derived by many authors including Achiloptas [Ach01], Cristofies-Markström [CM08], Recht [Recss], and Tropp [Tro11]. Here we just need a simple version that is particularly suitable for our usage from [Ver08].

\textbf{Theorem 6.1.4 ([Ver08])} \textit{Let }$X_1, X_2, \ldots, X_q$\textit{ be independent symmetric random }$k \times k$\textit{ matrices with zero means, }$S_q = \sum_i X_i$\textit{, }$\|X_i\|_2 \leq 1$\textit{ for all }$i$\textit{ a.s. Then for every }$t > 0$\textit{ we have}

$$\Pr \left[ \|S_q\|_2 > t \right] \leq k \max \left( \exp \left( -\frac{t^2}{4\sum_i \|\text{Var}(X_i)\|_2} \right), \exp \left( -\frac{t}{2} \right) \right).$$

A direct consequence of Theorem 6.1.4 is the following corollary.

\textbf{Corollary 6.1.5} \textit{Suppose }$X_1, X_2, \ldots, X_q$\textit{ are independent random symmetric }$k \times k$\textit{ matrices satisfying}

1. \textit{for all }$1 \leq i \leq q$, \textit{ }$\|X_i\|_2 \leq M$ a.s.,

2. \textit{for all }$1 \leq i \leq q$, \textit{ }$\|\text{Var}(X_i)\|_2 \leq M \|\mathbb{E}[X_i]\|_2$.

\textit{Then for any }$\epsilon \in (0, 1)$\textit{ we have}

$$\Pr \left[ \left\| \sum_i X_i - \sum_i \mathbb{E}[X_i] \right\|_2 > \epsilon \sum_i \|\mathbb{E}[X_i]\|_2 \right] \leq k \exp \left( -\frac{\epsilon^2 \sum_i \|\mathbb{E}[X_i]\|_2}{4M} \right).$$

\textbf{Proof.} \textit{Let us consider the following independent random symmetric matrices}

$$\frac{X_i - \mathbb{E}[X_i]}{M}$$

\textit{for }$1 \leq i \leq q$. \textit{Clearly they are independent symmetric random }$k \times k$\textit{ matrices with zero means satisfying}

$$\left\| \frac{X_i - \mathbb{E}[X_i]}{M} \right\|_2 \leq 1$$

\textit{for }$1 \leq i \leq q$. \textit{Also we note that}

$$\text{Var} \left( \frac{X_i - \mathbb{E}[X_i]}{M} \right) = \text{Var} \left( \frac{X_i}{M} \right) = \frac{\text{Var}(X_i)}{M^2}.$$
Thus, by applying the Theorem 6.1.4 we have
\[
\Pr \left[ \left\| \sum_i X_i - \mathbb{E} [X_i] \right\| > t \right] = \Pr \left[ \left\| \sum_i X_i - \sum_i \mathbb{E} [X_i] \right\| > tM \right] \leq k \max \left( \exp \left( -\frac{t^2 M^2}{4 \sum_i \| \text{Var} (X_i) \|_2} \right), \exp \left( -\frac{t}{2} \right) \right). \tag{6.2}
\]

Note that by condition (2) we obtain
\[
\sum_i \| \text{Var} (X_i) \|_2 \leq M \sum_i \| \mathbb{E} [X_i] \|_2.
\]

Thus if we set
\[
t = \frac{\epsilon \sum_i \| \mathbb{E} [X_i] \|_2}{M},
\]
the left term in the right hand side of Equation (6.2) can be bounded as follows.
\[
\frac{t^2 M^2}{4 \sum_i \| \text{Var} (X_i) \|_2} \geq \frac{(\epsilon \sum_i \| \mathbb{E} [X_i] \|_2)^2}{4 M \sum_i \| \mathbb{E} [X_i] \|_2} = \frac{\epsilon^2}{4 M} \sum_i \| \mathbb{E} [X_i] \|_2.
\]

Thus, the corollary follows. \qed

**Proof of Theorem 6.1.1.** Our algorithm samples edges from \( G \) independently with replacements, with probabilities \( p_e \) proportional to \( w_e(e) \). Note that Sampling \( q \) edges from \( G \) corresponds to sampling \( q \) columns from \( \Lambda \). So we can write
\[
\Lambda \Sigma \Lambda = \sum_e \Lambda (\cdot, e) S(e, e) \Lambda (\cdot, e)^T = \sum_e \frac{N_e}{q p_e} \Lambda (\cdot, e) \Lambda (\cdot, e)^T = \frac{1}{q} \sum_{i=1}^q y_i y_i^T
\]
for block matrices \( y_1, \ldots, y_q \in \mathbb{R}^{nd \times d} \) drawn independently with replacements from the distribution
\[
y = \frac{1}{\sqrt{p_e}} \Lambda (\cdot, e) \text{ with probability } p_e.
\]

In order to apply Corollary 6.1.5, we need to evaluate the expectation of and the variance of matrix \( yy^T \).
Claim 6.1.6 \( yy^T \) has the following properties.

1. \( E[yy^T] = \Lambda \).
2. \( \|E[yy^T]\|_2 = 1 \).
3. \( \|\text{Var}(yy^T)\|_2 \leq 2nd \|E[yy^T]\|_2 \).
4. \( \|y_iy_i^T\|_2 \leq nd \).

**Proof.** For property (1), we note that by the definition of \( yy^T \),

\[
E[yy^T] = \sum_e p_e \frac{1}{p_e} \Lambda(\cdot,e)\Lambda(\cdot,e)^T = \Lambda.
\]

This implies that

\[
\|E[yy^T]\|_2 = \|\Lambda\|_2 = 1
\]

and thus property (2) follows.

For property (3), we first observe that

\[
\|\text{Var}(yy^T)\|_2 = \|E[yy^Tyy^T] - (E[yy^T])^2\|_2 \\
\leq \|E[yy^Tyy^T]\|_2 + \|(E[yy^T])^2\|_2.
\]

Since the second term of the right hand of above inequality can be bounded by

\[
\|(E[yy^T])^2\|_2 = \|\Lambda^2\|_2 \quad \text{(as property (1))} \\
= \|\Lambda\|_2 \\
= 1,
\]

it is sufficient to bound the term \( \|E[yy^Tyy^T]\|_2 \). By the definition of expectation, we observe that

\[
\|E[yy^Tyy^T]\|_2 = \left\| \sum_e p_e \frac{1}{p_e} \Lambda(\cdot,e)\Lambda(\cdot,e)^T \Lambda(\cdot,e)\Lambda(\cdot,e)^T \right\|_2 \\
= \left\| \sum_e \frac{1}{p_e} \Lambda(\cdot,e)\Lambda(e,e)\Lambda(\cdot,e)^T \right\|_2.
\]
This implies that

\[
\left\| \mathbb{E} \left[ yy^T yy^T \right] \right\|_2 = \max_{f \in \text{im}(\mathcal{W}_1/2\mathbb{B})} \sum_e \frac{1}{p_e} \frac{f^T \Lambda(\cdot, e) \Lambda(\cdot, e)^T f}{f^T f} 
\]

Recall that the probability \( p_e \) is proportional to \( w_e \mathcal{R}_{\text{eff}}(e) \), i.e.

\[
p_e = \frac{w_e \mathcal{R}_{\text{eff}}(e)}{\sum_e w_e \mathcal{R}_{\text{eff}}(e)} = \frac{\| \Lambda(e, e) \|_2}{\sum_e \| \Lambda(e, e) \|_2},
\]

we have

\[
\left\| \mathbb{E} \left[ yy^T yy^T \right] \right\|_2 \leq \sum_e \| \Lambda(e, e) \|_2 \left( \max_{f \in \text{im}(\mathcal{W}_1/2\mathbb{B})} \sum_e \frac{f^T \Lambda(\cdot, e) \Lambda(\cdot, e)^T f}{f^T f} \right) 
\]

\[
= \sum_e \| \Lambda(e, e) \|_2 \| \Lambda \|_2 
\]

\[
= \sum_e \| \Lambda(e, e) \|_2 
\]

\[
\leq \sum_e \text{Tr} (\Lambda(e, e)) 
\]

\[
= \text{Tr} (\Lambda) 
\]

\[
\leq nd.
\]

Thus,

\[
\left\| \text{Var} (yy^T) \right\|_2 \leq nd + 1 \leq 2nd \left\| \mathbb{E} \left[ yy^T \right] \right\|_2
\]

and the proof of property (3) is completed.

To see the property (4), we note that

\[
\| y_i y_i^T \|_2 \leq \max_e \left( \frac{\| \Lambda(\cdot, e)^T \Lambda(\cdot, e) \|_2}{p_e} \right) 
\]

\[
= \max_e \left( \frac{\| \Lambda(e, e) \|_2}{p_e} \right) 
\]

\[
= \max_e \left( \frac{w_e \mathcal{R}_{\text{eff}}(e)}{p_e} \right). \quad (6.3)
\]
Since the probability $p_e$ is proportional to $w_e \mathbb{R}_{\text{eff}}(e)$, i.e.,

$$p_e = \frac{w_e \mathbb{R}_{\text{eff}}(e)}{\sum_e w_e \mathbb{R}_{\text{eff}}(e)} = \frac{\| \Lambda(e, e) \|_2}{\sum_e \| \Lambda(e, e) \|_2},$$

we have

$$\| y_i^T y_i^T \|_2 \leq \sum_e \| \Lambda(e, e) \|_2 \leq \sum_e \text{Tr} (\Lambda(e, e)) = \text{Tr} (\Lambda) \leq nd. \quad (6.4)$$

For all above, we complete the proof of our claim. \[\square\]

To complete the proof by applying Corollary 6.1.5, by setting

$$q = \frac{4nd \log(nd) \log(1/\xi)}{\epsilon^2}$$

and the fact that dimension of $yy^T$ is $md$, we have

$$\Pr \left[ \left\| \frac{1}{q} \sum_{i=1}^q y_i y_i^T - \mathbb{E} [yy^T] \right\|_2 > \epsilon \right] \leq md \exp \left( -\frac{\epsilon^2 \sum_{i=1}^q \| \mathbb{E} [y_i y_i^T] \|_2}{4nd} \right) \leq md \exp \left( -\frac{\epsilon^2 q}{4nd} \right) \leq \xi$$

for some constant $0 < \xi < 1$. Thus, the theorem follows. \[\square\]

The reader may generalize the the Oversampling Theorem in [KMP10] and obtain the following Theorem 6.1.7. Here, we will just give the sketch proof for it.

**Theorem 6.1.7 (Oversampling)** Suppose connection graphs $G = (V, E, O, w)$ and $\tilde{G} = \text{Sample}(G, p', q)$ have connection Laplacian $L_G$ and $L_{\tilde{G}}$ respectively where $p'_e \geq w_e \mathbb{R}_{\text{eff}}(e)$, $t = \sum_{e \in E} p'_e$ and $q = \frac{4t \log(t) \log(1/\xi)}{\epsilon^2}$. Then with probability at least $\xi$, for all $f : V \rightarrow \mathbb{R}^d$, we have

$$(1 - \epsilon)f^T L_G f \leq f^T L_{\tilde{G}} f \leq (1 + \epsilon)f^T L_G f.$$ 

**Proof.** In the proof of Theorem 6.1.1, the key is to bound the norm $\| y_i y_i^T \|_2$ by using inequalities (6.3) and (6.4). If $p'_e \geq w_e \mathbb{R}_{\text{eff}}(e)$, the norm $\| y_i y_i^T \|_2$ is bounded by $\sum_{e \in E} p'_e$. Thus, the theorem follows. \[\square\]
6.2 Variants of the Connection Resistance

Now let us consider a variation of the connection resistance denoted by

$$R_{\text{eff}}(e) = \text{Tr} (\Pi(e, e)).$$

Clearly, we have $R_{\text{eff}}(e)$ is an upper bound of $\overline{R}_{\text{eff}}(e)$ as

$$R_{\text{eff}}(e) = \text{Tr} (\Pi(e, e)) \geq \|\Pi(e, e)\|_2 = \overline{R}_{\text{eff}}(e)$$

and we note that

$$\sum_e w_e R_{\text{eff}}(e) = \sum_e \text{Tr} (\Lambda(e, e)) = \text{Tr} (\Lambda) \leq nd.$$

Using Theorem 6.1.7, we have the following corollary for using the variants of connection resistance to sparsify the connection graph.

**Corollary 6.2.1** For a given connection graph $G$ and some positive $\xi > 0$, we consider $\tilde{G} = \text{Sample}(G, p', q)$, where $p'_e = w_e \overline{R}_{\text{eff}}(e)$ and $q = \frac{4nd \log(nd) \log(1/\xi)}{\epsilon^2}$. Suppose $G$ and $\tilde{G} = \text{Sample}(G, p', q)$ have connection Laplacian $L_G$ and $L_{\tilde{G}}$, respectively. Then with probability at least $\xi$, for all $f : V \rightarrow \mathbb{R}^d$, we have

$$(1 - \epsilon) f^T L_G f \leq f^T L_{\tilde{G}} f \leq (1 + \epsilon) f^T L_G f.$$

Now, we will discuss some issues for computing the variants of connection resistance.

**Theorem 6.2.2** If a linear system $Lz = y$ can be solved in $T$ time where $z, y \in \mathbb{R}^{md}$, then there is an algorithm which computes $\overline{R}_{\text{eff}}'(e)$ for all $e \in E$ satisfying

$$(1 - \epsilon) \overline{R}_{\text{eff}}(e) \leq \overline{R}_{\text{eff}}'(e) \leq (1 + \epsilon) \overline{R}_{\text{eff}}(e)$$

in $\tilde{O} \left( \frac{T \log(nd)}{\epsilon^2} \right)$ time.

**Proof.** Recall that the connection resistance of $e = (u, v)$ is $R_{\text{eff}}(u, v) = \|\Pi(e, e)\|_2$ where $\Pi(e, e)$ can be written as

$$\Pi(e, e) = B(e, \cdot)L^+B(e, \cdot)^T$$

$$= B(e, \cdot)L^+LL^+B(e, \cdot)^T$$

$$= (B(e, \cdot)L^+B^T W^{1/2})(W^{1/2}B^T L^+B(e, \cdot)^T).$$
Let $\mu^v$ be $v$-th column of block matrix $W^{1/2}B^T L^+$, i.e.

$$
\mu^v = [W^{1/2}B^T L^+] (\cdot, v).
$$

Let $\mu^{v,u}$ be the block matrix

$$
\mu^{v,u} = \mu^v O_{vu}.
$$

Note that $\mu^v$ and $\mu^{v,u}$ are both of dimension $md \times d$. By the definitions of $\mu^v$, $\mu^{v,u}$ and $B(e, \cdot)$, we have

$$
\Pi(e, e) = (\mu^{v,u} - \mu^v)^T (\mu^{v,u} - \mu^v) = \|\mu^{v,u} - \mu^v\|_2^2.
$$

This implies that

$$
\mathbb{R}_{\text{eff}}(e) = \text{Tr} \left(\Pi(e, e)\right) = \sum_{j=1}^{d} \|\mu^{v,u}_j - \mu^v_j\|_2^2
$$

where $\mu^{v,u}_j, \mu^v_j \in \mathbb{R}^{md}$ for $1 \leq j \leq d$ are the $j$-th column of $\mu^v$ and $\mu^{v,u}$ respectively. Thus, $\mathbb{R}_{\text{eff}}(e)$ is just the sum of Euclidean distances between $md$-dimension vectors $\{\mu^{v,u}_j\}$ and $\{\mu^v_j\}$ and the total number of such vectors is at most $2md$.

To apply Lemma 3.3.1, we do the three following steps first.

<table>
<thead>
<tr>
<th>Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Let $Q$ be a random $\pm 1$ matrix of dimension $k \times md$ where $k = \frac{96 \log(md)}{\epsilon^2}$.</td>
</tr>
<tr>
<td>2. Compute $Y = QW^{1/2}B$ which takes $2md^2 \times k + md = \tilde{O} \left(\frac{md^3}{\epsilon^2}\right)$ time since block matrix $B$ has at most $2md^2$ nonzero entries and $W^{1/2}$ is a diagonal block matrix.</td>
</tr>
<tr>
<td>3. For $1 \leq i \leq k$, let $y_i$ denote the rows of $Y$ and $z_i$ denote the columns of $Z^T$ where $Z = QW^{1/2}BL^+$. Thus, $z_i = L^+ y_i$ can be solved by solving the linear equation $Lz_i = y_i$.</td>
</tr>
</tbody>
</table>

The key observation is that each column of $Z$ are the vectors of form $\{Q \mu^v_j\}$ and the vectors of form $\{Q \mu^{v,u}_j\}$ can be easily computed by multiplication of rotations $O_{vu}$ to the matrix $[Q \mu^v_1, Q \mu^v_2, \ldots, Q \mu^v_d]$. By Lemma 3.3.1,

$$
(1 - \epsilon) \|\mu^{v,u}_j - \mu^v_j\|_2^2 \leq \|Q \mu^{v,u}_j - Q \mu^v_j\|_2^2 \leq (1 + \epsilon) \|\mu^{v,u}_j - \mu^v_j\|_2^2
$$
and thus
\[(1 - \epsilon) \overline{R}_{\text{eff}}(e) \leq \overline{R}'_{\text{eff}}(e) \leq (1 + \epsilon) \overline{R}_{\text{eff}}(e)\]
where \(\overline{R}_{\text{eff}} = \sum_{j=1}^{d} \|Q_{\mu_j}^u - Q_{\mu_j}^v\|^2_2\).

However, it is not clear how to solve the linear system \(Lz = y\) efficiently. Here, we will give a slightly different sample technique for sampling edges from \(G\) where the sample probabilities can be computed efficiently. Without loss of generality, we may assume that all edge weights are ones in the later arguments. We remark that without too much work one can generalize so that it applies to weighted graphs.

\[
(\tilde{G} = (V, \tilde{E}, O, \tilde{w})) = \text{Sparse}(G = (V, E, O, w), \kappa)
\]

1. For every edge \(e = (u, v) \in E\), set
\[p_e = \min \left(1, \frac{\kappa}{\min(d(u), d(v))}\right).\]

2. For every edge \(e = (u, v) \in E\), with probability \(p_e\) add an edge of weight \(\frac{1}{p_e}\) between vertices \(u\) and \(v\) to \(\tilde{G}\).

3. Return \(\tilde{G}\).

Our proof of the following theorem applies similar techniques introduced by Vu [Vu05] and extended by Spielman and Teng [ST06].

**Theorem 6.2.3** Suppose connection graph \(G\) and \(\tilde{G} = \text{Sparse}(G, \kappa)\) have connection matrix \(A\) and \(\tilde{A}\) respectively. Then for all even integer \(k\),
\[
\Pr \left[ \left\| D^{-1/2} (\tilde{A} - A) D^{-1/2} \right\| \geq \frac{2k n^{1/k}}{\sqrt{\kappa}} \right] \leq \frac{1}{2k}.
\]

**Proof.** We define block matrix \(\Upsilon \in \mathcal{F}(n, n; \mathbb{R})\) by
\[
\Upsilon = D^{-1} (\tilde{A} - A),
\]
i.e. for each edge \((u, v)\), \(\Upsilon(u, v) = \delta_{uv}O_{uv}\) where

\[
\delta_{uv} = \begin{cases}
\frac{1}{d(u)} \left( \frac{1}{p_{uv}} - 1 \right) & \text{with probability } p_{uv}, \\
- \frac{1}{d(u)} & \text{with probability } 1 - p_{uv}.
\end{cases}
\]

Lemma 6.2.4 implies that, for even \(k\),

\[
\frac{n k^k}{\kappa^{k/2}} \geq \mathbb{E} \left[ \text{Tr} \left( \Upsilon^k \right) \right] \geq \mathbb{E} \left[ \lambda_{\max}(\Upsilon^k) \right]
\]

where \(\lambda_{\max}(M)\) for a matrix \(M\) is the largest eigenvalue of matrix \(M\). Applying Markov’s inequality, we obtain

\[
\Pr \left[ \text{Tr} \left( \Upsilon^k \right) \geq 2^k \frac{n k^k}{\kappa^{k/2}} \right] \leq \frac{1}{2^k}.
\]

Recalling that the eigenvalues of \(\Upsilon^k\) are the \(k\)-th powers of the eigenvalues of \(\Upsilon\), and taking the \(k\)-th roots, we conclude that

\[
\Pr \left[ \left\| \mathbb{D}^{-1/2}(\tilde{A} - A)\mathbb{D}^{-1/2} \right\|_2 \geq \frac{2 k n^{1/k}}{\sqrt{\kappa}} \right] \leq \frac{1}{2^k}.
\]

\[\square\]

**Lemma 6.2.4** For any even positive number \(k\),

\[
\mathbb{E} \left[ \text{Tr} \left( \Upsilon^k \right) \right] \leq \frac{n d k^k}{\kappa^{k/2}}.
\]

**Proof.** Note that

\[
\mathbb{E} \left[ \Upsilon^k(v_0, v_k) \right] = \sum_{v_1, \ldots, v_{k-1}} \mathbb{E} \left[ \prod_{i=1}^{k} \Upsilon(v_{i-1}, v_i) \right] = \sum_{v_1, \ldots, v_{k-1}} \mathbb{E} \left[ \prod_{i=1}^{k} \delta(v_{i-1}, v_i) \right] \prod_{i=1}^{k} O_{v_{i-1}v_i}
\]

where the summation above is over all the possible paths from \(v_0\) to \(v_k\) of length \(k\). Thus, by the commute property of expectation and trace and fact that the product of rotations \(\prod_{i=1}^{k} O_{v_{i-1}v_i}\) is also an orthogonal rotation, we have

\[
\text{Tr} \left( \prod_{i=1}^{k} O_{v_{i-1}v_i} \right) \leq d,
\]
and
\[
E \left[ \text{Tr} \left( \mathcal{Y}^k(v_0, v_k) \right) \right] = \text{Tr} \left( E \left[ \text{Tr}^k(v_0, v_k) \right] \right) \\
= \sum_{v_1, \ldots, v_{k-1}} E \left[ \prod_{i=1}^{k} \delta(v_{i-1}, v_i) \right] \text{Tr} \left( \prod_{i=1}^{k} O_{v_{i-1}v_i} \right) \\
\leq d \sum_{v_1, \ldots, v_{k-1}} E \left[ \prod_{i=1}^{k} \delta(v_{i-1}, v_i) \right]
\]

There are many ways to bound the summation over all possible sequences \(v_1, \ldots, v_{k-1}\) that could possibly contribute, e.g. by the Lemma 6.5 in [ST08], we have
\[
\sum_{v_1, \ldots, v_{k-1}} E \left[ \prod_{i=1}^{k} \delta(v_{i-1}, v_i) \right] \leq \frac{k^k}{\kappa^{k/2}}.
\]
Again, by the commute property of expectation
\[
E \left[ \text{Tr} \left( \mathcal{Y}^k \right) \right] = \sum_{v \in V} E \left[ \text{Tr} \left( \mathcal{Y}^k(v, v) \right) \right] \leq \frac{ndk^k}{\kappa^{k/2}}.
\]
Thus, the lemma follow.

6.3 Acknowledgement

Material in this chapter has appeared in the following article, accepted for publication:

Chapter 7

Local Consistency of a Connection Graph

Connection graphs arise in numerous applications, in particular for data and image processing involving high-dimensional data sets. To quantify the affinities between two data points, it is often not enough to use only a scalar edge weight. For example, if the high-dimensional data set can be represented or approximated by a low-dimensional manifold, the patterns associated with nearby data points are likely to related by certain rotations [StWss]. Besides being a general framework for data analysis and manifold learning, connection graphs are extremely useful tools for measuring alignments between objects such as one-dimensional periodic signals, two-dimensional images and three-dimensional shapes, and then can serve as basic platforms for performing objects partitioning and clustering in the related research fields e.g. cryo-electron microscopy [HS11, Sin11, SZSH11], angular synchronization of eigenvectors [CLSss, Sin11, SZSH11] and vector diffusion maps [StWss].

However, the rotations of a connection graph are normally contaminated by “noise” which comes from the sampling procedure of high-dimensional data points. The presence of noise would significantly flaw the quality of clustering base on the corresponding connection graph. Usually the underlying assumption is that data points lie exactly on a manifold, without any noise contamination, i.e. the corresponding connection graph is consistent. This assumption needs
to be examined before the further use of corresponding connection graphs. In the meanwhile, many information networks arising from massive data sets exhibit the small world phenomenon. Consequently exploring the “local inconsistency” or “local consistency” of a small portion of a connection graph also attract lots interests.

Here we consider two basic notions, the connection PageRank (which is a generalization of the usual PageRank) and the (local) inconsistency of a connection graph. We try to establish relationship between them. The connection PageRank vector can be used to provide sweep cut which has been proven to be a useful technique in spectra partitioning and clustering related to PageRank vector in the usual graphs [ACL06, AC07, ACL07]. Besides, PageRank vector can also be used to measure the how consistent of a subgraph. We will define the inconsistent coefficient by using PageRank and then exam how locally consistent for sweep cuts produced by the PageRank vector.

### 7.1 PageRank Vectors in a Connection Graph

The PageRank vector is based on random walks. Here we consider a lazy walk on $G$ with the transition probability matrix

$$Z = \frac{I + P}{2}.$$  

In [ACL06], a PageRank vector $pr_{\alpha,s}$ is defined by a recurrence relation involving a seed vector $s$ (as a probability distribution) and a positive jumping constant $\alpha < 1$ (or transportation constant). Namely [ACL06],

$$pr_{\alpha,s} = \alpha s + (1 - \alpha)Zpr_{\alpha,s}$$

$$= \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t Z^t s$$

$$= \alpha \left( I + (1 - \alpha)Z \sum_{t=0}^{\infty} (1 - \alpha)^t Z^t \right) s$$

$$= \alpha \hat{s} + (1 - \alpha)pr_{\alpha,z}.$$  

For the connection graph $G$, the PageRank vector $\hat{pr}_{\alpha,\hat{s}}: V \to \mathbb{R}^d$ is defined by the same recurrence relation involving a seed vector $\hat{s}: V \to \mathbb{R}^d$ and a positive
jumping constant $\alpha < 1$:

$$\hat{pr}_{\alpha,\hat{s}} = \alpha \hat{s} + (1 - \alpha)Z\hat{pr}_{\alpha,\hat{s}}$$

where

$$Z = \frac{I_{nd \times nd} + P}{2}$$

is the transition probability matrix of a lazy random walk on $G$. Similarly as PageRank defined in [ACL06], an alternative definition of the PageRank vector is the following geometric sum of random walks:

$$\hat{pr}_{\alpha,\hat{s}} = \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t Z^t \hat{s}$$

$$= \alpha \left( I + (1 - \alpha)Z \sum_{t=0}^{\infty} (1 - \alpha)^t Z^t \right) \hat{s}$$

$$= \alpha \hat{s} + (1 - \alpha)\hat{pr}_{\alpha,Z\hat{s}}. \quad (7.1)$$

By Theorem 4.4.1 and Equation (7.1), we here state the following useful facts concerning PageRank vectors for a consistent connection graph.

**Proposition 7.1.1** Suppose $G$ is a consistent that a connection graph. Then for any $u \in V$, $\alpha \in (0, 1)$ and any function $\hat{s} : V \to \mathbb{R}^d$ satisfying $\|\hat{s}(u)\|_2 = 1$ and $\hat{s}(v) = 0$ for $v \neq u$, we have

$$\left\| \hat{pr}_{\alpha,\hat{s}}(v) \right\|_2 = pr_{\alpha,\chi_u}(v).$$

In particular,

$$\sum_{v \in V} \left\| \hat{pr}_{\alpha,\hat{s}}(v) \right\|_2 = \left\| pr_{\alpha,\chi_u} \right\|_1 = 1.$$

**Proof.** Since function $\hat{s}$ satisfies $\|\hat{s}(u)\|_2 = 1$ and $\hat{s}(v) = 0$ for $v \neq u$, by Theorem 4.4.1, for a fixed $v \in V$, $[Z^t\hat{s}](v)$ are all equal to each other for all $t > 0$. By the
geometric sum expression of PageRank vector, we have

\[ \| \text{pr}_{\alpha,\hat{s}}(v) \|_2 = \left\| \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t [Z_t^t \hat{s}] (v) \right\|_2 \]

\[ = \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t \| [Z_t^t \hat{s}] (v) \|_2 \]

\[ = \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t [Z_t^t \chi_u] (v) \]

\[ = \text{pr}_{\alpha,\chi_u}(v). \]

Thus,

\[ \sum_{v \in V} \| \hat{\text{pr}}_{\alpha,\hat{s}}(v) \|_2 = \| \text{pr}_{\alpha,\chi_u} \|_1 = 1. \]

We will call such a PageRank vector \( \hat{\text{pr}}_{\alpha,\hat{s}} \) a connection PageRank vector on \( u \). To compute a connection PageRank, we need the following subroutine called \text{PushV} and Lemma 7.1.2 for proving Theorem 7.1.3.

**PushV\((u, \alpha)\)**

Let \( \hat{p}' = \hat{p} \) and \( \hat{r}' = \hat{r} \), except for these changes:

1. Let \( \hat{p}'(u) = \hat{p}(u) + \alpha \hat{r}(u) \) and \( \hat{r}'(u) = \frac{1-\alpha}{2} \hat{r}(u) \).

2. For each vertex \( v \) such that \( (u, v) \in E \):

\[ \hat{r}'(v) = \hat{r}(v) + \frac{(1-\alpha)w(u,v)}{2d(u)} O_{uv} \hat{r}(u). \]

**Lemma 7.1.2** Let \( \hat{p}' \) and \( \hat{r}' \) denote the resulting vectors after performing operation \text{PushV}\((u)\) with \( \hat{p} \) and \( \hat{r} \). Then

\[ \hat{p}' + \hat{\text{pr}}_{\alpha,\hat{r}'} = \hat{p} + \hat{\text{pr}}_{\alpha,\hat{r}}. \]

**Proof.** After the push operation, we have

\[ \hat{p}' = \hat{p} + \alpha \hat{r}(u) \chi_u \]

\[ \hat{r}' = \hat{r} - \hat{r}(u) \chi_u + (1 - \alpha) Z \hat{r}(u) \chi_u. \]
By Equation (7.1),
\begin{align*}
\hat{p} + \hat{pr}_{\alpha, \widehat{r}} &= \hat{p} + \hat{pr}_{\alpha, \widehat{r}}(u)\chi_u + \hat{pr}_{\alpha, \widehat{r}}(u)\chi_u \\
&= \hat{p} + \hat{pr}_{\alpha, \widehat{r}}(u)\chi_u + [\alpha \hat{r}(u)\chi_u + (1 - \alpha)\hat{pr}_{\alpha, \widehat{r}}(u)\chi_u] \\
&= [\hat{p} + \alpha \hat{r}(u)\chi_u] + \hat{pr}_{\alpha, [\hat{r}(u)\chi_u + (1 - \alpha)\hat{r}(u)\chi_u]} \\
&= \hat{p}' + \hat{pr}_{\alpha, \widehat{r'}}.
\end{align*}

Thus, the lemma follows.  

\[\hat{p}, \hat{r} = \text{ApproximatePRV}(\hat{s}, \alpha, \epsilon)\]

1. Let \(\hat{p}(v) = 0\) and \(\hat{r}(v) = \hat{s}(v)\) for all \(v \in V\).
2. While \(\|\hat{r}(u)\|_2 \geq \epsilon d(u)\) for some vertex \(u\):
   - Pick any vertex \(u\) where \(\|\hat{r}(u)\|_2 \geq \epsilon d(u)\) and apply operation PushV\((u, \alpha)\).
3. Return \(\hat{p}\) and \(\hat{r}\).

\textbf{Theorem 7.1.3} For a vector \(s\) with \(\sum_{v \in V} \|\hat{s}(v)\|_2 \leq 1\), and a constant \(0 < \epsilon < 1\), the algorithm \text{ApproximatePRV}(\hat{s}, \alpha, \epsilon)\) computes an approximate PageRank vector \(\hat{p} = \hat{pr}_{\alpha, \widehat{r}}\) such that the residual vector \(\hat{r}\) satisfies
\[
\max_{v \in V} \frac{\|\hat{r}(v)\|_2}{d(v)} \leq \epsilon,
\]
and
\[
\sum_{v: \|\hat{p}(v)\|_2 > 0} d(v) \leq \frac{1}{\epsilon \alpha}.
\]
Furthermore, the running time of the algorithm is \(O\left(\frac{d^2}{\epsilon \alpha}\right)\).

\textbf{Proof.} Let \(T\) denote the total number of PushV operations performed by \text{ApproximatePRV}, and let \(d_i\) be the degree of the vertex \(u_i\) used in the \(i\)-th push. At time
\( \| \hat{r}(u) \|_2 \geq \epsilon d_i \), the quantity \( \sum_v \| \hat{r}(v) \|_2 \) during the \( i \)-th push decreases by the following amount:

\[
\| \hat{r}(u_i) \|_2 - \frac{1 - \alpha}{2} \| \hat{r}(u_i) \|_2 + \sum_{(u_i, v) \in E} \| \hat{r}(v) \|_2 - \left( \| \hat{r}(v) + (1 - \alpha)w_{uv}O_{uv}\hat{r}(u_i) \|_2 \right) \\
\geq \frac{1 + \alpha}{2} \| \hat{r}(u_i) \|_2 - \sum_{(u_i, v) \in E} \frac{(1 - \alpha)w_{uv}}{2d_u} \| \hat{r}(u_i) \|_2 \\
= \alpha \| \hat{r}(u_i) \|_2 \\
\geq \alpha \epsilon d_i.
\]

Since \( \sum_{v \in V} \| \hat{s}(v) \|_2 \leq 1 \) initially, we have \( \alpha \epsilon \sum_{i=1}^{T} d_i \leq 1 \) and thus

\[
\sum_{i=1}^{T} d_i \leq \frac{1}{\epsilon \alpha}. \tag{7.2}
\]

Lemma 7.1.2 implies that at every step of the algorithm returns an approximate PageRank vector \( \hat{p} = \hat{p}_{\alpha, \hat{s} - \hat{r}} \). The stopping condition implies that \( \hat{r} \) satisfies \( \frac{\| \hat{r}(v) \|_2}{d(v)} \leq \epsilon \), for all \( v \in V \). To bound the volume of the support of \( \hat{p} \), we note that for each vertex with \( \| \hat{r}(v) \|_2 > 0 \) we must have performed at least one push operation on that vertex. If \( d_i \) is the degree of the vertex pushed during step \( i \), then Equation (7.2) implies that

\[
\sum_{v: \| \hat{r}(v) \|_2 > 0} d(v) \leq \sum_{i=1}^{T} d_i \leq \frac{1}{\epsilon \alpha}.
\]

For a PushV operation at a vertex \( u \), the required time, including necessary queue updates, is proportional to \( d^2d_u \). Therefore the total running time is \( O \left( \frac{d^2}{\epsilon \alpha} \right) \) and the theorem follows.

**Theorem 7.1.4** For constants \( 0 < \epsilon, \alpha < 1 \), and a seed vector \( \hat{s} \) with \( \sum_v \| \hat{s}(v) \|_2 \leq 1 \), the algorithm SharpApproximatePRV(\( \hat{s}, \alpha, \epsilon \)) computes approximate PageRank vector \( \hat{p} = \hat{p}_{\alpha, \hat{s} - \hat{r}} \) such that the residual vector \( \hat{r} \) satisfies

\[
\max_v \frac{\| \hat{r}(v) \|_2}{d(v)} \leq \epsilon
\]

and the running time is \( O \left( \frac{md^2 \log(1/\epsilon)}{\alpha} \right) \).
\((\hat{p}, \hat{r}) = \text{SharpApproximatePRV}(\hat{s}, \alpha, \epsilon)\)

1. Let \(\xi = 1\), \(\hat{r} = \hat{s}\) and \(\hat{p} = \vec{0}\).

2. While \(\xi > \epsilon\):
   a. Set \(\xi = \frac{\xi}{2}\).
   b. Let \(\hat{p}'\) and \(\hat{r}'\) be the output of \(\text{ApproximatePRV}(\hat{r}, \alpha, \xi)\).
   c. Let \(\hat{p} = \hat{p} + \hat{p}'\) and \(\hat{r} = \hat{r}'\).

3. Return \(\hat{p}\) and \(\hat{r}\).

**Proof.** The algorithm returns an approximate PageRank vector \(\hat{p} = \hat{pr}_{\alpha, \hat{s} - \hat{r}}\) when the residual vector satisfies that \(\max_v \|\hat{r}(v)\|_2 \leq \epsilon\).

To bound the running time, we examine one fixed round of while-loop in the second step. Let \(T\) denote the total number of \text{PushV} operations performed by \text{ApproximatePRV} and let \(d_i\) denote the degree of the vertex involved in the \(i\)th \text{PushV} operation. When the \(i\)th \text{PushV} operation was performed, the quantity \(\sum_v \|\hat{r}(v)\|_2\) decreases at least by the amount \(\alpha \xi d_i\). Since at the beginning of each while-loop \(\sum_v \|\hat{r}(v)\|_2\) is at most \(2\xi \sum_{v \in V} d(v) = 2m \xi\), we have

\[
\xi \alpha \sum_{i=1}^{T} d_i \leq 2m \xi,
\]

which implies that \(\sum_{i=1}^{T} d_i \leq \frac{2m}{\alpha}\). Since there are at most \(\log \left( \frac{1}{\epsilon} \right)\) rounds in the second step, the total running time is bounded by \(O\left( \frac{md^2 \log(1/\epsilon)}{\alpha} \right)\).

### 7.2 The Inconsistency Coefficient

Recall that the transition matrix for the random walk on a connection graph \(G\) is \(P = AD^{-1}\) and transition matrix for the lazy random walk is

\[
Z = \frac{I_{nd \times nd} + P}{2}.
\]
Consider any characteristic function \( \widehat{\chi}_v : V \rightarrow \mathbb{R}^d \) satisfying

\[
\| \widehat{\chi}_v(u) \|_2 = \begin{cases} 
1, & \text{if } u = v, \\
0, & \text{if } u \neq v,
\end{cases}
\]

and a \( t \)-step random walk starting from vertex \( v \) with \( d \)-dimensional vector \( \widehat{\chi}_v(v) \) for any \( t > 0 \). Ideally, if \( G \) is consistent and by Theorem 4.4.1 we will have

\[
\sum_{u \in V} \| Z_t \widehat{\chi}_v(u) \|_2^2 = 1
\]

for all \( t > 0 \), i.e. the “probability” doesn’t vanish during the random walk process. However if \( G \) is inconsistent, we will normally obtain the following inequality

\[
\sum_{u \in V} \| Z_t \widehat{\chi}_v(u) \|_2^2 < 1
\]

for some \( t > 0 \), i.e. some of the “probability” canceled out each other due to the inconsistency of the rotations.

It would be interesting to consider the “local consistency” around a fixed vertex \( u \in V \) or a small size subset of vertices. Instead of considering a fixed \( t \), we consider the geometric mean over all possible \( t \)-step random walks, i.e.

\[
\hat{\text{pr}}_{\alpha, \widehat{\chi}_v} = \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t Z_t \widehat{\chi}_v.
\]

Here, the positive parameter \( \alpha < 1 \) can be used to measure the locality.

In order to quantify the average loss of “probability” at each step of random walk, we consider the following coefficient

\[
\omega_{\alpha, G} = \max_{u \in V} \sup_{\widehat{\chi}_u} \frac{\sum_{v \in V} \| Z \hat{\text{pr}}_{\alpha, \widehat{\chi}_u}(v) \|_2}{\sum_{v \in V} \| \hat{\text{pr}}_{\alpha, \widehat{\chi}_u}(v) \|_2}
\]

called the inconsistency coefficient of \( G \). The following are some useful facts for \( \omega_G \).

**Lemma 7.2.1** The inconsistency coefficient \( \omega_G \) of a connection graph \( G \) satisfies

(i) \( \omega_G \leq 1 \);

(ii) if \( G \) is consistent, then \( \omega_G = 1 \).
Proof. To see (i), for any function \( \hat{p} : V \to \mathbb{R}^d \) we can rewrite \( \left[ Z \hat{p} \right](v) \) as

\[
\left[ Z \hat{p} \right](v) = \left[ \left( \frac{I + \mathbb{P}}{2} \right) \hat{p} \right](v) = \frac{\hat{p}(v)}{2} + \frac{1}{2} \sum_{(u,v) \in E} O_{uv} \hat{p}(u) \frac{d(u)}{d(u)}.
\]

Thus, we have

\[
\sum_{v \in V} \left\| \left[ Z \hat{p} \right](v) \right\|_2 \leq \frac{1}{2} \sum_{v \in V} \left\| \hat{p}(v) \right\|_2 + \frac{1}{2} \sum_{v \in V} \sum_{(u,v) \in E} \left\| O_{uv} \hat{p}(u) \right\|_2 \frac{d(u)}{d(u)}
\]

\[
= \frac{1}{2} \sum_{v \in V} \left\| \hat{p}(v) \right\|_2 + \frac{1}{2} \sum_{v \in V} \sum_{(u,v) \in E} \left\| \hat{p}(u) \right\|_2 \frac{d(u)}{d(u)}
\]

\[
= \frac{1}{2} \sum_{v \in V} \left\| \hat{p}(v) \right\|_2 + \frac{1}{2} \sum_{v \in V} \left\| \hat{p}(v) \right\|_2
\]

\[
= \sum_{v \in V} \left\| \hat{p}(v) \right\|_2.
\]

This implies that \( \omega_G \leq 1 \).

For (ii), let \( \hat{p} = Z^t \hat{\chi}_v \) and note that \( \hat{p}(u) \) is the summation of all \( d \) dimensional vectors resulted from rotating \( \hat{\chi}_v(v) \) via rotations along all possible paths of length \( t \) from \( v \) to \( u \). Since \( G \) is consistent, the rotated vectors arrive at \( u \) via different paths are the vectors with the same direction. Also the rotations maintain the 2-norm of vectors. Thus, \( \left\| \hat{p}(u) \right\|_2 \) is simply the probability that a random walk in \( G \) arriving at \( u \) from \( v \) after \( t \) steps, i.e.

\[
\left\| \hat{p}(u) \right\|_2 = Z^t(v,u).
\]

Now, by the facts that

\[
\hat{p}_{\alpha, \hat{\chi}_v} = \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t Z^t \hat{\chi}_v,
\]

\[
Z \hat{p}_{\alpha, \hat{\chi}_v} = \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t Z^{t+1} \hat{\chi}_v
\]

we have

\[
\left\| \hat{p}_{\alpha, \hat{\chi}_v}(u) \right\|_2 = \left[ \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t Z^t \hat{\chi}_v \right](u) = \hat{p}_{\alpha, \hat{\chi}_v}(u),
\]

\[
\left\| Z \hat{p}_{\alpha, \hat{\chi}_v}(u) \right\|_2 = \left[ \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t Z^{t+1} \hat{\chi}_v \right](v) = \left[ Z \hat{p}_{\alpha, \hat{\chi}_v} \right](u).
\]
Thus the lemma follows by the fact that \( \sum_u \text{pr}_{\alpha,\chi_v}(u) = 1 \).

## 7.3 The Local Inconsistency of a Connection Graph

In the previous section, we defined the inconsistency coefficient for a connection graph \( G \) as \( \omega_{\alpha, G} \). Given a subset of vertices \( S \subseteq V \), we consider the subgraph \( G(S) \) induced by \( S \). Similarly, we can define by \( \omega_{\alpha, G(S)} \) the (local) inconsistency coefficient for the subgraph \( G(S) \), i.e.

\[
\omega_{\alpha, G(S)} = \max_{v \in V} \sup_{\hat{\chi}_u \subseteq S} \frac{\| \sum_{v \in S} \| Z|_S \hat{\text{pr}}_{\alpha, \hat{\chi}_u}(v) \|_2 \|}{\sum_{v \in S} \| \hat{\text{pr}}_{\alpha, \hat{\chi}_u}(v) \|_2}
\]

where block matrix \( Z|_S \in \mathcal{F}(n,n,d; \mathbb{R}) \) is the restriction of \( Z \) on \( S \). For any function \( \hat{p} : V \to \mathbb{R}^d \) any subset \( S \) of vertices, Let

\[
\hat{p}(S) = \sum_{v \in S} \| \hat{p}(v) \|_2
\]

and we have the following lemma.

**Lemma 7.3.1** For any positive constant \( \alpha < 1 \), any vertex \( v \in V \), any connection PageRank vector \( \hat{\text{pr}}_{\alpha, \hat{\chi}_v} : V \to \mathbb{R}^d \), and any subset \( S \) of vertices with inconsistency coefficient \( \omega_{\alpha, G(S)} \leq 1 - \xi \) for some \( \xi > 0 \), we have

\[
[Z \hat{\text{pr}}_{\alpha, \hat{\chi}_v}](S) \leq (1 - \xi) \hat{\text{pr}}_{\alpha, \hat{\chi}_v}(S) + \frac{1}{2} \hat{\text{pr}}_{\alpha, \hat{\chi}_v}(\bar{S}).
\]

**Proof.** Let \( \hat{p} = \hat{\text{pr}}_{\alpha, \hat{\chi}_v} \) and we note that if we take one step lazy random walk on the connection graph \( G \) we will have

\[
[Z \hat{p}](S)
= \sum_{v \in S} \left\| \frac{\hat{p}(v)}{2} + \frac{1}{2} \sum_{(u,v) \in E} \frac{O_{uv}\hat{p}(u)}{d(u)} \right\|_2
= \sum_{v \in S} \left\| \frac{\hat{p}(v)}{2} + \frac{1}{2} \sum_{(u,v) \in E, u \in S} \frac{O_{uv}\hat{p}(u)}{d(u)} \right\|_2 + \sum_{v \in S} \left\| \frac{1}{2} \sum_{(u,v) \notin S} \frac{O_{uv}\hat{p}(u)}{d(u)} \right\|_2.
\]
By the definition of the local inconsistency coefficient for the subgraph, we have

$$\sum_{v \in S} \left\| \frac{\hat{p}(v)}{2} + \frac{1}{2} \sum_{(u,v) \in E, u \in S} \frac{O_{uv}\hat{p}(u)}{d(u)} \right\|_2 \leq \omega_{(G(S))}\hat{p}(S).$$

Also, since the “probability” coming into $S$ from $\bar{S}$ is bounded by the total “probability” in $S$, we have

$$\sum_{v \in S} \left\| \frac{1}{2} \sum_{(u,v) \in E, u \notin S} \frac{O_{uv}\hat{p}(u)}{d(u)} \right\|_2 \leq \frac{1}{2}\hat{p}(\bar{S}).$$

For all the inequalities above, we can get

$$[Z\hat{p}] (S) \leq \omega_{G(S)}\hat{p}(S) + \frac{1}{2}\hat{p}(\bar{S}) \leq (1 - \xi)\hat{p}(S) + \frac{1}{2}\hat{p}(\bar{S})$$

and the lemma follows.

Consider a connection graph $G$ and its underlying graph $G$ together with a PageRank vector on $G$. Andersen, Chung and Lang in [ACL06] gave the following theorem for bounding the portion of a PageRank vector in the outside of a subset $C$ of vertices.

**Theorem 7.3.2 ([ACL06])** For any graph $G = (V, E, w)$ and any subset $C \subset V$ of vertices and any constant $\alpha \in (0, 1]$, there is a subset $C_\alpha \subseteq \text{with volume } \text{Vol}(C_\alpha) \geq \frac{\text{Vol}(C)}{2}$ such that for any vertex $v \in C_\alpha$, the PageRank vector satisfies

$$\text{pr}_{(G, \alpha)}(\bar{C}) \leq \frac{h_G(C)}{\alpha}.$$  

where $\chi_v(u) = 1$ if and only if $u = v$ otherwise $\chi_v(u) = 0$.

Similarly, we can prove Theorem 7.3.4 as Theorem 7.3.2 for PageRank vectors on a connection graph $G$. First of all, we need the following theorem to prove Theorem 7.3.4.
Theorem 7.3.3 Suppose $G$ is the underlying graph of a connection graph $G = (V, E, O, w)$. Then for any function $\hat{s} : V \rightarrow \mathbb{R}^d$ and $s : V \rightarrow \mathbb{R}$ satisfying $s(v) = \|\hat{s}(v)\|_2$, we have

$$\|\hat{\text{pr}}_{\alpha, \hat{s}}(v)\|_2 \leq \text{pr}_{\alpha, s}(v),$$

where $\text{pr}_{\alpha, s}$ is a PageRank vector on the graph $G$. In particular, if $G$ is consistent and $\hat{s}_u$ for some $u$, then

$$\|\hat{\text{pr}}_{\alpha, \hat{s}}(v)\|_2 = \text{pr}_{\alpha, s}(v).$$

Proof. By the definition of PageRank vector $\text{pr}_{\alpha, s}$ and the definition of connection PageRank vector $\hat{\text{pr}}_{\alpha, \hat{s}}$, we observe that

$$\hat{\text{pr}}_{\alpha, \hat{s}}(v) = \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t \left[Z^t \hat{s}\right](v),$$

$$\text{pr}_{\alpha, s}(v) = \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t \left[Z^t s\right](v).$$

Since

$$[Z^t \hat{s}](v) = \sum_{u \in V} [Z^t \hat{s}_u](v) \quad \text{and} \quad [Z^t s](v) = \sum_{u \in V} [Z^t s_u](v)$$

where $\hat{s}_u(u) = \hat{s}(u)$ and $\hat{s}_u(v) = 0$ for $v \neq u$, $s_u(u) = s(u)$ and $s_u(v) = 0$ for $v \neq u$. Note that $[Z^t \hat{s}_u](v)$ is simply a $d$-dimensional vector obtained from the summation over all the possible paths of lengths $t$ starting from $\hat{s}(u)$ at vertex $u$, through rotations and ending at vertex $v$. This implies that

$$\|\left[Z^t \hat{s}_u\right](v)\|_2 \leq \|Z^t s_u\|(v)$$

and thus

$$\|\hat{\text{pr}}_{\alpha, \hat{s}}(v)\|_2 \leq \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t \|Z^t s\|(v)$$

$$\leq \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t \sum_{u \in V} \|Z^t \hat{s}_u\|(v)$$

$$\leq \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t \sum_{u \in V} [Z^t s_u](v)$$

$$= \alpha \sum_{t=0}^{\infty} (1 - \alpha)^t [Z^t s](v)$$

$$= \text{pr}_{\alpha, s}(v).$$
The theorem follows.

Now, we are ready to state and prove the following Theorem 7.3.4.

**Theorem 7.3.4** For any connection graph \( G = (V, E, O, w) \) and any subset \( C \subset V \) of vertices and any constant \( \alpha \in (0, 1] \), there is a subset \( C_\alpha \subseteq \) with volume \( \text{Vol}(C_\alpha) \geq \frac{\text{Vol}(C)}{2} \) such that for any vertex \( v \in C_\alpha \), the PageRank vector on \( G \) satisfies

\[
\hat{\text{pr}}_{\alpha, \hat{\chi}_v}(\bar{C}) \leq \frac{h_G(C)}{\alpha}
\]

where \( \hat{\chi}_v : V \to \mathbb{R}^d \) is a function satisfying \( \|\hat{\chi}_v(u)\|_2 = 1 \) for \( u = v \) otherwise \( \|\hat{\chi}_v(u)\|_2 = 0 \) for \( u \neq v \).

**Proof.** Let graph \( G \) be the underlying graph of \( G \). Then by Theorem 7.3.2, there is a subset \( C_\alpha \subseteq \) with volume \( \text{Vol}(C_\alpha) \geq \frac{\text{Vol}(C)}{2} \) such that for any vertex \( v \in C_\alpha \), the PageRank vector satisfies

\[
\text{pr}_{\alpha, \chi_v}(\bar{C}) \leq \frac{h_G(C)}{\alpha}.
\]

Note that by Theorem 7.3.3,

\[
\hat{\text{pr}}_{\alpha, \hat{\chi}_v}(\bar{C}) = \sum_{u \in C} \|\hat{\text{pr}}_{\alpha, \hat{\chi}_v}(u)\|_2 \\
\leq \sum_{u \in C} \text{pr}_{\alpha, \chi_v}(u) \\
= \text{pr}_{\alpha, \chi_v}(\bar{C}) \\
\leq \frac{h_G(C)}{\alpha}.
\]

Thus, our theorem follows.

Now, we can combine our Lemma 7.3.1 and Theorem 7.3.4 to prove the following Theorem

**Theorem 7.3.5** For any connection graph \( G = (V, E, O, w) \), any positive constant \( \alpha < 1 \), and any subset \( C \subset V \) of vertices with local inconsistency coefficient

\[
\omega_{\alpha, G(C)} \leq 1 - \xi
\]
for some $\xi > 0$, there is a subset $C_\alpha \subseteq \mathbb{G}$ such that for any vertex $v \in C_\alpha$, the PageRank vector on $\mathbb{G}$ satisfies

$$\hat{\text{pr}}_{\alpha, \tilde{x}_v}(C) \leq \frac{\alpha + \frac{(1-\alpha)h_G(C)}{2\alpha}}{1 - (1 - \xi)(1 - \alpha)}.$$  

**Proof.** First of all, by the definition of PageRank vector, we have

$$\hat{\text{pr}}_{\alpha, \tilde{x}_v} = \alpha \tilde{x}_v + (1 - \alpha)Z\hat{\text{pr}}_{\alpha, \tilde{x}_v}.$$  

Also, by Lemma 7.3.1 and Theorem 7.3.4, we have

$$[Z\hat{\text{pr}}_{\alpha, \tilde{x}_v}] (C) \leq (1 - \xi)\hat{\text{pr}}_{\alpha, \tilde{x}_v}(C) + \frac{1}{2}\hat{\text{pr}}_{\alpha, \tilde{x}_v}(\tilde{C}) \leq (1 - \xi)\hat{\text{pr}}_{\alpha, \tilde{x}_v}(C) + \frac{h_G(C)}{2\alpha}.$$  

This implies that

$$\hat{\text{pr}}_{\alpha, \tilde{x}_v}(C) \leq \alpha \tilde{x}_v(C) + (1 - \alpha)Z\hat{\text{pr}}_{\alpha, \tilde{x}_v}(C) \leq \alpha + (1 - \xi)(1 - \alpha)\hat{\text{pr}}_{\alpha, \tilde{x}_v}(C) + \frac{(1 - \alpha)h_G(C)}{2\alpha}.$$  

By rearranging the terms, we get

$$\hat{\text{pr}}_{\alpha, \tilde{x}_v}(C) \leq \frac{\alpha + \frac{(1-\alpha)h_G(C)}{2\alpha}}{1 - (1 - \xi)(1 - \alpha)}.$$  

Thus, the theorem follows.  

### 7.4 Sweep Cuts in a Connection Graph

A sweep is a technique for producing a cut from a vector, and is widely used in spectral partitioning. We will use the following degree-normalized version of a sweep. Given a PageRank vector $\hat{p}$ with support size $N_{\hat{p}} = \|\text{Supp}(\hat{p})\|$, let $v_1, \ldots, v_{N_{\hat{p}}}$ be an ordering of the vertices from highest to lowest probability-per-degree, so that

$$\frac{\|\hat{p}(v_i)\|_2}{d(v_i)} \geq \frac{\|\hat{p}(v_{i+1})\|_2}{d(v_{i+1})},$$
This produces a collection of sets, with one set $S^\hat{p}_j = \{v_1, \ldots, v_j\}$, which we call sweep sets. We let $\Phi(\hat{p})$ be the smallest conductance of any of these sweep sets

$$h_G(\hat{p}) = \min_j h_G(S^\hat{p}_j).$$

To measure how a vector $\hat{p}$ is distributed in the graph, we define a function $\hat{p}[x]$ that gives an upper bound on the amount of probability on any set of vertices with volume $x$. We refer to this function as the Lovász-Simonovits curve, since it was introduced by Lovász and Simonovits [LS90, LS93]. This function is defined for all real numbers $x$ in the interval $[0, 2m]$, i.e.

$$\hat{p}\left[\text{Vol}\left(S^\hat{p}_j\right)\right] = \hat{p}\left(S^\hat{p}_j\right) = \sum_{v \in S^\hat{p}_j} \Vert \hat{p}(v) \Vert_2$$

for each $j \in [0, n]$, and define $\hat{p}[x]$ to be piecewise linear between these points. In other words, for any point $x \in [0, 2m]$, if $\hat{p}[x]$ is between $\text{Vol}\left(S^\hat{p}_j\right)$ and $\text{Vol}\left(S^\hat{p}_{j+1}\right)$, then

$$\hat{p}[x] = \hat{p}\left[\text{Vol}\left(S^\hat{p}_j\right)\right] + \frac{x - \hat{p}\left[\text{Vol}\left(S^\hat{p}_j\right)\right]}{d_{\hat{p}_{j+1}}} \hat{p}\left[\text{Vol}\left(S^\hat{p}_{j+1}\right)\right].$$

The function $\hat{p}[x]$ is increasing and concave. It is not hard to see that $\hat{p}[x]$ is an upper bound on the amount of probability from $\hat{p}$ on any set with volume $x$; for any set $S$, we have

$$\hat{p}(S) \leq \hat{p}\left[\text{Vol}\left(S\right)\right].$$

We view each undirected edge $(u, v)$ as a pair of directed edges $(u, v)$ and $(v, u)$. For each directed edge $(u, v)$ we let

$$\hat{p}(u, v) = \frac{\|\hat{p}(u)\|_2}{d(u)},$$

and for any set of directed edges $F$, we define

$$\hat{p}(F) = \sum_{(u, v) \in F} \hat{p}(u, v).$$

For any set $S$ of vertices, we define the set of directed edges into $S$,

$$\text{In}(S) = \{(u, v) \in E \mid v \in S\},$$
and the set of directed edges out of $S$,

$\text{Out}(S) = \{(u, v) \in E \mid u \in S\}$.

**Lemma 7.4.1** For any function $\hat{\rho} : V \to \mathbb{R}^d$, and any set $S$ of vertices,

$$[\hat{Z}\hat{\rho}](S) \leq \frac{1}{2} \hat{\rho}(\text{In}(S) \cup \text{Out}(S)) + \frac{1}{2} \hat{\rho}(\text{In}(S) \cap \text{Out}(S)).$$

**Proof.** Note that by the definition of the random walk on a connection graph $G$,

$$[\hat{Z}\hat{\rho}](u) = \frac{1}{2} \hat{\rho}(u) + \frac{1}{2} \sum_{(v, u) \in E} \frac{O_{vu} \hat{\rho}(v)}{d(v)}$$

$$= \frac{1}{2} \sum_{(u, v) \in E} \hat{\rho}(u) + \frac{1}{2} \sum_{(v, u) \in E} \frac{O_{vu} \hat{\rho}(v)}{d(v)}.$$

This implies that

$$[\hat{Z}\hat{\rho}](\{u\}) \leq \frac{1}{2} \sum_{(u, v) \in E} \frac{\|\hat{\rho}(u)\|_2}{d(u)} + \frac{1}{2} \sum_{(v, u) \in E} \frac{\|\hat{\rho}(v)O_{vu}\|_2}{d(v)}$$

$$= \frac{1}{2} \sum_{(u, v) \in E} \frac{\|\hat{\rho}(u)\|_2}{d(u)} + \frac{1}{2} \sum_{(v, u) \in E} \frac{\|\hat{\rho}(v)\|_2}{d(v)},$$

and

$$[\hat{Z}\hat{\rho}](S) = \sum_{u \in S} [\hat{Z}\hat{\rho}](\{u\})$$

$$\leq \frac{1}{2} \hat{\rho}(\text{In}(S)) + \frac{1}{2} \hat{\rho}(\text{Out}(S))$$

$$= \frac{1}{2} \hat{\rho}(\text{In}(S) \cup \text{Out}(S)) + \frac{1}{2} \hat{\rho}(\text{In}(S) \cap \text{Out}(S)).$$

Thus, the lemma follows.

**Lemma 7.4.2** If $\hat{\rho} = \hat{\rho}r_{\alpha, \tilde{s}}$ is a connection PageRank vector, then for any set of $S$ of vertices,

$$\hat{\rho}(S) \leq \alpha \tilde{s}(S) + \frac{1 - \alpha}{2} (\hat{\rho}(\text{In}(S) \cup \text{Out}(S)) + \hat{\rho}(\text{In}(S) \cap \text{Out}(S))).$$

Furthermore, for each $j \in [1, n - 1]$,

$$\hat{\rho}[\text{Vol}(S_j)] \leq \alpha \tilde{s}[\text{Vol}(S_j)] + \frac{1 - \alpha}{2} \left( \hat{\rho}[\text{Vol}(S_j)] + |\partial(S_j)| \right) + \hat{\rho} \left[ \text{Vol}(S_j) - |\partial(S_j)| \right].$$
Proof. Let $\hat{p} = \hat{p}_{\alpha, \hat{s}}$ and we have

$$\hat{p} = \alpha (\hat{s} - \bar{r}) + (1 - \alpha) \mathbb{E} \hat{p}$$

Applying Lemma 2, we obtain

$$\hat{p}(S) \leq \alpha \hat{s}(S) + \frac{1 - \alpha}{2} (\hat{p} (\text{In}(S) \cup \text{Out}(S)) + \hat{p} (\text{In}(S) \cap \text{Out}(S))).$$

This proves the first part of the lemma. To prove the second part, recall that

$$\hat{p} \left( \text{Vol} \left( S_{\hat{p}\alpha}^j \right) \right) = \hat{p} \left( S_{\hat{p}\alpha}^j \right)$$

for any $j \in [0, n]$. It can be verified that the curve $p[x]$ gives an upper bound on the amount of probability moving over any set of directed edges $F$, i.e.

$$\hat{p}(F) \leq p[|F|].$$

Thus,

$$\hat{p} \left[ \text{Vol} \left( S_{\hat{p}\alpha}^j \right) \right] = \hat{p} \left( S_{\hat{p}\alpha}^j \right) \leq \alpha \hat{s} \left( S_{\hat{p}\alpha}^j \right) + \frac{1 - \alpha}{2} (\hat{p} (\text{In}(S_{\hat{p}\alpha}^j) \cup \text{Out}(S_{\hat{p}\alpha}^j)) + \hat{p} (\text{In}(S_{\hat{p}\alpha}^j) \cap \text{Out}(S_{\hat{p}\alpha}^j))) + \frac{1 - \alpha}{2} \hat{p} \left[ \text{In} \left( S_{\hat{p}\alpha}^j \right) \cap \text{Out} \left( S_{\hat{p}\alpha}^j \right) \right] = \alpha \hat{s} \left[ \text{Vol} \left( S_{\hat{p}\alpha}^j \right) \right] + \frac{1 - \alpha}{2} \hat{p} \left[ \text{Vol} \left( S_{\hat{p}\alpha}^j \right) \right] \left[ \text{Vol} \left( S_{\hat{p}\alpha}^j \right) + \left| \partial \left( S_{\hat{p}\alpha}^j \right) \right| \right] + \frac{1 - \alpha}{2} \hat{p} \left[ \text{Vol} \left( S_{\hat{p}\alpha}^j \right) \right] \left[ \text{Vol} \left( S_{\hat{p}\alpha}^j \right) + \left| \partial \left( S_{\hat{p}\alpha}^j \right) \right| \right].$$

Thus, the lemma follows.

Lemma 7.4.3 For any starting vector $\hat{s}$, and any $x \in [0, 2m]$,

$$\hat{p}_{\alpha, \hat{s}}[x] \leq \hat{s}[x].$$
Proof. Let \( \hat{p} = \hat{p}_{r, \hat{\chi}} \), and by Lemma 7.4.2 we have
\[
\hat{p} \left[ \text{Vol} \left( S_{j}^{\hat{p}} \right) \right] \leq \alpha \hat{s} \left[ \text{Vol} \left( S_{j}^{\hat{p}} \right) \right] + \frac{1 - \alpha}{2} \left( \hat{p} \left[ \text{Vol} \left( S_{j}^{\hat{p}} \right) + |\partial \left( S_{j}^{\hat{p}} \right)| \right] + \hat{p} \left[ \text{Vol} \left( S_{j}^{\hat{p}} \right) - |\partial \left( S_{j}^{\hat{p}} \right)| \right] \right)
\leq \alpha \hat{s} \left[ \text{Vol} \left( S_{j}^{\hat{p}} \right) \right] + \frac{1 - \alpha}{2} \left( \hat{p} \left[ \text{Vol} \left( S_{j}^{\hat{p}} \right) \right] + \hat{p} \left[ \text{Vol} \left( S_{j}^{\hat{p}} \right) \right] \right)
\leq \alpha \hat{s} \left[ \text{Vol} \left( S_{j}^{\hat{p}} \right) \right] + (1 - \alpha) \hat{p} \left[ \text{Vol} \left( S_{j}^{\hat{p}} \right) \right]
\]
where the last line follows from the concavity of \( \hat{p}[x] \). This implies that
\[
\hat{p} \left[ \text{Vol} \left( S_{j}^{\hat{p}} \right) \right] \leq \hat{p} \left[ \text{Vol} \left( S_{j}^{\hat{p}} \right) \right].
\]
The lemma follows by the piecewise linearity of function \( \hat{p}[x] \) and \( \hat{s}[x] \).

7.5 Finding the Locally Consistent Area of a Vertex

In this section, we will propose an algorithm to find a local consistent area around a vertex. The sweep cuts technique together with PageRank vector has been well developed as a tool in spectral partitioning for finding local cuts with small Cheeger ratio and bounded volume [ACL06, AC07, ACL07]. Ideally if a connection graph \( G \) is consistent, by using theorem 7.3.3, we can show that the functionalities of the connection PageRank vector and the PageRank vector in the underlying graph are exactly the same in producing the sweep cuts. This implies that the local cuts produced by sweep cuts techniques via (connection) PageRank vectors are the same in terms of Cheeger ratio.

Given a vertex \( v \), we randomly pick a function \( \hat{\chi}_v : V \rightarrow \mathcal{R}^d \) and compute a connection PageRank vector \( \hat{p}_{r, \hat{\chi}_v} \) for some positive \( \alpha < 1 \). The vector \( \hat{p} = \hat{p}_{r, \hat{\chi}_v} \) will be further used to provide sweep cuts \( C_1, C_2, \ldots \). We will exam these cuts sequentially and try to identify the inconsistent ones. We summarize the details in algorithm \texttt{LocalConsistentSet} and the following theorem.

**Theorem 7.5.1** For any connection graph \( G = (V, E, O, w) \), any positive constant \( \alpha \in (0, 1] \), if there exists a subset \( C_\alpha \subseteq V \) with volume \( \text{Vol}(C_\alpha) \geq \frac{\text{Vol}(C)}{2} \) such that for
any vertex $v \in C_\alpha$, the PageRank vector on $\mathbb{G}$ satisfies

$$\hat{\text{pr}}_{\alpha,\hat{\chi}_v}(C) > \frac{\alpha + \frac{(1-\alpha)h_\mathbb{G}(C)}{2\alpha}}{1 - (1 - \xi)(1 - \alpha)}.$$  

Then, local inconsistency coefficient of $C$ satisfies that

$$\omega_{\alpha,\mathbb{G}(C)} > 1 - \xi.$$  

**Proof.** Proof by contradiction, we may assume that local inconsistency coefficient for the subset $C \subset V$ satisfies

$$\omega_{\alpha,\mathbb{G}(C)} \leq 1 - \xi$$

for some $\xi > 0$. However, by Theorem 7.3.5, we have

$$\hat{\text{pr}}_{\alpha,\hat{\chi}_v}(C) \leq \frac{\alpha + \frac{(1-\alpha)h_\mathbb{G}(C)}{2\alpha}}{1 - (1 - \xi)(1 - \alpha)}$$

which is a contradiction! Thus, the theorem follows. 

---

$(C) = \text{LocalConsistentSet}(\mathbb{G} = (V, E, O, w), v, \alpha, \xi)$

1. Set $\hat{\chi}_v(v)$ to be an arbitrary $d$-dimensional vector satisfying $\|\hat{\chi}_v(v)\|_2 = 1$ and $\|\hat{\chi}_v(u)\|_2 = 0$ for $u \neq v$.

2. Compute the connection PageRank vector $\hat{\text{pr}}_{\alpha,\hat{\chi}_v}$ with seed vector $\hat{\chi}_v$ and jumping constant $\alpha$.

3. Compute the sweep sets/cuts $C_1, C_2, \ldots$ by using PageRank vector $\hat{\text{pr}}_{\alpha,\hat{\chi}_v}$.

4. Set $i = 1$:

   4.a If $\hat{\text{pr}}_{\alpha,\hat{\chi}_v}(C) > \frac{\alpha + \frac{(1-\alpha)h_\mathbb{G}(C)}{2\alpha}}{1 - (1 - \xi)(1 - \alpha)}$, then $i = i + 1$.

   4.b Else output $C_i$. 

7.6 Acknowledgement

Material in this chapter has appeared in the following article, accepted for publication:

Bibliography


