Scalable Algorithms for Data and Network Analysis

Scalable Algorithms for Data and Network Analysis

Shang-Hua Teng
Computer Science and Mathematics
University of Southern California
shanghua.teng@gmail.com

now
the essence of knowledge
Boston — Delft

Foundations and Trends® in Theoretical Computer Science

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

S.-H. Teng. *Scalable Algorithms for Data and Network Analysis*. Foundations and Trends® in Theoretical Computer Science, vol. 12, no. 1-2, pp. 1–274, 2016.

This Foundations and Trends® issue was typeset in LaTeX using a class file designed by Neal Parikh. Printed on acid-free paper.

ISBN: 978-1-68083-130-6

Foundations and Trends® in Theoretical Computer Science
Volume 12, Issue 1-2, 2016

Editorial Board

Editorial Scope

Topics

Foundations and Trends® in Theoretical Computer Science publishes surveys and tutorials on the foundations of computer science. The scope of the series is broad. Articles in this series focus on mathematical approaches to topics revolving around the theme of efficiency in computing. The list of topics below is meant to illustrate some of the coverage, and is not intended to be an exhaustive list.

- Algorithmic game theory
- Computational algebra
- Computational aspects of combinatorics and graph theory
- Computational aspects of communication
- Computational biology
- Computational complexity
- Computational geometry
- Computational learning
- Computational Models and Complexity
- Computational Number Theory
- Cryptography and information security
- Data structures
- Database theory
- Design and analysis of algorithms
- Distributed computing
- Information retrieval
- Operations research
- Parallel algorithms
- Quantum computation
- Randomness in computation

Information for Librarians

Foundations and Trends® in Theoretical Computer Science, 2016, Volume 12, 4 issues. ISSN paper version 1551-305X. ISSN online version 1551-3068. Also available as a combined paper and online subscription.

Foundations and Trends® in
Theoretical Computer Science
Vol. 12, No. 1-2 (2016) 1–274

DOI: 10.1561/0400000051

now
the essence of knowledge

Scalable Algorithms for Data and Network Analysis

Shang-Hua Teng
Computer Science and Mathematics
University of Southern California
shanghua.teng@gmail.com

Contents

Abstract

In the age of Big Data, efficient algorithms are now in higher demand more than ever before. While Big Data takes us into the asymptotic world envisioned by our pioneers, it also challenges the classical notion of efficient algorithms: Algorithms that used to be considered efficient, according to polynomial-time characterization, may no longer be adequate for solving today's problems. It is not just desirable, but essential, that efficient algorithms should be *scalable*. In other words, their complexity should be nearly linear or sub-linear with respect to the problem size. Thus, *scalability*, not just polynomial-time computability, should be elevated as the central complexity notion for characterizing efficient computation.

In this tutorial, I will survey a family of algorithmic techniques for the design of *provably-good* scalable algorithms. These techniques include local network exploration, advanced sampling, sparsification, and geometric partitioning. They also include spectral graph-theoretical methods, such as those used for computing electrical flows and sampling from Gaussian Markov random fields. These methods exemplify the fusion of combinatorial, numerical, and statistical thinking in network analysis. I will illustrate the use of these techniques by a few basic problems that are fundamental in network analysis, particularly for the identification of significant nodes and coherent clusters/communities in social and information networks. I also take this opportunity to discuss some frameworks beyond graph-theoretical models for studying conceptual questions to understand multifaceted network data that arise in social influence, network dynamics, and Internet economics.

S.-H. Teng. *Scalable Algorithms for Data and Network Analysis*. Foundations and Trends® in Theoretical Computer Science, vol. 12, no. 1-2, pp. 1–274, 2016.
DOI: 10.1561/0400000051.

Preface

In 1997, I attended an invited talk given by Shafi Goldwasser at the *38th Annual Symposium on Foundations of Computer Science*. It was a very special talk. The title of her talk, printed in the conference program, "New Directions in Cryptography: Twenty Some Years Later," was modest. However, the talk was beautiful and poetic. In particular, the talk's subtitle, "Cryptography and Complexity Theory: a Match Made in Heaven," has stayed with me after all these years.

The rise of the Internet, digital media, and social networks has introduced another wonderful match in the world of computing. The match between *Big Data and Scalable Computing* may not be as poetic as the match between *Cryptography and Complexity Theory*: Big Data is messier than cryptography and scalable computing uses more heuristics than complexity theory. Nevertheless, this match — although practical — is no less important: "Big Data and Scalable Computing: a Pragmatic Match Made on Earth."

Reasons to Write and People to Thank

I would like start by thanking Madhu Sudan and James Finlay for inviting me to write a survey for *Foundations and Trends in Theoretical Computer Science*, and for their patience, support, and guidance during this long process.

When Madhu and James first reached out to me in the February of 2012 to write a survey on *graph sparsification*, I was noncommittal and

used my busy schedule as the chair of a large department as my excuse. When they came back to me in the Fall of 2012 — knowing that I had successfully become a former chair — I did not reply until the June of 2013, when I had received the confirmation that the USC Daycare finally accepted my 8 month-old daughter Sonia off the wait-list. But during the span of these 16 months, many things happened that were relevant to their initial invitation.

- Nisheeth Vishnoi completed a wonderful and comprehensive survey, titled, $\mathbf{Lx} = \mathbf{b}$ [344], that appeared in the May issue of *Foundations and Trends in Theoretical Computer Science*.

- Joshua Batson, Dan Spielman, Nikhil Srivastava, and I completed our long overdue 8-page article, "Spectral Sparsification of Graphs: Theory and Algorithms," [43] for the *Research Highlights* of *Communications of the ACM*. That article appeared in August 2013.

- Several exciting new results emerged in spectral graph theory that were enabled or inspired by spectral sparsification and scalable Laplacian solvers.

These developments had reduced the need for another longer survey solely devoted to (spectral) sparsification. But I got unexpected encouragement to write a survey from researchers outside my usual theory community. Yan Liu, my machine learning/data mining colleague at USC, invited me to present my work on spectral graph theory and network analysis at the 2012 *SIAM Data Mining Conference*. I gave a talk titled, "Algorithmic Primitives for Network Analysis: Through the Lens of the Laplacian Paradigm," based on my joint work with Dan Spielman. Although the talk was a typical theoretical computer science talk, I was excited by the reception that I received from the Big Data experts: Huan Liu, Joydeep Ghosh, Vipin Kumar, Christos Faloutsos, and particularly Yan Liu, who strongly encouraged me to write a survey on these scalable algorithmic techniques for readers beyond theoretical computer science.

It was a tall order! But given this potential interest from the Big Data community, I reconnected with Madhu and James and proposed

a tutorial to further expand my talk. My goal was to survey some basic theoretical developments (on scalable algorithms) and their techniques that might be useful for practical data/network analysis. The plan was to select a collection of fundamental and illustrative topics of potential practical relevances. I naturally favor problems and algorithms whose rigorous mathematical analysis are clean enough to present for researchers outside theory. Towards this end — in the survey — I selectively encapsulate some "heavy duty" mathematical materials, state them as theorems without proofs, and only expose the relevant essentials aiming to make the survey readable without losing its rigor.

Here, I would like to thank Yan Liu for initiating all this and her valuable feedback on the draft. I thank Madhu and James again for supporting this changed plan and their advice on how to proceed with this writing. I thank Amy Schroeder of the USC Viterbi Engineering Writing Program for editing this monograph, and the anonymous referee and my Ph.D. student Yu Cheng for valuable feedback. I thank Dan Spielman and all my collaborators who have directly contributed to this survey: To Nina Amenta, Reid Andersen, Nina Balcan, Joshua Batson, Marshall Bern, Christian Borgs, Michael Brautbar, Mark Braverman, Jennifer Chayes, Paul Christiano, Wei Chen, Xi Chen, Dehua Cheng, Yu Cheng, Siu-Wing Cheng, Ken Clarkson, David Eppstein, Tamal Dey, John Dunagan, Herbert Edelsbrunner, Matthias Eichstaedt, Michael Elkin, Yuval Emek, Michael Facello, Alan Frieze, Daniel Ford, John Gilbert, Rumi Ghosh, John Hopcroft, Kamal Jain, Jon Kelner, Marco Kiwi, James Lee, Tobin Lehman, Kristina Lerman, Xiangyang Li, Yan Liu, Qi Lu, Aleksander Mądry, Adrian Marple, Gary Miller, Vahab Mirrokni, Richard Peng, Greg Price, Heiko Röglin, Horst Simon, Nikhil Srivastava, Carl Sturtivant, Dafna Talmor, Bill Thurston, Steve Vavasis, Konstantin Voevodski, Noel Walkington, Yu Xia, and Xiaoran Yan, thank you!

Theory and Practice

While I believe in the importance of provably-good algorithms in data and network analysis, my own experiences at Intel, NASA, and Aka-

mai have also taught me the limitation of "provably-good algorithms." The gap between proof-based theory and relevance-based practice is beyond the limitation of the worst-case or average-case analyses. The theory-practice gap exists also because essentially all measures of qualities — from the centrality of a node in a network, to the similarity between two datasets/networks, to the coherence of a network cluster and community — may have their limitations.

> *Many conceptual questions that arise in modeling real-world data are fundamentally challenging.*

Thus, as much as I would like — in this survey — to make an algorithmic connection between theory and practice in the area of data and network analysis, I would also like readers to approach this survey with an open mind: Theory is usually my guide for understanding practical problems, but theoretical thinking has too often been the obstacle that makes me struggle in connecting with practice. On the few occasions that theoretical thinking provided me with the insight to make a connection, I was elated. But indeed, I usually found myself unable to balance the connection between theory and practice, and then decided to do theory for its mathematical beauty.

For example, in theoretical computer science, we have also encountered the notion of clusterability in various settings, including VLSI layout, parallel processing, network clustering, community identification, and more generally, the design of divide-and-conquer algorithms for matrix/graph problems. However, as illustrated throughout this survey, I remain unsure about how to evaluate and validate the relevance of various mathematical notions of clusterability, particularly the conductance or cut-ratio measures that I have been studying for more than a decade. We use these partition/clusterability measures because they appear to be reasonable, and because, using them, we have obtained mathematical proofs that are aligned with our intuition.

My Ph.D. advisor Gary Miller once said to me, "coping with the uncertainty between theory and practice gives rise to plausible and sometimes good research questions, but questioning the certainty can lead to excellent questions." So I do think it is more than reasonable to question and challenge every notion one chooses.

Discrete vs Continuous and Beyond

I would like to conclude the preface by remarking that several subjects of this survey are at the intersection between combinatorial optimization and numerical analysis. Thus, I think they serve as good examples of the interplay between combinatorial thinking and numerical thinking [330]. While it is more conventional to view many network analysis problems as graph-theoretical problems, it can often be constructive to view them as numerical, statistical, or game-theoretical subjects as well. Network data is richer than its graph representation, and network science is beyond graph theory.

Numerical thinking is also more than numerical analysis — it is a creative process of discovering useful numerical connections that may not be apparent [330]. For example, in the 70s, Hall [162], Donath and Hoffman [115, 116], and Fiedler[1] [133, 134] made insightful connection between graphs and matrices — beyond just the matrix representation of graphs — which set the stage for spectral graph theory. The field of algorithm design has benefited greatly from the deep connection between graph properties (such as connectivity, conductance, and mixing time) and algebraic properties (spectral bounds of Laplacian/adjacency/random-walk matrices [82]). Over the last decade, scientists have made even broader and deeper connections between numerical solutions and network solutions [211, 73], between numerical representations and digital representations [110, 108], between

[1]To the memory of Miroslav Fiedler (1926 – 2015): It is difficult to overstate the impact of Fiedler's work to spectral graph theory. His paper, "Algebraic Connectivity of Graphs," [133] established a far-reaching connection between graph theory and linear algebra. Fiedler's spectral theorem, together with Koebe's disk-packing characterization of planar graphs [213] (see Theorem 5.34), Sperner's lemma for Brouwer's fixed-point theorem [310], and Cheeger's inequality [82] (see Theorem 4.2), are my favorite mathematical results — they beautifully connect continuous mathematics with discrete mathematics. I have always cherished my only meeting with Fiedler. After my talk, "The Laplacian Paradigm: Emerging Algorithms for Massive Graphs," [329] at *7th Annual Conference on Theory and Applications of Models of Computation*, Jaroslav Nešetřil (my former officemate at Microsoft Research Redmond) thoughtfully invited my wife Diana (a US historian) and I to a dinner with him and Fiedler. I still vividly remembered that special evening in the beautiful Prague on June 10, 2010. At age eighty four, Miroslav was charming and talkative, not just about mathematics but also about history.

numerical methods and statistical methods [111, 118], between numerical concepts and complexity concepts [314], and between numerical formulations and privacy formulations [121]. Numerical analysis has played an increasing role in data analysis through dimension reduction [111] and in machine learning through optimization.

In the examples of this survey, the Laplacian paradigm [319] has not only used numerical concepts such as preconditioning to model network similarity and graph sparsification, but also used combinatorial tools to build scalable solvers for linear systems [313, 318, 317, 319], Gaussian sampling [86], and geometric median [98]. Scalable techniques for PageRank approximation [65] have also led to algorithmic breakthroughs in influence maximization [64, 325, 324] and game-theoretical centrality formulation [84]. These results illustrate the rich connection between network sciences and numerical analysis.

A Quick Guide of the Survey

The survey begins with two background chapters. Chapter 1 discusses scalability measures for evaluating algorithm efficiency. It also introduces the basic characterizations of scalable algorithms, which are the central subjects of this survey. Chapter 2 reviews mathematical models for specifying networks, and highlights a few basic problems in data and network analysis. We will use these problems as examples to illustrate the design and analysis of scalable algorithms in the technical chapters that follow the background chapters. In Section 1.5, I will give a more detailed outline of these technical chapters.

Acknowledgements

The writing of this article is supported in part by a Simons Investigator Award from the Simons Foundation and the NSF grant CCF-1111270.

Shang-Hua Teng
Los Angeles

1

Scalable Algorithms

In light of the explosive growth in the amount of available data and the diversity of computing applications, efficient algorithms are in higher demand now more than ever before.

1.1 Challenges of Massive Data

Half a century ago, the pioneers of Theoretical Computer Science began to use asymptotic analysis as the framework for complexity theory and algorithm analysis [167]. At the time, computers could only solve small-scale problems by today's standards. For example, linear programs of fifty variables, linear systems with a hundred variables, or graphs with a thousand vertices, were considered large scale. In those days, the world's most powerful computers (e.g., the IBM main frames) were far less capable than the iPhones of today. Even though these pioneers were mindful that constant factors did matter for practical computing, they used asymptotic notations — such as Big-O — to define complexity classes such as P and NP to characterize efficient algorithms and intractable problems [122, 100, 233, 189, 141]. Asymptotic complexity simplifies analyses and crucially puts the focus on the order of the

leading complexity terms of algorithms [103, 209, 307]. However, with respect to the size of practical inputs at the time, the asymptotic world seemed to belong to the distant future.

Our pioneers persisted because they had a vision that one day computational problems would be massive, and therefore, the rate of complexity growth in relation to the growth of the inputs would be essential for characterizing the efficiency of an algorithm as well as the computational difficulty of a problem.

The asymptotic world has arrived with the rise of the Internet!

Today, the Web has grown into a massive graph with trillions of nodes; social networks and social media have generated an unbounded amount of digital records; smart phones have produced billions of images and videos; and tasks of science and engineering simulation have created equations and mathematical programs that involve hundreds of millions of variables. Even our computing devices have grown rapidly in size and complexity: In the middle 90s, the Intel Pentium processor had 2 millions transistors; today's PCs contain more than a billion.

While *Big Data* has taken algorithm design into the asymptotic world envisioned by our pioneers, the explosive growth of problem size has also significantly challenged the classical notion of *efficient algorithms*, particularly the use of *polynomial time* as the characterization of *efficient computation* [122]: Algorithms that used to be considered efficient (in the classification according to P) — such as a neat $O(n^2)$-time or $O(n^{1.5})$-time algorithm — may no longer be adequate for solving today's problems.

> *Therefore, more than ever before, it is not just desirable, but essential, that efficient algorithms should be* scalable*. In other words, their complexity should be nearly linear or sublinear with respect to the problem size. Thus, scalability — not just polynomial-time computability — should be elevated as the central complexity notion for characterizing efficient computation.*

1.2 The Scalability of Algorithms

The *scalability* of an algorithm measures the growth of its complexity — such as the running time — in relation to the growth of the problem size. It measures the capacity of an algorithm to handle *big* inputs.

Suppose Π is a computational problem[1] whose input domain is Ω. For each instance $x \in \Omega$, let $\text{size}(x)$ denote the *size* of input x. The input domain Ω can be viewed as the union of a collection of subdomains $\{..., \Omega_n, ...\}$, where Ω_n denotes the subset of Ω with input size n.

Now, suppose A is an algorithm for solving Π. For $x \in \Omega$, let $T_A(x)$ denote the time complexity for running $A(x)$. Instead of directly using *instance-based complexity* $T_A(x)$ to measure the performance of algorithm A for solving x, we consider the following related quality measure:

Definition 1.1 (Instance-Based Scalability)**.** The *scalability* of an algorithm A for solving an instance $x \in \Omega$ is given by:

$$\text{scalability}(A, x) = \frac{T_A(x)}{\text{size}(x)} \tag{1.1}$$

We now *summarize* the instanced-based scalability of algorithm A over all instances in Ω_n as:[2]

$$\text{scalability}_A(n) := \sup_{x \in \Omega_n} \text{scalability}(A, x) = \sup_{x \in \Omega_n} \frac{T_A(x)}{\text{size}(x)}.$$

Then, $\text{scalability}_A(n)$ is a function that measures the growth of the complexity of A in relation to the growth of the problem. Let $T_A(n) = \sup_{x \in \Omega_n} T_A(x)$ denote the (worst-case) complexity of algorithm A on inputs of size n. Note that:

$$\text{scalability}_A(n) = \frac{T_A(n)}{n}.$$

Thus, A is a polynomial-time algorithm iff $\text{scalability}_A(n)$ is polynomial in n. However, the scalability measure puts the focus on scalable algorithms:

[1] See Appendix A.1.1 for a review of the basic types of computational problems.

[2] We may also use other *beyond worst-case* formulae for performance summarization [314]. For more discussion, see Section 8.4.

Definition 1.2 (Scalable Algorithms)**.** An algorithm A is *scalable* if there exists a constant $c > 0$ such that:

$$\text{scalability}_A(n) = O(\log^c n).$$

In the special case when $c = 0$, we say A is *linearly-scalable*. We say algorithm A is *super-scalable*, if $\text{scalability}_A(n) = o(1)$. Super-scalable algorithms have a complexity that is sub-linear in problem size. Thus, necessarily, these algorithms must find solutions *without examining the entire input data set*. Sampling and local data/network exploration are two basic tools for designing super-scalable algorithms, and will be the subjects of Chapters 3 and 4.

***Remark**: One may say that* $\text{scalability}_A(n)$ *encodes no more information than* $T_A(n)$ *about algorithm A, because* $\text{scalability}_A(n) = T_A(n)/n$*. For example, A is scalable if and only if* $T_A(n)$ *is* nearly linear[3] *in n as referred to in [313]. However, the former identifies scalability as an essential concept for the characterization of efficient algorithms.*

> *The scalability measure puts the emphasis not on polynomial-time algorithms, but on scalable algorithms. It highlights the* exponential *gap between* $O(\log^c n)$ *and n, and between scalable algorithms and quadratic-time algorithms.*

To capture the essence of scalable algorithms, throughout the article, we will adopt the following commonly-used asymptotic notation.

Definition 1.3 ($\widetilde{O}$-Notation)**.** For a given function $g(n)$, we denote by $\widetilde{O}(g(n))$ the set of functions:

$$\widetilde{O}(g(n)) = \{f(n) : \exists \text{ constant } c > 0, \text{ such that } f(n) = O(g(n)\log^c g(n))\}$$

For any positive integer n, we also use $\widetilde{O}_n(1)$ to denote:

$$\{f(n) : \exists \text{ constant } c > 0, \text{ such that } f(n) = O(\log^c n))\}.$$

---

[3]We say A is an *almost* linear-time algorithm (or is *almost scalable*) if $T_A(n) = O(n^{1+o(1)})$, i.e., $\text{scalability}_A(n) = n^{o(1)}$.

In other words, $\widetilde{O}$ is a variant of the asymptotic O-notation that hides additional poly-logarithmic factors. For example, $n\log^3 n = \widetilde{O}(n)$. With this notation, an algorithm is scalable if its scalability measure on an input of size n is $\widetilde{O}_n(1)$. In other words, its complexity on an input of size n is $\widetilde{O}(n)$.

***Remarks**: The scalability analysis of algorithms is not unique to Big Data. For example, in parallel processing [71, 223], the notion of scalability is used to measure the efficiency of a parallel algorithm in utilizing parallel machines: Let $T_A(p,n)$ denote the parallel complexity of an algorithm A on a machine with p processors. So, the* speed-up *of this parallel algorithm with respect to the sequential one is $\frac{T_A(n)}{T_A(p,n)}$.*

Then, $\frac{T_A(n)}{T_A(p,n)} \cdot \frac{1}{p}$ measures the ratio of the achievable speedup to maximum-possible speedup by running A on p processors. In this context, a parallel algorithm A is linearly-scalable *if this ratio is bounded from below by a constant. Although the focus of scalability analysis of parallel algorithms is different from the scalability analysis of sequential algorithms, we can draw on insights from previous studies.*

1.3 Complexity Class S

A basic step in algorithm analysis and complexity theory is to characterize the family of problems that have efficient algorithmic solutions. In the world of *Big Data*, instead of using the traditional polynomial-time as the criterion for efficient algorithms, we require that efficient algorithms must be scalable.

Definition 1.4 (Complexity Class S)**.** We denote by S the set of computational problems that can be solved by a deterministic scalable algorithm.

In other words, a computational problem Π is in class S if there exists an algorithm A that solves Π with $\text{scalability}_A(n) = \widetilde{O}_n(1)$.

Complexity class S is analogous to the traditional complexity classes P and FP. In complexity theory, one usually classifies computational problems into three basic types [307, 103]: *decision problems*, *search*

problems, and *optimization problems* (see Appendix A.1.1 for a quick review). Formalism is needed to precisely define complexity classes in order to address the subtlety among different types of computational problems. For example, to use polynomial time as the benchmark for efficient computation, the class P is usually reserved only for decision problems that can be solved in polynomial time without randomization [307]. The search version of complexity class P is known as FP, i.e., the class of functions that can be deterministically computed in polynomial time. In this article, we intend to be less formal in this regard, so that the focus will be on the notion of scalability rather than the difference between decision, search, and optimization problems.

Definition 1.5 (Complexity Class RS)**.** We denote by RS the set of computational problems that can be solved by a randomized scalable algorithm.

Randomization also introduces its own subtlety, which has given rise to classes such as BPP, RP, and ZPP for decision problems. For example, ZPP denotes the set of problems that can be solved by a *Las Vegas* algorithm with an expected polynomial runtime, which makes no errors in all instances. BPP and RP relax the latter condition by allowing the polynomial-time randomized algorithms to make bounded errors. Algorithms for RP are allowed to make errors only for YES instances, but algorithms for BPP are allowed to make errors for both YES and NO instances.

Again, with regard to randomization, we intend to be less formal as well so the focus will be on the notion of scalability rather than different types of errors.

Remarks*: In computational complexity theory, one has to first define a computational model in order to define a complexity class. Commonly used models are Turing machines and random-access machines (RAM). It's well known that complexity classes, such as* P *and* FP*, are essentially robust with respect to these models.*

The scalable classes S *and* RS *can be formally defined according to these computational models. Their robustnesses with respect to computational models require further investigation, but are outside the scope of*

this article. As universal as these computational models are, data and network analysis programs on Turing machines or abstract random-access machines could be cumbersome.

However, as Sipser said [307], "Real computers are quite complicated — too much so to allow us to set up a manageable mathematical theory of them directly." In the world of Big Data, massive networks, and large-scale optimization, without getting bogged down with details, I encourage readers to think about the real RAM model of Blum, Shub, and Smale [54], but with unit-cost computation of basic rational operations over reals at a given machine precision $\epsilon_{machine}$.

1.4 Scalable Reduction and Algorithmic Primitives

Algorithm design for scalable computing is like building a software library. Once we develop a new scalable algorithm, we can add it to our scalable library, and use it as a subroutine to design the next wave of scalable algorithms.

At the heart of this perspective is the notion of *scalable reduction*.

Definition 1.6 (Scalable Reduction)**.** A computational problem Π (over domain Ω) is S-reducible to another computational problem Π' (over domain Ω'), denoted by $\Pi \leq_{\mathsf{S}} \Pi'$, if the following is true: Given any solver B for Π', there exists an algorithm A for solving Π such that for every instance $x \in \Omega$, $A(x)$ takes $\widetilde{O}\left(\text{size}(x)\right)$ steps including (i) generating a collection of instances $y_1, ..., y_{L(x)} \in \Omega'$ with:

$$\sum_{i=1}^{L(x)} \text{size}(y_i) = \widetilde{O}\left(\text{size}(x)\right)$$

and (ii) making calls to B on these instances.

In other words, $\Pi \leq_{\mathsf{S}} \Pi'$ if Π has a scalable *Turing-Cook-reduction* to Π'. In this definition, A is assumed to be a deterministic algorithm. We say $\Pi \leq_{\mathsf{RS}} \Pi'$ if we use a randomized algorithm A in the definition above. Directly from the definition, we have:

Proposition 1.7. (1) $\Pi \leq_{\mathsf{S}} \Pi'$ and $\Pi' \leq_{\mathsf{S}} \Pi''$ imply $\Pi \leq_{\mathsf{S}} \Pi''$. (2) $\Pi \leq_{\mathsf{RS}} \Pi'$ and $\Pi' \leq_{\mathsf{RS}} \Pi''$ imply $\Pi \leq_{\mathsf{RS}} \Pi''$.

Proposition 1.8. (1) If $\Pi \leq_{\mathsf{S}} \Pi'$ and $\Pi' \in \mathsf{S}$, then $\Pi \in \mathsf{S}$. (2) If $\Pi \leq_{\mathsf{RS}} \Pi'$ and $\Pi' \in \mathsf{RS}$, then $\Pi \in \mathsf{RS}$.

The field of computing has produced a number of remarkable scalable algorithms in various applicational domains. The following are a few examples:

- **Basic Algorithms**: FFT, merge sort, median selection, Huffman codes
- **Graph Algorithms**: minimum (maximum) spanning trees, shortest path trees, breadth-first search, depth-first search, connected components, strongly connected components, planar separators
- **Optimization Algorithms**: linear programming in constant dimensions
- **Probabilistic Algorithms**: VC-dimension based sampling, quick sort
- **Data structures**: many wonderful data structures
- **Numerical and Geometric Algorithms**: the multigrid method, the fast multipole method, 2D Delaunay triangulations and Voronoi diagrams, 3D convex hulls, quadtrees and its fixed dimensional extensions, ϵ-nets, nearest neighbors, and geometric separators in any fixed dimensions

More recently, *property testing* [154, 288], a subfield of theoretical computer science — inspired by PAC learning [340], holographic proofs

[37], and the PCP theorem [31] — has led to thriving developments of sub-linear-time algorithms [291]. These results, together with the work of Vapnik-Chervonenkis [342] and Johnson-Lindenstrauss [180], have demonstrated the power of sampling in scalable algorithm design. On the practical side, a rich body of scalable algorithms has been developed in fields of network science [184, 183, 230, 357, 354, 349, 64, 325, 324, 84], machine learning [182, 218, 143], and numerical computing [68, 190].

1.5 Article Organization

In this article, we will survey scalable algorithmic techniques, particularly those based on rapid progress in spectral graph theory, numerical analysis, probabilistic methods, and computational geometry. Many of these techniques are simple on their own, but together they form a powerful toolkit for designing scalable algorithms. After a brief review of network models in Chapter 2, we will proceed with the technical chapters of this article as follows:

In Chapter 3, "Significant Nodes: Sampling — Making Data Smaller," we will start by focusing on the smallest structures in big networks — ***nodes***. We will discuss results from [65] that incorporate efficient local network exploration methods into advanced sampling schemes.

Both sampling and local network exploration are widely used techniques for designing efficient algorithms. Here, the combination of the two leads to the first super-scalable algorithm for identifying *all* nodes with significant PageRank values in *any* network [65].[4] Leaving the details of local network exploration for the next chapter, this short chapter will focus on an *annealing* approach to construct robust PageRank estimators. This approach uses a multi-precision sampling scheme to

[4]This result was covered by Richard Lipton and Kenneth Regan, under title "Shifts In Algorithm Design," on their popular blog *Gödel's Lost Letter and P=NP* (https://rjlipton.wordpress.com/2014/07/21/shifts-in-algorithm-design/). The surprising conclusion that one can in fact identify all nodes with significant PageRank values without examining the entire network also landed this result on the list of "Top ten algorithms preprints of 2012" by David Eppstein (http://11011110.livejournal.com/260838.html).

guide a local *statistics-gathering algorithm*. We will show that this annealing method can build a robust PageRank estimator by visiting only a sub-linear number of nodes in the network. The reverse-structural techniques of this algorithm have been used in subsequent scalable algorithms for computing other network centrality measures [84], and for influence maximization [64, 325, 324].

Keywords: *Markov processes; PageRank; personalized PageRank; PageRank matrix; multi-precision sampling; multi-precision annealing; Riemann estimator.*

In Chapter 4, "Clustering: Local Exploration of Networks," we will turn our attention to slightly larger structures in networks — ***clusters***. We will survey the basic framework of *local network exploration algorithms* introduced in [318]. These algorithms conduct *locally expandable searches* of networks: Starting from a small set of input nodes of a network, local algorithms iteratively expands its knowledge about the "hidden" network by only exploring the neighborhood structures of the explored nodes.

We first discuss a family of provably-good local clustering algorithms[5] [318, 22, 93, 24, 23]. We will then analyze two scalable algorithms for personalized-PageRank approximation. These local algorithms highlight the usefulness of graph-theoretical concepts, such as random walks, personalized PageRank, and conductance, for analyzing network structures, both mathematically and algorithmically. We then conclude the section with a study of the interplay between network structures and dynamic processes (such as random walks, social influence, or information propagation). In particular, we will discuss the framework introduced in [146] for quantifying the impact of this interplay on the *clusterability* of subsets in the network, and prove a parameterized Cheeger's inequality [82].

This chapter, together with Section 5.8 Chapter 6 and Chapter 7, also provide us with a quick tour through spectral graph theory.

[5]Several of these local clustering algorithms have been implemented by Konstantin Voevodski of Google (http://gaussian.bu.edu/lpcf.html) and used in the context of protein network analysis.

> **Keywords**: *clusterability measures; conductance; Laplacian matrix; Cheeger's inequality; sweep with eigenvectors; local network-analysis algorithms; local clustering; power methods; random-walk sampling; interplay between dynamic processes and networks; spectral graph theory.*

In Chapter 5, "Partitioning: Geometric Techniques For Data Analysis," we will focus on ***networks defined by geometric data.*** We illustrate the power of geometric techniques — such as spatial decomposition and divide-and-conquer — in scalable data analysis. The geometric structures, such as nearest neighborhood graphs, also offer potentially useful measures of clusterability and cluster stability [38], which have both structural and algorithmic consequences. We will discuss a family of geometric partitioning techniques,[6] and apply them to the computation of nearest neighborhood graphs and geometric clusters. These techniques lead to powerful scalable geometric divide-and-conquer schemes [252, 253]. They also provide a beautiful bridge between spectral graph theory and network analysis, involving a popular spectral partitioning method [315].

> **Keywords**: *nearest neighborhood graphs; geometric graphs; geometric partitioning; spectral partitioning; separator theorems; centerpoints; evolutionary algorithms, conformal maps; geometric divide-and-conquer; VC dimensions; random projection; spectral projection.*

In Chapter 6, "Spectral Similarity: Sparsification — Making Networks Simpler," we will focus on what is means to say "one network is similar to another network." We will address three basic questions:

- **Conceptual Question**: *How should we measure the similarity between two given networks?*
- **Mathematical Question**: *Does every graph have a "similar" sparse graph?*

[6] Many algorithms discussed in this chapter have been implemented [148] and are available in MESHPART, a Matlab mesh partitioning and graph separator toolbox (http://www.cerfacs.fr/algor/Softs/MESHPART/).

- **Algorithmic Question**: *Is there a scalable algorithm for constructing a good sparsifier?*

This is my favorite subject in spectral graph theory.

> **Keywords**: *spectral similarity; cut-similarity; effective resistances; spectral sparsification; low-stretch spanning tree; matrix sampling; conjugate gradient; PageRank completion of networks.*

In Chapter 7, "Electrical Flows: Laplacian Paradigm for Network Analysis," we survey a ***family of scalable algorithmic toolkits***. At the heart of these toolkits is a scalable solver for Laplacian linear systems and electrical flows [313, 318, 317, 319]. This scalable Laplacian solver has initiated and enabled many new scalable algorithms for spectral approximation [319], geometric and statistical approximation [98], graph embedding [335], machine learning [358], numerical methods, random-walk approximation, and Gaussian sampling in graphical models [86].

These applications illustrate a powerful, general algorithmic framework — called the *Laplacian paradigm* — for network analysis. In this framework, we attempt to reduce an optimization problem to one or more linear algebraic or spectral graph-theoretical problems that can be solved efficiently by the Laplacian solver or primitives from this scalable family. In addition to the applications above, this framework has also led to several recent breakthrough results — including scalable algorithms for max-flow/min-cut approximation [303, 197, 275, 243, 282] and sparse Newton's method [86] — that have gone beyond the original scalable Laplacian linear solvers [319, 215, 200, 97, 217].

These success stories point to an exciting future for scalable algorithm design in data and network analysis. I hope that continuing advancements will help enrich our scalable algorithmic library, and inspire new efficient solutions for data and network analysis.

> **Keywords**: *SDD primitive; electrical flows; Laplacian linear systems; spectral approximation; graph embedding; machine learning; Gaussian sampling; Gaussian Markov random fields; random walk sparsification; sparse Newton's method; Laplacian paradigm.*

In Chapter 8, "Remarks and Discussions," we conclude this article with a few "inconclusive" remarks. The inconclusiveness reflects the conceptual challenges that we usually face in data and network analysis. We also discuss a few frameworks beyond the commonly-used graph-theoretical network models to address fundamental conceptual questions in data analysis and network science.

Keywords: *centrality; clusterability; k-means methods; multifaceted network data; beyond graph-based network models; incentive networks; interplay between influence processes and social networks; social choice theory; axiomatization; game theory; cooperative games; Shapley value; behaviors of algorithms; beyond worst-case analysis.*

A.1 Appendix to the Chapter

A.1.1 Basic Types of Computational Problems

In complexity theory, one usually classifies basic computational problems into three types [307, 103]:

Decision Problems: A decision problem concerns the membership of a language $L \subseteq \{0,1\}^*$: Given an input string $x \in \{0,1\}^*$, one is asked to determine if $x \in L$. The output of a decision problem has constant complexity as the answer is either YES or NO (or DON'T KNOW, when randomization is used). This family of problems is commonly used to capture the computational challenge in deciding whether or not an input problem has a feasible solution.

Search Problems (or function problems:) A search problem typically works with a binary relation, $R \subseteq \{0,1\}^* \times \{0,1\}^*$: Given an input x, one is asked to determine if there exists y such that $(x,y) \in R$, and furthermore, when such a y exists, one must also produce an element from $\text{solution}_R(x) = \{y \in \{0,1\}^* | (x,y) \in R\}$. A search problem has two basic size measures: the size of the input x and the size of an output y. Thus, the complexity for solving a search problem is measured by a function in terms of either or both of these sizes. If the scalability of an algorithm A (for a search problem Π) is bounded by a poly-logarithmic

function with respect to the size of the output it produces, then we say that A is *output-scalable*.

Optimization Problems: In a basic constrained optimization problem, we have a single utility/cost function u, whose value depends on multiple decision parameters $(x_1, x_2, ..., x_n)$. Each x_i has its own domain $x_i \in \Omega^{(i)}$, and the feasible region is given by a global constraint:

$$(x_1, x_2, ..., x_n) \in C.$$

The optimization problem could either be a maximization or a minimization problem:

$$\begin{array}{rl} \text{optimize} & u(x_1, x_2, ..., x_n) \\ \text{subject to} & (x_1, x_2, ..., x_n) \in C \quad \text{and} \quad x_i \in \Omega^{(i)}, \ \forall\ i. \end{array}$$

An input instance of an optimization problem is given by a representation of $(u, C, \Omega^{(1)} \times \cdots \times \Omega^{(n)})$. Like search problems, the complexity of an optimization problem (or an optimization algorithm) can be measured in terms of either or both input and output sizes.

Beyond Decision, Search, and Optimization: Other types of computational problems exist. For a binary relation R, for example, when given an input x, the *counting problem* aims to determine the number of solutions in $|\text{solution}_R(x)|$, the *enumeration problem* needs to identify all members in $\text{solution}_R(x)$, while the *sampling problem* generates an element from $\text{solution}_R(x)$, chosen according to uniform or a given distribution over $\text{solution}_R(x)$. The *multi-objective optimization problem* captures the potential trade-offs among several — possibly competing — objective functions in *multiple-criteria* decision-making:

$$\begin{array}{rl} \text{optimize} & u_1(x_1, x_2, ..., x_n), ..., u_k(x_1, x_2, ..., x_n) \\ \text{subject to} & (x_1, x_2, ..., x_n) \in C \quad \text{and} \quad x_i \in \Omega^{(i)}, \ \forall\ i. \end{array}$$

A basic solution concept for multi-objective optimization is the *Pareto set*, which contains every feasible solution not strictly dominated by other feasible solutions.

The *game-theoretical problem* captures possible compromises among multiple decision makers in strategic decision-making, where each decision maker has his/her own utility function and can only determine

his/her own subset of decision parameters. The basic solution concept in game theory is the *Nash Equilibrium* [259, 258]. Schematically, there are n players. The i^{th} player has utility function u_i, and controls only input parameter $x_i \in \Omega^{(i)}$. These players have to jointly set their decision parameters in order to satisfy a global constraint: $(x_1, ..., x_n) \in C$. Each player's utility usually depends on all decision parameters. The domain $\Omega^{(i)}$ is referred to as the *strategy space* for player i. The simplest example of a game is a two-player matrix game [256, 259, 258].

We can formulate the computation of *Pareto points* and *Nash equilibria* as search problems [104, 268, 269, 272, 287]. However, we can capture more complex real-world phenomena using multi-objective optimization and game theory than with search and optimization [271, 269, 270].

A.1.2 Convention for Basic Notation

In this article, we will largely follow the conventions below for mathematical notation [314]:

- *Lower-case English and Greek letters*:
  - Scalar constants, variables, and functions
  - Vertices in a graph
- *Upper-case English and Greek letters*:
  - Sets and graphs
  - Distributions and events
  - Constants (occasionally)
- *Lower-case English and Greek letters in bold font*:
  - Vectors

    If an n-dimensional vector is denoted by $\boldsymbol{v}$, then we assume $\boldsymbol{v} = (v_1, v_2, ..., v_n)^T$, where T denotes the transpose operator. We use v_i or $\mathbf{v}[i]$ to denote the i^{th} entry of $\boldsymbol{v}$. We always assume $\mathbf{v}$ is a column vector. Thus, its transpose, $\mathbf{v}^T$, is a row vector.

  - Permutation (mostly in Greek letters)
    If $\boldsymbol{\pi}$ denotes a permutation of n elements, then we use π_i or $\boldsymbol{\pi}[i]$ to denote its i^{th} element.

- *Upper-case English and Greek letters in bold font*:

  - Matrices
    If an $m \times n$-matrix vector is denoted by $\boldsymbol{M}$, then we use $m_{i,j}$ or $\mathbf{M}[i,j]$ to denote its $(i,j)^{th}$ entry.

- *Special matrices and vectors*:

  - $\mathbf{I}_n$ denotes the identity matrix in n dimensions. When it is clear from the context, we use $\mathbf{I}$ to denote the identity matrix of the assumed dimensions.
  - $\mathbf{1}$ and $\mathbf{0}$, respectively, denote the vectors of all 1s or 0s in the assumed dimensions.
  - For $v \in [n]$, $\mathbf{1}_v$ denotes the n-place vector that has 1 at entry v and 0 everywhere else.
  - For $S \subset [n]$, $\mathbf{1}_S$ denotes the n-place vector that has 1 at entries in S and 0 everywhere else.

- $[n]$ or $[1:n]$ denotes the set of integers between 1 and n. More generally, for integers $a \leq b$, $[a:b]$ denotes the set of integers between a and b.

- *Matrix entry-wise inequality*

  - For two $m \times n$ matrices $\mathbf{A}$ and $\mathbf{B}$, and parameters $\epsilon > 0$ and $c > 0$, we use $\mathbf{A} \leq c \cdot \mathbf{B} + \epsilon$ to denote $a_{i,j} \leq c \cdot b_{i,j} + \epsilon$, $\forall i \in [m], \forall j \in [n]$.

- Respectively, log and ln denote the logarithm base 2 and the natural logarithm.

- The indicator random variable for an event A is $\mathbf{I}[A]$ or $[A]$.

- Respectively, $\Pr_D[A]$ and $\mathrm{E}_D[X]$ denote the probability of event A and the expectation of variable X, over a distribution D.

2

Networks and Data

Real-world network data can be complex. But at the most basic level, a network can be modeled as a graph $G = (V, E)$, which characterizes the structure of the network in terms of *nodes* (Webpages, Internet routers, scholarly articles, or people) and edges (links, connections, citations, or friends). Depending on applications, the graph may be directed or undirected. Examples of directed networks include: (1) the Web, where nodes represent Webpages and edges represent hyperlinks, (2) Twitter: where edges reflect "following" and "retweeting," and (3) the citation graphs of scholarly publications. Meanwhile, Facebook forms a massive undirected graph, with each undirected edge representing a pair of "Facebook friends."

2.1 Weighted Graphs and Affinity Networks

In general, edges usually have additional features that characterize the *level* of pair-wise interaction or *closeness* in a network. Two basic classes of weighted graphs often appear in applications. The first class consists of *distance networks*, where each edge $e \in E$ is assigned a number $l_e \geq 0$ representing the *length* of edge e; for $e \notin E$, we usually assume $l_e = +\infty$.

The network of highways between cities is a wonderful example of a distance network. The second class consists of *affinity networks*, where each edge $(u, v) \in E$ is assigned a weight $w_{u,v} \geq 0$ representing u's *affinity weight* towards v; for $(u, v) \notin E$, we assume $w_{u,v} = 0$. It is usually more natural to view a social network as an affinity network.

Unless otherwise stated, we will focus on affinity networks in most of this article. We can define an affinity network based on a given distance network, and vice versa, to reflect the intuition that the closer two nodes are, the higher their affinity becomes. A standard transformation between affinity and length, as we will discuss in more details in Chapters 6 and 7, is $w_{u,v} = \frac{1}{l_{u,v}}$. In such a setting, as will become clear in Chapter 7, we will refer to $l_{u,v}$ and $w_{u,v}$, respectively, as the *resistance* and *conductance* of the edge between u and v, and view the network as an *electrical network of resistors* [313, 318, 317, 319, 91].

An affinity network with n nodes can be mathematically represented as a weighted graph $G = (V, E, \mathbf{W})$. Unless otherwise stated, we assume $V = [n]$ and $\mathbf{W}$ is an $n \times n$ non-negative matrix. We will follow the convention that for $i \neq j$, $w_{i,j} = 0$, if and only if, $(i, j) \notin E$. If $\mathbf{W}$ is a symmetric matrix, then we say the network is *undirected*. If $w_{i,j} \in \{0, 1\}$, $\forall i, j \in V$, then we say the network is *unweighted*. Although not always the case, we sometimes assume $w_{i,i} = 0$ for $i \in V$. We usually assume that the affinity weights are *scaled*, that is, $0 \leq w_{i,j} \leq 1$. A special family of the scaled affinity networks is the following:

Definition 2.1 (Normalized Affinity Networks)**.** A *normalized affinity network* uses a non-negative *weighted affinity vector*:

$$\mathbf{a}_u^T = (a_{u,1}, ..., a_{u,n}), \quad \text{where } \sum_{v \in V} a_{u,v} = 1$$

to specify the degree of each member's preferences for all members in V. We refer to $\mathbf{A}$ — whose u^{th} row is $\mathbf{a}_u^T$ — as a *normalized affinity matrix*. It defines an affinity network $G = (V, E, \mathbf{A})$.

Normalized affinity matrices are also known as *Markov matrices* or *stochastic matrices*. Given any weighted network $G = (V, E, \mathbf{W})$, one

can derive several normalized affinity matrices. These matrices will be covered in later chapters. They include the *personalized PageRank matrix* (Definition 3.5) and the basic *random walk matrix* $\mathbf{D}_{\mathbf{W}}^{-1}\mathbf{W}$, where $\mathbf{D}_{\mathbf{W}}$ is the diagonal matrix of row sums of $\mathbf{W}$.

Weighted graphs allow us to model both the structure and the strengths of interactions among nodes in a network. For instance, consider the following possible scenario of social interactions: Suppose there are n people, and the affinities among them in the network are specified by a normalized affinity matrix $\mathbf{A}$, as given in Definition 2.1. Suppose further, each person u has an *attention capacity* c_u, and distributes her attention capacity based on her affinities to others, say, according to an *attention distribution function*[1] $\theta_u : [0,1] \to \mathbb{R}_+$. For person v in the network, u's attention to v is:

$$c_{u,v} := \left(\frac{\theta_u(a_{u,v})}{\sum_{v \in V} \theta_u(a_{u,v})} \right) \cdot c_u.$$

Note that $\mathbf{c}_u = (c_{u,1}, ..., c_{u,n})^T$ satisfies $\sum_{v \in V} c_{u,v} = c_u$. We can model this social interaction as the weighted graph $G = (V, E, \mathbf{C})$, where the u^{th} row of $\mathbf{C}$ is $\mathbf{c}_u^T$.

2.2 Possible Sources of Affinities

For practical applications, the following are two popular data models for defining pair-wise affinity weights of networks. The first one is defined geometrically, while the second is defined by features.

Metric Models

In this model, one assumes that there exists an underlying "geometric" space, $\mathcal{M} = (V, \text{dist})$, that impacts the interactions among nodes in the network. Usually, the *distance function* satisfies the basic properties of a metric space, that is, for all $u, v, w \in V$, (i) $\text{dist}(u, v) \geq 0$, and $d(u, v) = 0$ iff $u = v$, (ii) $\text{dist}(u, v) = \text{dist}(v, u)$, and (iii) $\text{dist}(u, v) \leq \text{dist}(u, w) + \text{dist}(w, v)$.

[1] In the simplest *linear model*, $\theta_u(a_{u,v}) := a_{u,v}$.

In addition to Euclidean spaces and normed vector spaces, other abstract metrics are commonly used. These include the metrics defined by shortest path distances, effective resistances, or commute times of random walks.[2] The *doubling spaces* are also popular metrics for capturing the underlying geometry of networks [171, 187]. Other geometric spaces, such as hyperbolic planes, have demonstrated remarkable routing properties [212], which are desirable for understanding information flow and network dynamics.

The affinities between nodes may then be determined by their *distances* from the underlying metric space: The closer two elements are, the higher their affinity becomes, and the more interactions they have. A standard way to define affinity weights for $u \neq v$ is:

$$w_{u,v} = \frac{1}{(\text{dist}(u,v))^{\alpha}}, \qquad \text{for some } \alpha > 0.$$

Feature Models

In this model, one assumes that there exists an underlying "feature" space, $\mathcal{F} = (V, \mathbf{F})$, that impacts the interactions among nodes in the network. This is a widely-used alternative data model for information networks. In a d-dimensional feature space, $\mathbf{F}$ is an $n \times d$ matrix, where $f_{u,i} \in \mathbb{R}^{+} \cup \{0\}$ denotes u's *quality score* with respect the i^{th} feature. Let $\mathbf{f}_u$ denote the u^{th} row of $\mathbf{F}$, i.e., the *feature vector* of node u. The affinity weights $w_{u,v}$ between two nodes u and v may then be determined by the *correlation* between their features:

$$w_{u,v} \sim \left(\mathbf{f}_u^T \cdot \mathbf{f}_v\right) = \sum_{i=1}^{d} f_{u,i} \cdot f_{v,i}.$$

2.3 Beyond Graph Models for Social/Information Networks

Real-world network data is usually much richer than its graph-theoretical representation. Network processes such as viral marketing [114, 286], social influence [201], political alliance [166, 67, 48, 333, 290,

[2] For an undirected network, the commute time between nodes is proportional to the effective resistances between them. See Chapters 6 and 7, for definition.

283], and strategic/economic cooperation [302] are more complex than what can be captured by nodes and edges. The needs to understand diverse network phenomena and to find solutions to vast network applications have led to mathematical frameworks beyond graph theory, as different network models enable us to study different facets of network data.

In computer science — without doubt — (weighted) graphs are the most widely-used model for social and information networks. In social sciences and mathematical economics, several other natural models have been used to study social interactions and the formation of coalitions/communities. Below, we give a few examples.

Preference Networks

In many applications, one often uses *ordinal models*,[3] rather than *cardinal models*, to capture the preferences/affinities among elements [33]. Two classical applications of preference frameworks are voting in social choices [33] and stable marriage [140, 289, 160].

Preference frameworks are also used in the study of coalition formation [67]. An example of the modern use of preference models is the *Border Gateway Protocol* (BGP) for network routing between autonomous Internet systems [284, 77].

In a recent axiomatic study of community identification in social networks, Borgs *et al.* [66, 39] considered the following *abstract* social/information network framework. Below, for a non-empty finite set V, let $L(V)$ denote the set of all *linear orders* on V.

[3] Weighted formulations of preferences/affinities are indeed more common in computer science and network science. However, it is well known that Arrow [33] decided to use preference rankings over cardinal expressions in his social-choice theory for modeling people's preferences and utilities, and their collective aggregation. He once declared, "Modern economic theory has insisted on the ordinal concept of utility; that is, only orderings can be observed, and therefore no measurement of utility independent of these orderings has any significance. In the field of consumer's demand theory the ordinalist position turned out to create no problems; cardinal utility had no explanatory power above and beyond ordinal. Leibniz' Principle of the Identity of the Indiscernibles demanded then the excision of cardinal utility from our thought patterns."

Definition 2.2 (Preference Networks)**.** A *preference network* is a pair $A = (V, \Pi)$, where V is a non-empty finite set and $\Pi = \{\boldsymbol{\pi}_u\}_{u \in V} \in L(V)^V$ is a *preference profile on* V. Here $\boldsymbol{\pi}_u$ specifies the total ranking V in the order of u's preference: $\forall s, u, v \in V$, *s prefers u to v*, denoted by $u \succ_{\boldsymbol{\pi}_s} v$, if and only if $\boldsymbol{\pi}_s[u] < \boldsymbol{\pi}_s[v]$.

The stable marriage problem [160] and subsequent work in coalition formation [166, 67, 48, 333, 290, 283] can be viewed as studies of matchings or clusterings, respectively, over preference networks. When working with general weighted graphs, one usually has to address the interpretation of missing data. In contrast, Borgs *et al.*, [66] used complete-information preference networks — as in Definition 2.2 — to focus on axiomatic characterizations of community-identification rules for social networks:

- *What constitutes a community in a social network of preferences?*

The complete-information preference networks also enable researchers [160, 284, 66] to focus on conceptual questions such as:

- *What clusterings are stable, according to individual preferences?*
- *What BGP routing policies are stable, according to nodes' path preferences*?

The simplicity of the preference framework makes it an attractive baseline network model for studying the fundamental questions highlighted above (see see Section 8.1 for more discussion).

Multifaceted Preference Networks

A key task for network clustering and community identification is to formulate the clusterability/coherence of *groups* of nodes based on *individual* preference/affinity data:

> *To characterize if a group $S \subseteq V$ is a good cluster/community, we need to capture how well each node $u \in S$ "interact" or "fits in" with each subset $T \subset S \setminus \{u\}$.*

The use of a single preference per node — as in Definition 2.2 — provides a strict interpretation of individual preferences and their projection onto node-group interactions. In this model, each node u uses a consistent preference ranking $\boldsymbol{\pi}_u$ to interact with any group of nodes. In contrast, in the graph-based network model, a node favors different fractions of its edges, when interacting with different subsets of nodes: It prioritizes the edges connecting with the subset.

The simplicity of Definition 2.2 comes with limitations. The preference model has a far simpler structure than the traditional graph-theoretical network model. In particular, as illustrated in Section 8.1, every preference network has only a linear number of cliques (Theorem 8.7), and these cliques form a hierarchical structure.

For practical social networks, node-group interactions are more flexible than those allowed by preference networks. However, node-group interactions might be far less flexible than those allowed in the graph-theoretical network model. This greater flexibility of node-group interactions may be one of the main challenges for formulating clustering and communities. In fact, in his axiomatic study of clustering rules, Kleinberg [207] focused on networks whose pair-wise similarities or affinities are derived from an underlying metric space. Like the preference model, Kleinberg's "geometric network model" also has a stricter interpretation of individual affinities and their projection onto node-group interactions than standard weighted graphs.

An intermediate model is the *multifaceted preference network* [39], where each node has multiple rankings over other nodes:

Definition 2.3 (Multifaceted Preference Networks)**.** A *multifaceted preference network* is a triple $A = (V, \mathbf{d}, \Pi)$, where V is a non-empty finite set, $\mathbf{d} : V \to \mathbb{Z}^+$ is an integer vector. For $u \in V$, d_u specifies the *dimension* of u's preferences, and Π is a *preference profile on* V: $\Pi = \{\boldsymbol{\Pi}_u : u \in V\}$, where $\boldsymbol{\Pi}_u \in L(V)^{d_u}$ specifies the d_u different rankings of V associated with node $u \in V$.

This definition reflects that in a social network of preferences, some members may have multiple preference rankings, based on various cri-

teria. For example, some members have two preference rankings, based on (1) love of books and (2) passion for sports, while others have three, based on (1) family/friends, (2) academic interests, and (3) business interests.

We say $A = (V, \mathbf{d}, \Pi)$ is *d-regular*, if $d_u = d$, $\forall u \in V$. Note that a d-regular multifaceted preference network is less restrictive than the simple union of d standard preference networks. In other words, in a multifaceted preference network, there is no *global* correlation among preferences. In Section 8.1, we will discuss (multifaceted) preference networks in the context of community identification.

Game-Theoretical Network Models

Game theory, mechanism design, and mathematical economics also provide natural frameworks for modeling social and information networks [175, 109, 192, 85, 210]. Indeed, incentive measure is the most efficacious means for measuring network interactions. For example, the following incentive model explicitly captures the utilities of node-group interactions in a network.

Definition 2.4 (Incentive Networks). An *incentive network* is a pair $U = (V, \boldsymbol{u})$, where V is a non-empty finite set and for all $s \in V$, $u_s : 2^{V \setminus \{s\}} \to \mathbb{R}$ is a function that defines s's *incentive utility* over subsets of $V \setminus \{s\}$.

There are $|S|$ utility values, $(u_s(S \setminus \{s\}) : s \in S)$, associated with each group $S \subseteq V$ in the incentive network U. In other words, for each $s \in S$, the *value* of its interaction with the rest of the group $S \setminus \{s\}$ is explicitly defined as $u_s(S \setminus \{s\})$. These utility measures could be useful for formulating the group's clusterability and coherence. Compared with the preference and geometric network models, the incentive network is a more explicit mathematical model for studying conceptual questions in clustering and community identification.

The incentive network of Definition 2.4 is similar to the model of classical cooperative game theory [300, 302], which characterizes the *cooperative utility* $v(S)$ of each group $S \subseteq V$: The value $v(S)$ denotes

the utility of the group when all its members agree to cooperate with each other. However, these two models have a subtle difference. The incentive network characterizes the (*asymmetric*) *individual utility* of each member in its interaction or cooperation with the rest of the group. In contrast, the cooperative-game-theoretical model characterizes the *symmetric* utility of every group [300, 302].

To highlight this difference, we say that an incentive network is *symmetric*, if $u_s(S \setminus \{s\}) = u_t(S \setminus \{t\})$, $\forall S \subseteq V$ and $\forall s, t \in S$. In this case, each set S has a *cooperation value*, $v(S)$, that is equal to $u_s(S \setminus \{s\})$, for any $s \in S$. We call the weighted hypergraph model, $(V, \{v(S)\}_{S \subseteq V})$, a *cooperative network* [331].

The incentive network is a complete-information model. Like many formulations in game theory, mathematical economics, and social sciences, the complete-information descriptions of node-group interactions enable researchers to focus on conceptual studies of relevant solution concepts. Like utility functions in mechanism design, these node-group utility functions may also enjoy rich mathematical properties [222], such as *submodularity*[4] [262], *complementarity* [130], or their combinations [128]. The incentive network, as given in Definition 2.4, is a highly theoretical model. The dimensionality of its data is exponential in n, as it is defined by $n2^{n-1}$ utility values.

For practical network applications, node-group interaction utilities should have succinct representations, either with a graphical [193] or hyper-graphical model [130, 128], or with low dimensionality [4]. For example, given a weighted network $G = (V, E, \mathbf{W})$ or a trust-based recommendation system [19] $G = (V, E, \mathbf{W})$ with $\mathbf{W} \in \{-1, 0, 1\}^{n \times n}$, Deng and Papadimitriou [112] defined a cooperative-game-theoretical network, whose group value is $v(S) = \sum_{s,t \in S} w_{s,t}$. One can similarly define an asymmetric incentive network, with node-group interaction utility $u_s(S \setminus \{s\}) = \sum_{t \in S \setminus \{s\}} w_{s,t}$.

In a recent study of the interplay between social influence and network centrality, Chen and Teng [84] used social-influence models [114, 286, 201] to formulate more sophisticated (symmetric) incen-

[4] A subset function $f : 2^V \to \mathbb{R}$ is *submodular* if the following is true: $\forall S, T \subseteq V$, $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$.

tive networks: The cooperative utility of a group $S \subseteq V$ is its *influence spread*, i.e., the expected number of nodes that can be influenced when S is the seed set. Other "static sphere-of-influences" can also be used to define symmetric incentive networks based on a given graph [322, 257, 249, 1, 323]. These incentive networks have succinct representations. We will further discuss them in Section 8.1 in the context of network centrality and community identification.

2.4 Basic Problems in Data and Network Analysis

The emerging need to understand real-world data and networks has given rise to many fundamental conceptual and algorithmic problems. Below, we discuss a few examples concerning social and information networks.

Significant Nodes: Ranking and Centrality

Identifying nodes of relevance and significance is an important problem in network analysis. For example, in Web search [73, 211] and viral marketing [114, 286, 201], one needs answers to the following questions:

> *Which nodes are the most significant nodes in a network or a sub-network? How quickly can we identify them?*

The former question involves conceptual formulation of the *significance* — in terms of *centrality or popularity* — of nodes in a network. One can either measure the significances of nodes numerically [73, 267, 211], or characterize their significances by ranking them [15].

- For weighted networks, for instance, *PageRank* [73] was introduced and widely used as a measure of centrality. Recall that the PageRank of a node is its stationary probability in an underlying Markov process defined over the network (see Chapter 3 for definition).
- To rank nodes in a preference network $A = (V, \Pi)$, we can apply the social-choice-theoretical approach to aggregate the preferences in Π into a collective-ranking of the nodes [33].

This social-choice connection to network centrality is particularly relevant, because Arrow's celebrated impossibility theorem on voting [33] highlights the conceptual challenge in centrality formulation. Network centrality is a form of "dimensionality reduction" from "high dimensional" network data to "low dimensional" centrality measures or rankings. In this process, some information will be lost regardless of the scheme for centrality formulation. Arrow's impossibility theorem reveals the fundamental challenge in keeping information loss in a consistent manner. It shows that for various (desirable) properties, *conforming* centrality scheme may not even exist. We will have more discussion about centrality formulation in Section 8.1.

The latter question requires algorithmic studies. As networks evolve and become increasingly vast, it is imperative to develop algorithms that can efficiently identify significant nodes, without actually examining the entire input network. This is particularly important for Web search, where the search engine can only effectively index a fraction of the webpages. In Chapter 4, we will discuss a class of *local network algorithms* [318, 22, 21]. The combination of efficient local algorithms and advanced network sampling methods — see Chapter 3 — provides a promising framework for these challenging computational tasks [65].

Coherent Groups: Clustering and Communities

Identifying groups with significant structural properties is another basic problem in network analysis. These groups may represent communities of people, political/economic coalitions in social networks, or categories of documents in information networks. In this context, one may ask:

- *Which groups should be identified as coherent communities?*
- *What are the significant clusters in a data set?*
- *What constitutes a community in a preference, cooperative, or incentive network?*
- *How fast can we identify one, uniformly sample one, or enumerate all significant groups?*

The first two questions involve the conceptual formulation of *group coherence* in networks or data sets. In the literature, a diverse set of formulations of group coherence has been proposed. The following are some examples:

- ***Numerical Measures***: Quantities based on intuitions from physics or statistics — such as conductance, modularity, density, or compressibility — are widely used to formulate clusterability measures, and to characterize network communities and data clusters. Optimization objectives — such as for graph partitioning, geometric clustering, and data compression — are also used to identify coherent substructures in networks and data sets.

- ***Structural Characterization***: Existing graph-theoretical concepts, such as k-cores,[5] dense subgraphs (cliques), and expanders, are used as basic group characterization. Researchers have also introduced new graph-theoretical concepts, such as (α, β)-communities [254] and hidden communities [169].

- ***Social Choice Formulation***: Developing a group characterization based on data local to individual nodes — as given in social networks — is an essential task for community identification. This view has led researchers to examine various (preference) aggregation rules — from social-choice theory [39, 66] — to model the formation of network communities.

One may also need to address other structural questions such as:

- *Shall we use a clustering that must (hierarchically) partition the nodes, or shall we consider overlapping communities?*

- *Shall we remove outliers before constructing the clustering?*

One may also take a step back — as Arrow did when studying voting and social choices in the 1950s [32] — to understand some meta-questions regarding clustering and community identification schemes:

[5] For an integer $k > 0$, a non-empty set $S \subseteq V$ is a *k-core* of a graph $G = (V, E)$ if for every vertex $u \in S$, its degree in S, $d_u^{(S)} = |\{v \in S : (u, v) \in E\}|$ is at least k. For example, in the graph defined by "Facebook friends", a k-core is a group of people where every one has at least k friends in the group.

- *How should we evaluate the consistency of a clustering or community-identification scheme?*
- *What desirable properties should clustering or community-identification schemes satisfy?*

For example, focusing on data points in metric spaces, Kleinberg [207] provided a holistic analysis of (non-overlapping) clustering. His celebrated impossibility theorem illustrates the fundamental challenge in defining a consistent clustering scheme. In Section 8.1, we will survey some recent axiomatic studies of community-identification rules in preference and game-theoretical network models.

Group-coherence formulations also have algorithmic implications. For example, there is a simple scalable algorithm for determining if a graph $G = (V, E)$ has a k-core.[6] In contrast, it is NP-hard to detect the presence of cliques or dense subgraphs of certain size [70].

In Chapter 4, we will survey a family of local clustering algorithms, which use techniques from spectral graph theory [318, 22, 93, 24, 23]. These algorithms are both efficient and provably good. They guarantee that the conductance of the clusters they produce satisfy the target quality.

Local clustering algorithms only use the local structure of networks — typically near an input node — and usually output quality clusters without exploring entire networks. Thus, they are desirable for practical implementation [348].

The design of efficient and provably-good local algorithms for broader classes of graph-theoretical problems will continue to be an important area of network and data analysis.

In Chapter 5, we will survey a family of scalable geometric methods for partitioning graphs that arise from geometrically embedded data. We will also illustrate the intrinsic connection between spectral partitioning and geometric partitioning methods.

[6] Repeatedly delete nodes whose current degree is less than k, and their incident edges, from G, until either all remaining nodes have degree at least k, or there are no more nodes left. None of the deleted nodes can be a member of any k-core. Thus, if the algorithm terminates with a non-empty graph, then their nodes form the largest k-core of G.

Interplay between Networks and Dynamic Processes

The conceptual challenge in modeling centrality and group coherence has another dimension. A given social network can be part of different dynamic processes, such as epidemic spreading, random-walk diffusion, viral marketing, and ideas/innovation propagation. These processes can potentially affect the relations between nodes of the network, which in turn influence the formation of clusters or communities, as well as impact significances of nodes. Thus, understanding this interplay between dynamic processes and their underlying networks is a fundamental task in network science [146, 84]. In this area of research, we have to address the following basic questions:

- *How should we model the interaction between network nodes in a given dynamic process?*
- *How should we characterize node significance and group coherence with respect to a dynamic process?*
- *How fast can we identify influential nodes and significant communities?*

We will consider these fundamental and largely open questions in Chapters 4 and 8. In the former chapter, we will review a spectral-graph-theoretical framework proposed by Ghosh *et al.* [146] for studying how diffusion models affect clusterability. In the latter chapter, we will discuss a recent game-theoretical approach for characterizing how social influence processes impact network centrality [84].

Social influence [201] and viral marketing [114, 286] are wonderful exemples of network dynamics. They also help to highlight the richness and multifacetedness of real-world network data. While some network processes, such as standard random-walks, can be sufficiently captured from the graph-theoretical facet of network data, many phenomena and applications require us to have a holistic understanding of network data. In social influence, for instance, the underlying interactions are not pairwise, but groupwise. Thus, they can be more naturally captured by the cooperative network model [201, 84], or by a probabilistic model

with exponential dimensionality that specifies the pairwise influence between each pair of groups under the influence process [84]. Although it was not stated explicitly, the framework of Kempe, Kleinberg, and Tardos [201] can be viewed as one that studies influence maximization under a succinctly-defined cooperative network, in which each group's utility is its influence spread (Eqn. 8.1). The work of [84] provides a multi-model approach for analyzing the impact of social-influence processes on network centrality. Its axiomatic framework captures the two-step "game-theoretical" dimension reduction from the probabilistic influence model between groups to the utility model over groups, and then to the centrality formulation over nodes. We believe that multi-model approaches to network analysis will become increasingly more essential for studying major subjects in network science.

Multiple Networks: Composition and Similarity

In addition to studying individual networks, another important task in network science is to understand multiple networks. One fundamental problem is *network composition* [228]. In the real-world, people usually have access to multiple social networks, with combinations of traditional physical interactions (meetings and collaborations) and modern digital interactions (email and social media). In other words, people are participating in a *multi-layer social network.*

- Can we formulate an integrated social network that captures a multi-layer social network?

- How should we measure centrality, clusterability, and network influence, in a multilayer social network?

Another fundamental problem is *network similarity*:

- *How should we measure the similarity between two networks* $G = (V, E, \mathbf{W})$ *and* $\bar{G} = (V, \bar{E}, \bar{\mathbf{W}})$*, over the same set of nodes?*

- *Given a network* $G = (V, E, \mathbf{W})$*, can we construct a* sparser *network* $\tilde{G} = (V, \tilde{E}, \tilde{\mathbf{W}})$ *that approximates* G*?*

We will survey the concept of *spectral similarity* of graphs [317, 43] and its algorithmic impact on network sparsification (in Chapter 6) and on scalable algorithm design (in Chapter 7).

2.5 Sparse Networks and Sparse Matrices

> *In the real-world, big networks are usually and fundamentally sparse.*

In vast applications, networks are intrinsically sparse. For example, social networks are sparse because people have limited attention spans for social interaction. Protein networks are sparse because the three-dimensional space has physical constraints. In other applications, ideal networks are dense. But, because of economic, physical, computational constraints, sparse networks are used in place of ideal ones. These sparse networks usually are "merely" good enough to "support" or approximate their ideal counterparts. For example, although the underlying communication are between all pairs of nodes, sparse networks — such as hypercubes and hierarchical trees — are used as the main architectural infrastructures in parallel systems or communication networks. Similarly, sparse discretizations are widely used in the finite-element simulations (of earthquakes or combustions) [320, 170], and N-body simulations of (particles or stars) [158, 327]. The sparse meshes are used to approximate the underlying physical interactions that are dense or continuous.

Therefore, for data and network analysis, it is important to understand both dense and sparse networks. It is even more important to understand the underlying relationship between them. For instance:

- N-Body simulation interaction graphs are the numerical approximation of the underlying dense physical interactions between particles or stars [158, 327].
- Sparse finite-element meshes are the geometric approximation of the underlying continuous domains [320].
- Spectral sparsifiers are the combinatorial, statistical, and algebraic approximation of dense networks [317].

These sparse graphs are generated by a dense underlying interaction model to meet various computational and approximation requirements.

In the same spirit — as argued in [39] — a real-life social and information network may be viewed as a sparse observed network, induced by a dense underlying preference, affinity, statistical, or incentive network. Thus, for mathematical studies of network phenomena, it might be beneficial to consider the underlying dense networks, in addition to working directly with sparse observed networks. We will come back to this topic in Section 8.1, with focus on the formation of network communities [66] and clusterings [207].

Algorithmically, we need scalable methods to work directly with sparse networks and sparse matrices. As these networks become massive, $O(n^2)$ time is prohibitively expensive. Thus, in any task for identifying structures whose formulations are based on underlying dense interaction models, we also need scalable algorithms to derive a sparse approximation of the dense model. The design of scalable algorithms for sparse networks and for network sparsification will be a central theme of this article, as highlighted in Chapters 6 and 7.

In conclusion, we will briefly review some basic concepts for representing and manipulating sparse networks and their sparse matrices. We will use the notation for *sparse vectors/matrices* introduced by Gilbert, Moler, and Schreiber [149] for Matlab.

Suppose $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a matrix. Let Sparse($\mathbf{A}$) denote the *sparse form* of matrix $\mathbf{A}$ by "squeezing out" any zero elements in $\mathbf{A}$. Let support($\mathbf{A}$) be the set of all indices whose entries in $\mathbf{A}$ are nonzeros, and let nnz($\mathbf{A}$) := |support($\mathbf{A}$)|. Conceptually, one can view Sparse($\mathbf{A}$) as a list of nnz($\mathbf{A}$) index-entry pairs, one for each element of support($\mathbf{A}$). We will also adopt the following notations: Let Sparse([ ]) denote the vector or matrix with all-zeros in the sparse form. For any $i, j \in \mathbb{Z}^+$ and $b \in \mathbb{R} \setminus \{0\}$, let Sparse$(i, b)$ denote the sparse vector with one nonzero element of value b located in the i^{th} place in the vector, and let Sparse$((i, j), b)$ denote the sparse matrix with one nonzero element of value b located at entry (i, j).

Throughout the rest of the article, we shall assume that all vectors and matrices are expressed in the sparse form. For programming de-

velopments, a sparse vector can be implemented using a binary search tree or a hash table. For some of our algorithmic presentations, we will explicitly or implicitly use the following properties of sparse matrices/vectors.

Proposition 2.5 (Sparse Matrix/Vector Operations)**.** (1) For any $m \times n$ matrix $\mathbf{A}$, the space complexity of Sparse($\mathbf{A}$) is $\widetilde{O}_{m+n}(1) \cdot \text{nnz}(\mathbf{A})$. (2) For any $i \in [m]$ and $j \in [n]$, it takes $\widetilde{O}_{m+n}(1)$ time to obtain $a_{i,j}$. (3) For any vector $\mathbf{b} \in \mathbb{R}^n$, it takes $\widetilde{O}_{m+n}(1) \cdot \text{nnz}(\mathbf{A})$ to compute the matrix-vector product $\mathbf{A} \cdot \mathbf{b}$. (4) For $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, $\mathbf{a} + \mathbf{b}$ can be implemented in time $O(\min(\text{nnz}(\mathbf{a}), \text{nnz}(\mathbf{b})) \cdot \log n)$.

Proof. Statements (1), (2) and (3) are straightforward. We now prove the last statement. Each sparse vector can be stored as a binary search tree, where the key for a non-zero entry is the index of the entry. Suppose $\text{nnz}(\mathbf{b}) \leq \text{nnz}(\mathbf{a})$. We can then update the binary search tree of $\mathbf{a}$ by inserting the elements of $\mathbf{b}$ into it to form a sparse representation of $\mathbf{a} + \mathbf{b}$. The statement follows from the fact that each such insertion operation can be implemented with $O(\log n)$ operations. □

3

Significant Nodes: Sampling - Making Data Smaller

In this chapter, we will use *PageRank* — a popular notion of network centrality [73, 266] — to illustrate the alg orithmic challenges and advances in network analysis. Suppose $G = (V, E, \mathbf{W})$ is a weighted directed graph. The *PageRank centrality* uses an additional parameter $\alpha \in (0, 1)$ — known as the *restart constant* — to define a Markov process whose transition rule, for any node $V \in V$, is the following:

- with probability α, restart at a random node in V, and
- with probability $(1 - \alpha)$, move to a random neighbor of v, chosen with probability proportional to the weight of the edge connecting v to that neighbor.

Then, the *PageRank centrality* of any $v \in V$ (with restart constant α) is proportional to v's stationary probability in this Markov process.

Recall that a *Markov process* over $V = [n]$ is defined by an $n \times n$ transition matrix $\mathbf{M}$, which satisfies the *stochastic condition*:

$$\mathbf{M} \cdot \mathbf{1} = \mathbf{1}.$$

A probability vector $\boldsymbol{\pi}$ is the *stationary distribution* of this Markov process if:

$$\mathbf{M}^T \boldsymbol{\pi} = \boldsymbol{\pi}.$$

Every ergodic Markov process has a stationary distribution. $\mathbf{M}$ defines a *detailed-balanced* Markov process, if for all $u, v \in V$, $\pi_u m_{u,v} = \pi_v m_{v,u}$. The following fact is well known [9]:

Proposition 3.1 (Detailed-Balanced)**.** Suppose $\mathbf{M}$ is a transition matrix with stationary distribution $\boldsymbol{\pi}$. Let $\boldsymbol{\Pi}$ be the diagonal matrix defined by $\boldsymbol{\pi}$. Then, $\boldsymbol{\Pi} \cdot \mathbf{M}$ is symmetric if and only if the Markov process defined by $\mathbf{M}$ is detailed-balanced.

Let $d_u^{out} = \sum_{v \in V} w_{u,v}$ and $d_u^{in} = \sum_{v \in V} w_{v,u}$, respectively, denote the *out-degree* and *in-degree* of $u \in V$ in G. Let $d_u = d_u^{out} + d_u^{in}$.

Mathematically, we normalize the PageRank vector so that the sum of its values over all nodes is equal to n. Then, the *PageRank vector* of G is the solution to the following *Markov random-walk* equation [266, 168]:

$$\mathbf{PageRank}_{\mathbf{W},\alpha} = \alpha \cdot \mathbf{1} + (1-\alpha) \cdot \mathbf{W}^T \left(\mathbf{D}_{\mathbf{W}}^{out}\right)^{-1} \mathbf{PageRank}_{\mathbf{W},\alpha} \quad (3.1)$$

In this equation, $\mathbf{D}_{\mathbf{W}}^{out} = \mathrm{diag}([d_1^{out}, ..., d_n^{out}])$ is the diagonal matrix of out degrees, and recall that $\mathbf{1}$ denotes the n-place vector of all 1s. It is well known that $\mathbf{PageRank}_{\mathbf{W}}$ is non-negative for any G [266]. In addition, the normalization ensures that:

$$\sum_{u \in V} \mathbf{PageRank}_{\mathbf{W},\alpha}(u) = n \quad (3.2)$$

We will consider a fundamental problem in network analysis:

Problem 3.2 (Significant PageRank)**.** Given a network $G = (V, E, \mathbf{W})$, a restart constant $\alpha \in (0, 1)$, and a threshold value $\Delta > 0$, identify all nodes whose PageRank values are at least Δ.

This basic problem illustrates a powerful *multiscale sampling framework* for designing super-scalable algorithms. We first note that the output of Significant PageRank has at most n/Δ nodes. We also note that in most practical applications, such as Web search, one only

cares about nodes with sufficiently large PageRank, i.e., $\Delta \gg 1$. Thus, in these applications, the number of significant nodes is much smaller than n. For example, when $\Delta = n^{\frac{1}{5}}$, there are at most $n^{\frac{4}{5}}$ nodes whose PageRank values are larger than Δ, and $n^{\frac{4}{5}}$ is sub-linear in n. These applications inspire the following basic algorithmic question:

> *Can we identify nodes with significant PageRank values without needing to explore entire input networks?*

We will now discuss an extremely simple, super-scalable sampling algorithm that solves following variant of Significant PageRank [65].

Problem 3.3 (Significant-PageRank Identification)**.**
Input: (G, α, Δ, c) where:

- $G = (V, E, \mathbf{W})$ defines a (directed) network, and $0 < \alpha < 1$
- $1 \leq \Delta \leq |V|$ specifies a threshold PageRank value
- $c > 1$ defines the approximation accuracy.

Output: A subset $S \subseteq V$ such that S contains all nodes of PageRank values at least Δ and no node with PageRank values less than $\frac{\Delta}{c}$.

Proposition 3.4 (Sparse Output)**.** For any $G = (V, E, \mathbf{W})$ of n vertices, constant $\alpha \in (0, 1)$, $\Delta \in [n]$, and $c > 1$, if S is a valid output for Significant-PageRank Identification then:

$$|S| \leq c \cdot \frac{n}{\Delta}$$

Proof. The statement is true because (1) every node in the output has PageRank at least $\frac{\Delta}{c}$, and (2) the total PageRank values is n. □

Borgs *et al.*'s algorithm [65] has complexity $\widetilde{O}_n(1) \cdot \frac{n}{\Delta}$ — which is optimal up to logarithmic factors — and usually runs in sub-linear time in n.

3.1 Personalized PageRank Matrix

In order to obtain a super-scalable algorithm for SIGNIFICANT-PAGERANK IDENTIFICATION, it is beneficial to first take a step back, and examine a larger matrix structure than the original input network. We will show that this matrix, together with two effective sampling schemes, provides an efficient solution to this fundamental problem in network analysis.

This matrix view of PageRank offers a local structure, which is essential for efficient computation and sampling. The local concept is personalized PageRank, introduced by Haveliwala [168]. The *personalized PageRank* $\mathbf{p}_u$ of $u \in V$ is given by the following equation:

$$\mathbf{p}_u = \alpha \cdot \mathbf{1}_u + (1-\alpha) \cdot \mathbf{W}^T \cdot \left(\mathbf{D}_{\mathbf{W}}^{out}\right)^{-1} \cdot \mathbf{p}_u \tag{3.3}$$

where $\mathbf{1}_u$ is the n-place vector whose u^{th} location is 1; all other entries in $\mathbf{1}_u$ are zeros. As $\mathbf{p}_u$ is the stationary distribution of the random walk that restarts at u with probability α, we can explicitly express $\mathbf{p}_u$ as:

$$\mathbf{p}_u = \alpha \sum_{k=0}^{\infty} (1-\alpha)^k \cdot \left(\mathbf{W}^T \cdot \left(\mathbf{D}_{\mathbf{W}}^{out}\right)^{-1}\right)^k \cdot \mathbf{1}_u. \tag{3.4}$$

Personalized PageRank is asymmetric, and hence to emphasize this fact, we express $\mathbf{p}_u$ as:

$$\mathbf{p}_u = (p_{u\to 1}, \cdots, p_{u\to n})^T$$

Together, $\{\mathbf{p}_u\}_{u\in V}$ define the following matrix:

Definition 3.5 (Personalized PageRank Matrix)**.** The *personalized PageRank matrix* of n-node network $G = (V, E, \mathbf{W})$ and restart constant $\alpha > 0$ is:

$$\mathbf{PPR}_{\mathbf{W},\alpha} = [\mathbf{p}_1, ..., \mathbf{p}_n]^T = \begin{bmatrix} p_{1\to 1} & \cdots & p_{1\to n} \\ \vdots & \ldots & \vdots \\ p_{n\to 1} & \cdots & p_{n\to n} \end{bmatrix} \tag{3.5}$$

For any matrix $\mathbf{A}$, let $\text{columnSum}(\mathbf{A}) = \mathbf{1} \cdot \mathbf{A}$ denote the vector whose k^{th} entry is equal to the sum of $\mathbf{A}$'s k^{th} column. Then:

Proposition 3.6 (Personalized Contributions to PageRank)**.** For any $G = (V, E, \mathbf{W})$ and restart constant $\alpha > 0$:

$$\mathbf{PageRank}^T_{\mathbf{W},\alpha} = \text{columnSum}(\mathbf{PPR}_{\mathbf{W},\alpha}) = \mathbf{1}^T \cdot \mathbf{PPR}_{\mathbf{W},\alpha} \tag{3.6}$$

Like PageRank, personalized PageRank matrix applies to both directed and undirected networks. It has several elegant properties.

Theorem 3.7 (PageRank Markov Process)**.** For any weighted directed network $G = (V, E, \mathbf{W})$ and restart constant $\alpha > 0$:

1. $\mathbf{PPR}_{\mathbf{W},\alpha}$ is a stochastic matrix, i.e., $\mathbf{PPR}_{\mathbf{W},\alpha} \cdot \mathbf{1} = \mathbf{1}$. Thus, $\mathbf{PPR}_{\mathbf{W},\alpha}$ defines its own Markov process.
2. $\mathbf{PPR}_{\mathbf{W},\alpha}$ and $\left(\mathbf{D}^{out}_{\mathbf{W}}\right)^{-1} \cdot \mathbf{W}$ have the same eigenvectors. Thus, they have the same stationary distribution.
3. $\mathbf{PPR}_{\mathbf{W},\alpha}$ is detailed-balanced if and only if $\mathbf{W}$ is symmetric. Let $\mathbf{D}_{\mathbf{W}}$ be the diagonal matrix of $\mathbf{d} = \mathbf{W} \cdot \mathbf{1}$. Then, $\mathbf{D}_{\mathbf{W}} \cdot \mathbf{PPR}_{\mathbf{W},\alpha}$ is a symmetric matrix.

Proof. By Eqn: (3.4), we can explicitly express $\mathbf{PPR}_{\mathbf{W},\alpha}$ as:

$$\mathbf{PPR}_{\mathbf{W},\alpha} = \alpha \sum_{k=0}^{\infty} (1-\alpha)^k \cdot \left(\left(\mathbf{D}^{out}_{\mathbf{W}}\right)^{-1} \cdot \mathbf{W}\right)^k. \tag{3.7}$$

Note that:

$$\alpha \sum_{k=0}^{\infty} (1-\alpha)^k = 1.$$

Thus, $\mathbf{PPR}_{\mathbf{W},\alpha}$ is a convex combination of (multi-step) random-walk matrices defined by $\left(\mathbf{D}^{out}_{\mathbf{W}}\right)^{-1} \cdot \mathbf{W}$. Statements (1) and (2) follow directly from the fact that $\left(\left(\mathbf{D}^{out}_{\mathbf{W}}\right)^{-1} \cdot \mathbf{W}\right)^k$ is a stochastic matrix for any integer $k \geq 0$.

When $\mathbf{W}$ is symmetrix, Eqn. (3.7) becomes:

$$\mathbf{PPR}_{\mathbf{W},\alpha} = \alpha \sum_{k=0}^{\infty} (1-\alpha)^k \cdot \left(\mathbf{D}_{\mathbf{W}}^{-1} \cdot \mathbf{W}\right)^k \tag{3.8}$$

Thus, $\mathbf{D}_{\mathbf{W}} \cdot \mathbf{PPR}_{\mathbf{W},\alpha}$ is a symmetric matrix. The stationary distribution of $\mathbf{D}_{\mathbf{W}}^{-1}\mathbf{W}$ — and hence of $\mathbf{PPR}_{\mathbf{W},\alpha}$ — is proportional to $\mathbf{d} = \mathbf{W}\cdot\mathbf{1}$. Thus, by Proposition 3.1, $\mathbf{PPR}_{\mathbf{W},\alpha}$ is detailed balanced. □

The *PageRank contribution* of u to v is $p_{u\to v}$ [21]. The vector $\mathbf{p}_u$ can also be viewed as u's distribution of affinities to all nodes in the network. Let's call $p_{u\to v}$ the *PageRank affinity* of node u to v.

Corollary 3.8 (PageRank Affinity)**.** For any restart constant $\alpha > 0$, $\mathbf{PPR}_{\mathbf{W},\alpha}$ is a normalized affinity network — as defined in Definition 2.1 — derived from a weighted network $G = (V, E, \mathbf{W})$.

3.2 Multi-Precision Annealing for Significant PageRank

Because $\mathbf{PPR}_{\mathbf{W},\alpha}$ has n^2 entries, on the surface, Proposition 3.6 is not useful for developing scalable algorithms for computing or estimating PageRank vector. The key idea of Borgs *et al.* [65] is to characterize PageRank without calculating all entries in $\mathbf{PPR}_{\mathbf{W},\alpha}$. The efficiency of their algorithm comes from combining two basic methodologies for designing scalable algorithms:

- *Local algorithms for exploring networks* [318, 22, 21]
- *Provably good sampling for robust estimates* [65]

Schematically, the approach of [65] follows the simulated annealing framework [248, 205]. Recall that the traditional annealing methods uses a — "high temperature" — top-level randomization process to formulate multiple local searches, aiming at finding a good approximation of the global optimum (for an optimization problem). By contrast, the PageRank annealing algorithm aims at building a robust estimator

of the PageRank vector. It also conducts a two-level network search. Its top-level mechanism emulates the "high temperature" process by sampling rows of $\mathbf{PPR}_{\mathbf{W},\alpha}$. For each row selected by the top-level process, a local network exploration algorithm is applied to gather statistical information in order to approximate that row of $\mathbf{PPR}_{\mathbf{W},\alpha}$. Robust statistical estimators are built simultaneously for all PageRank entries. In this Chapter, we will focus on the top-level sampling process. We will discuss models and algorithms for local network accesses in the next Chapter. This two-level approach employs the following property:

> *Although PageRank values are "global" statistical quantities, they can be formulated as a summarization problem of personalized PageRanks. Crucially, personalized PageRank can be approximated by local computation.*

This multi-precision annealing approach faces two challenges in order to identify all nodes with significant PageRank values. First, it does not know *a priori* which nodes have significant PageRank values. Second, the local personalized-PageRank-approximation algorithm takes more time to obtain more accurate local statistics. Thus, the multi-precision annealing algorithm must carefully manage the overall cost for gathering local statistics, while ensuring the robustness of global estimators. Its top-level process must strategically select precision levels for the local personalized-PageRank algorithm.

In Section 3.3, we will state the cost for the local approximation of personalized PageRank in terms of the approximation accuracy. Under this cost model, the annealing approach introduces a basic and beautiful data summarization problem. In Section 3.4, we will show how to build a robust estimator for this data summarization problem using multi-precision sampling. In Section 3.5, we will solve Significant-PageRank Identification using this robust estimator.

3.3 Local Approximation of Personalized PageRank

First note that every personalized PageRank vector has a sparse approximate representation.

Proposition 3.9 (Sparsity of Approximate Distribution)**.** For any directed network $G = (V, E, \mathbf{W})$ and precision parameter $\epsilon > 0$, for any $u \in V$, there are at most

$$\frac{1}{\epsilon}$$

entries in the personalized PageRank vector $\mathbf{p}_u$ whose values are more than ϵ.

We will discuss two basic approaches for approximating personalized PageRank. When given a node $u \in V$, both methods analyze the stochastic process that underlies the personalized PageRank vector $\mathbf{p}_u$. The former directly approximates the stationary distribution of this process by carefully handling small entries. The latter uses a collection of short-range random walks to sample from this stationary distribution.

Both approaches can be implemented by *local algorithms*, which conduct locally expandable searches when given a starting node $u \in V$. Both aim at identifying all nodes whose entries in $\mathbf{p}_u$ are at least ϵ, by excluding many nodes with smaller entries. As part of the sparse representation of an approximate $\mathbf{p}_u$, the entries of the unexplored nodes will be set to 0.

The goal of these approaches is to build a sparse representation of an approximate $\mathbf{p}_u$ that has only $\widetilde{O}_n(1) \cdot \frac{1}{\epsilon}$ non-zeros, and build it fast in time:

$$\widetilde{O}_n(1) \cdot \frac{1}{\epsilon}.$$

Thus, the complexity of these local personalized PageRank algorithms optimally depends upon the approximation parameter, but only mildly depends upon the size of the network.

Mathematically, such an approximation approach is possible because personalized PageRank is largely a *local* structure, i.e., its distribution is sufficiently concentrated "near" the starting node.

Personalized PageRank Approximation is an example of local algorithms [318], which will be the theme of the next chapter. Thus, we defer the technical discussion of these two algorithms to Section

4.4. Here, we only summarize the results that are needed for solving Significant-PageRank Identification.

Theorem 3.10 (Local Personalized PageRank Approximation)**.** Suppose $G = (V, E, \mathbf{W})$ is a weighted directed network with n nodes, $\alpha \in (0, 1)$ and $\epsilon \in (0, 1)$. Let $\mathbf{p}_u = \mathbf{PPR}_{\mathbf{W},\alpha}[u, :]$.

1. Andersen-Chung-Lang [22]: One can compute in time $O(d_{\max}/\epsilon)$ a sparse vector $\widetilde{\mathbf{p}}_u$ with $O(\frac{1}{\epsilon})$ non-zeros, such that:

$$\mathbf{p}_u[v] - \epsilon \leq \widetilde{\mathbf{p}}_u[v] \leq \mathbf{p}_u[v], \quad \forall v \in V$$

   where $d_{\max}$ is the largest vertex degree of graph (V, E).

2. Borgs-Brautbar-Chayes-Teng [65]: For any $0 < \rho < 1$, one can compute in time $\widetilde{O}_n(1) \cdot \frac{1}{\epsilon} \cdot \frac{1}{\rho^2}$ a sparse vector $\widetilde{\mathbf{p}}_u$ with $\widetilde{O}_n(1) \cdot \frac{1}{\epsilon}$ non-zeros such that with high probability:

$$(1 - \rho) \cdot \mathbf{p}_u[v] - \epsilon \leq \widetilde{\mathbf{p}}_u[v] \leq (1 + \rho) \cdot \mathbf{p}_u[v] + \epsilon, \quad \forall v \in V.$$

The first algorithm [22] is deterministic. It is faster for unweighted networks where all nodes have low out-degrees. The second algorithm [65] is a randomized *Monte Carlo* algorithm, which is more effective for weighted networks in which some nodes have large out-degrees. While the former has only additive errors, the latter has both multiplicative and additive errors. In the rest of this chapter, we will focus on the top-level process of the multi-precision PageRank annealing algorithm.

3.4 Multi-Precision Sampling

Theorem 3.10 offers a key subroutine — a tool for *approximately accessing* the rows of the PageRank matrix. We will now show how to use this tool to design a super-scalable algorithm for Significant-PageRank Identification.

The High-Level Idea of Multi-Precision Sampling

In algorithm design and statistical analysis, sampling is the most powerful technique available to reduce problem size. Carefully chosen or even randomly chosen samples can be used to build robust estimators for critical mathematical quantities associated with input data and networks. Sampling is also often necessary for achieving super-scalability, as such algorithms cannot process entire inputs.

At a high level, Borgs *et al.*'s algorithm [65] approximates Eqn, (3.6) (Proposition 3.6) by sampling. In order to build a sparse estimator of the PageRank vector, it applies the local personalized PageRank algorithm of Theorem 3.10 to a collection of randomly chosen rows of $\mathbf{PPR}_{\mathbf{W},\alpha}$. However, there is a trade-off between the desired precision ϵ and the running time $\widetilde{O}_n(1) \cdot \frac{1}{\epsilon}$ for accessing personalized PageRank vectors. In this application, the trade-off between precision and running time introduces the following compelling decision for sampling:

> *Shall we take more samples with low precision or fewer samples with high precision?*

In multi-precision sampling, we will translate this question into the following related question of data summarization:

> *How many samples shall we take at each precision level in order to build a robust estimator for the column sum of a matrix?*

A Numerical Model for Data Summarization

To better understand these questions, we will now consider a simple numerical model for data summarization. This model reflects the fact that the higher precision the results need, the more expensive computationally data analysis becomes. In particular, this numerical model accurately captures the trade-off between cost and precision — as stated in Theorem 3.10 — for personalized PageRank approximation. We will consider a simple and abstract summarization problem for this numerical model, aiming to feature the essence of multi-precision sampling in

Significant-PageRank Identification. We will present an optimal sampling algorithm for solving this abstract summarization problem. In Section 3.5, we then extend the solution from this section to PageRank summarization, and design a super-scalable algorithm for Significant-PageRank Identification.

Definition 3.11 (Numerical Black-box)**.** Suppose one has a vector $\mathbf{p} = (p_1, ..., p_n) \in [0,1]^n$, whose entries can only be accessed through queries of the form:

$$\text{Access}(\mathbf{p}, i, \epsilon, \rho)$$

which states an index i, a multiplicative precision $\rho \in (0,1)$, and an additive precision $\epsilon \in (0,1)$. The query $\text{Access}(\mathbf{p}, i, \epsilon, \rho)$ returns a non-negative $\tilde{p}_i$, such that:

$$(1-\rho) \cdot p_i - \epsilon \leq \widetilde{p}_i \leq (1+\rho) \cdot p_i + \epsilon.$$

Furthermore, $\text{Access}(\mathbf{p}, i, \epsilon, \rho)$ incurs a *cost* of:

$$\text{poly}\left(\frac{1}{\rho}\right) \cdot \frac{1}{\epsilon}.$$

The cost of $\frac{1}{\epsilon}$ above reflects the complexity of the state-of-the-art algorithms — as stated in Theorem 3.10 — for personalized PageRank approximation. For other potential applications of the numerical black-box, the trade-off of cost and precision could be different. In fact, our algorithms below can sustain an additional $\log \frac{1}{\epsilon}$ factor in the cost associated with $\text{Access}(\mathbf{p}, i, \epsilon, \rho)$. The inclusion of the multiplicative error also reflects the current best personalized-PageRank approximation algorithm (Theorem 3.10, Part 2) for arbitrary weighted directed networks. In that algorithm,

$$\text{poly}\left(\frac{1}{\rho}\right) = O\left(\frac{1}{\rho^2}\right).$$

The following basic data summarization problem is inspired by Significant-PageRank Identification.

Definition 3.12 (vectorSummarization)**.** Given a threshold parameter $\Delta \in [1:n]$, a gap parameter $c > 1$, and a numerical black-box — Access() — for accessing an unknown vector $\mathbf{p} = (p_1, ..., p_n) \in [0,1]^n$, return "significant" if $\sum_i p_i \geq \Delta$, or "insignificant" if $\sum_i p_i < \frac{\Delta}{c}$.

In this section, we analyze a simple multi-precision sampling procedure and prove the following theorem:

Theorem 3.13 (Fast Vector Summarization)**.** For any confidence parameter $\delta \in (0,1)$, vectorSummarization can be solved with cost $\widetilde{O}_n(1) \cdot \left(\frac{n}{\Delta} \cdot \log \frac{1}{\delta}\right)$, and with probability at least $1 - \delta$.

***Remarks**: In the theorem above, $\widetilde{O}$ also hides a polynomial factor of $\frac{c}{c-1}$, where c is the gap parameter of* vectorSummarization*.*

Robust Estimators for Theorem 3.13

We will now prove Theorem 3.13. Its algorithm is exceptionally simple. In order to illustrate the concept of multi-precision sampling, we will first prove Theorem 3.13 under the following simpler model of the numerical black-box.

Definition 3.14 (Noiseless Numerical Black-box)**.** Suppose one has an unknown vector $\mathbf{p} = (p_1, ..., p_n) \in [0,1]^n$, and one can only access the entries of $\mathbf{p}$ through querying:

$$\text{NoiselessAccess}(\mathbf{p}, i, \epsilon).$$

NoiselessAccess$(\mathbf{p}, i, \epsilon)$ returns p_i if $\epsilon \leq p_i$; otherwise, it returns 0. Furthermore, NoiselessAccess$(\mathbf{p}, i, \epsilon)$ incurs a *cost* of $\frac{1}{\epsilon}$.

Any algorithm for vectorSummarization can only access the unknown vector $\mathbf{p}$ through the numerical black-box. Thus, the algorithm

can only select a sequence of indices $(i_1, ..., i_t)$ and corresponding precision parameters $(\epsilon_1, ..., \epsilon_t)$, and use the querying results:

$$(\text{NoiselessAccess}(i_1, \epsilon_1), \cdots, \text{NoiselessAccess}(i_t, \epsilon_t))$$

to build an estimator for the sum of $\mathbf{p}$. The algorithm also needs to run within the total cost budget of:

$$\sum_{j=1}^{t} \frac{1}{\epsilon_j} = \widetilde{O}_n(1) \cdot \left(\frac{n}{\Delta}\right) \tag{3.9}$$

The algorithm has no prior information about the black-box vector $\mathbf{p}$, except that $\mathbf{p}$ has n entries between 0 and 1, and it can only make sub-linear number of queries, i.e., $t = o(n)$. Using a basic information-theoretical argument, one can show that, if t is fixed, then the best strategy for the algorithm is to simply select $(i_1, ..., i_t)$ uniformly at random from $[n]^t$, independent of the precision parameters it will set.

Therefore, we will focus on the design of precision parameters $(\epsilon_1, ..., \epsilon_t)$. The design must (1) satisfy the cost budget given by Eqn. (3.9), and (2) be sufficient for constructing a robust estimator for the sum of $\mathbf{p}$. Below, focusing on the case when $c \geq 3$, we analyze a simple discrete approach for constructing a super-scalable estimator. The general case, $c > 1$, can be achieved by using an analogous continuous formulation (more on this below).

The discrete formulation below uses the following basic scheme: First, we select a discretization parameter $0 < \beta \leq 1$ and let the sequence of multi-scale parameters be $(\epsilon_i := (1 + \beta)^{-i})$. Accordingly, we divide the interval $(0, 1]$ into subintervals:

$$(\epsilon_1, \epsilon_0], (\epsilon_2, \epsilon_1], \cdots, (\epsilon_k, \epsilon_{k-1}], \cdots$$

To summarize an unknown vector $\mathbf{p}$, the above discretization of $(0, 1]$ suggests the following natural data partition:

$$P_i = \{j \mid \epsilon_i < p_j \leq \epsilon_{i-1}\} \tag{3.10}$$

Therefore, we have:

$$\sum_{j=1}^{n} p_j = \sum_{i=1}^{\infty} \sum_{j \in P_i} p_j \leq (1 + \beta) \cdot \sum_{i=1}^{\infty} |P_i| \epsilon_i < (1 + \beta) \cdot \sum_{j=1}^{n} p_j \tag{3.11}$$

We then use the following observation to limit summation only to a small number of intervals. For $\Delta \in [n]$, let:

$$t_{\beta,\Delta} = \left\lceil \log_{(1+\beta)} \left(\frac{n}{\beta\Delta} \right) \right\rceil \tag{3.12}$$

We say that an entry in $\mathbf{p}$ is *negligible* if its value is less than $\epsilon_{t_{\beta,\Delta}}$. Note that the sum of all negligible entries of $\mathbf{p}$ is at most $n \cdot \epsilon_{t_{\beta,\Delta}} \leq \beta\Delta$. Thus, we can truncate the discrete formula of Eqn. (3.11) at $t_{\beta,\Delta}$. The truncated formula is only inaccurate by at most $\beta\Delta$.

Algorithmically, Eqn. (3.11) reduces the approximation of VECTOR-SUMMARIZATION to approximate counting at each precision scale. It costs $\frac{1}{\epsilon}$ to make a query with precision ϵ. Thus the algorithm can make at most $\widetilde{O}_n(1) \cdot (\epsilon_t \cdot \frac{n}{\Delta})$ queries at precision ϵ_t when given a cost budget of $\widetilde{O}_n(1) \cdot (\frac{n}{\Delta})$. Let τ be a parameter that we will specify later. For each $t \in [t_{\beta,\delta}]$, let $J_t = \{j_{t_1}, ..., j_{n_t}\}$ be $n_t = \tau \cdot \epsilon_t \cdot \frac{n}{\Delta}$ i.i.d. indices chosen uniformly from $[n]$. For each $j_{t_k} \in J_t$, let $q_{j_k} = \text{NoiselessAccess}(\mathbf{p}, j_{i_k}, \epsilon_t)$ be the result returned from the (noiseless) numerical black-box. Let:

$$Q_t = \{j_{t_k} \mid \epsilon_t \leq \mathbf{q}[j_{t_k}] < \epsilon_{t-1}\}.$$

We have:

$$\mathrm{E}\left[\sum_{s \in Q_t} q_s\right] = \frac{n_t}{n} \cdot \sum_{h \in P_t} p_h \tag{3.13}$$

$$\mathrm{E}\left[|Q_t|\right] = \frac{n_t}{n} \cdot |P_t| \tag{3.14}$$

Thus, $\left(\sum_{s \in Q_t} q_s\right) \cdot \frac{n}{n_t}$ provides an estimator for $\left(\sum_{h \in P_t} p_h\right)$. Given these $\sum_{t=1}^{t_{\beta,\Delta}} n_t$ samples, we use the following natural estimator:

$$S_{\Delta,\beta,\tau} = \sum_{t=1}^{t_{\beta,\Delta}} \left(\sum_{s \in Q_t} q_s \right) \cdot \frac{n}{n_t} \tag{3.15}$$

whose expectation is:

$$\mathrm{E}\left[S_{\Delta,\beta,\tau}\right] = \sum_{t=1}^{t_{\beta,\Delta}} \left(\sum_{h \in P_t} p_h \right).$$

We use estimator $S_{\Delta,\beta,\tau}$ in the following algorithm.

Algorithm: DiscreteVectorSummarization(Access, Δ, c, δ)

1: $\beta = \frac{c-1}{1+7c}$; $t_{\beta,\Delta} = \left\lceil \log_{(1+\beta)} \left(\frac{n}{\beta\Delta}\right) \right\rceil$, $\tau = 4 \cdot \ln \frac{2t_{\beta,\Delta}}{\delta} \cdot \frac{t_{\beta,\Delta}}{\beta^3}$.
2: **for** $t = 1 : t_{\beta,\Delta}$ **do**
3: $\epsilon_t = (1+\beta)^{-t}$; let $\mathbf{s}_t$ be a set of $\tau \cdot \frac{n\epsilon_t}{\Delta}$ i.i.d. indices chosen uniformly from $[n]$.
4: **for** $j = 1 : |\mathbf{s}_t|$ **do** $q_j = \text{NoiselessAccess}(\mathbf{s}_t[j], \epsilon_t)$. **end for**
5: $Q_t = \{s \mid \epsilon_t \le q_s < (1+\beta)\epsilon_t\}$.
6: **end for**
7: $S_{\Delta,\beta,\tau} = \sum_{t=1}^{t_{\beta,\Delta}} \left(\sum_{s\in Q_t} q_s\right) \cdot \frac{n}{n_t}$.
8: **if** $S_{\Delta,\beta,\tau} \ge (1-4\beta)\Delta$ **then** return "significant" **else** return "insignificant" **end if**

Eqn. (3.15) defines a family of estimators parameterized by $0 < \beta < 1$, $\tau \ge 1$, and $\Delta \in [n]$. In the rest of the section, we prove the following lemma:

Lemma 3.15 (Robustness of the Estimator)**.** Let $\delta \in (0,1)$ and $\Delta, c > 1$ be the input parameters for VECTORSUMMARIZATION given in Definition 3.12. Let $\beta = \frac{c-1}{7c}$ and $\tau = 4 \cdot \ln \frac{2t_{\beta,\Delta}}{\delta} \cdot \frac{1}{\beta^3} \cdot t_{\beta,\Delta}$. Then, with probability at least $1-\delta$, $S_{\Delta,\beta,\tau}$ is an estimator for VECTORSUMMARIZATION, satisfying the following *robustness conditions*:

- if $\sum_i p_i \ge \Delta$ then $S_{\Delta,\beta,\tau} \ge (1-4\beta)\cdot\Delta$, and
- if $\sum_i p_i < \frac{\Delta}{c}$ then $S_{\Delta,\beta,\tau} < (1-4\beta)\cdot\Delta$.

In addition, the cost for evaluating estimator $S_{\Delta,\beta,\tau}$ is:

$$O\left(\frac{\tau}{\beta^2} \cdot \log\left(\frac{n}{\beta\Delta}\right) \cdot \frac{n}{\Delta}\right) = \widetilde{O}_n(1) \cdot \frac{n}{\Delta}.$$

The key in proving this lemma is to show the following:

> *Although* $\left(\sum_{s\in Q_t} q_s\right) \cdot \frac{n}{n_t}$ *may not be a robust estimator for* $\left(\sum_{h\in P_t} p_h\right)$*, the choice of* $(n_1, ..., n_{t_{\beta,\Delta}})$ guarantees *that* $S_{\Delta,\beta,\tau}$ *is a robust estimator of* $\sum_{t=1}^{t_{\beta,\Delta}} \left(\sum_{h\in P_t} p_h\right)$.

We say P_t is *heavy* in $\mathbf{p}$ if $t \leq t_{\beta,\Delta}$, and:

$$\sum_{h \in P_t} p_h \geq \frac{\beta\Delta}{t_{\beta,\Delta}} \tag{3.16}$$

Let $\text{Heavy}(\mathbf{p}) := \{t : P_t \text{ is heavy}\}$. The next lemma shows that $\text{Heavy}(\mathbf{p})$ alone provides sufficient data for approximating the sum:

Lemma 3.16 (Discrete Multi-Scale Approximation)**.** For any $\mathbf{p} = (p_1, ..., p_n) \in [0,1]^n$, $\beta \in (0, \frac{1}{3})$, and $\Delta \in [n]$:

$$(1+\beta)^{-1}\left(\sum_{j=1}^{n} p_j\right) - 2\beta\Delta \leq \sum_{t \in \text{Heavy}(\mathbf{p})} |P_t|\epsilon_t \leq \sum_{j=1}^{n} p_j \tag{3.17}$$

Proof.

$$\begin{aligned}
\sum_{j=1}^{n} p_j &= \sum_{t=1}^{\infty} \sum_{h \in P_t} p_h \leq \sum_{t=1}^{t_{\beta,\Delta}} \sum_{h \in P_t} p_h + \beta\Delta \\
&= \left(\sum_{t \in \text{Heavy}(\mathbf{p})} \sum_{h \in P_t} p_h\right) + \left(\sum_{t \notin \text{Heavy}(\mathbf{p}) \& t \leq t_{\beta,\Delta}} \sum_{h \in P_t} p_h\right) + \beta\Delta \\
&\leq (1+\beta) \cdot \left(\sum_{t \in \text{Heavy}(\mathbf{p})} |P_t|\epsilon_t\right) + \beta\Delta + \beta\Delta
\end{aligned}$$

from which the Lemma follows. □

Lemma 3.16 also provides a guarantee regarding $\sum_{t \in \text{Heavy}(\mathbf{p})} \sum_{h \in P_t} p_h$, because:

$$\sum_{t \in \text{Heavy}(\mathbf{p})} |P_t|\epsilon_t \leq \sum_{t \in \text{Heavy}(\mathbf{p})} \sum_{h \in P_t} p_h \leq (1+\beta) \cdot \sum_{t \in \text{Heavy}(\mathbf{p})} |P_t|\epsilon_t.$$

Therefore, to prove that $S_{\Delta,\beta,\tau}$ (of Eqn. (3.15)) is a robust estimator for the sum of $\mathbf{p}$, it is sufficient to show that the following holds with high probability:

1. $\forall t \in \text{Heavy}(\mathbf{p})$, the n_t samples J_t induce a robust estimating set Q_t for P_t, and
2. $\forall t \notin \text{Heavy}(\mathbf{p})$, $\left(\sum_{s\in Q_t} q_s\right) \cdot \frac{n}{n_t}$ does not excessively overestimate $(\sum_{h\in P_t} p_h)$. More precisely, the probability that $\left(\sum_{s\in Q_t} q_s\right) \cdot \frac{n}{n_t} > (\sum_{h\in P_t} p_h) + \beta \cdot \frac{\Delta}{\log n}$ is small.

To see Part (1), note that if $\mathbf{p}$ is heavy at scale ϵ_t, then by (3.16):

$$(1+\beta)|P_t|\epsilon_t \geq \sum_{h\in P_t} p_h \geq \frac{\beta\Delta}{t_{\beta,\Delta}}.$$

Thus, $|P_t| \geq \frac{\beta\cdot\Delta}{(1+\beta)\epsilon_t \cdot t_{\beta,\Delta}}$, and consequently:

$$\mathrm{E}\left[|Q_t|\right] = \tau \cdot \frac{n\epsilon_t}{\Delta} \cdot \frac{|P_t|}{n} \geq \frac{\tau\epsilon_t}{\Delta} \cdot \frac{\beta\cdot\Delta}{(1+\beta)\epsilon_t \cdot t_{\beta,\Delta}} = \frac{\tau\cdot\beta}{(1+\beta)t_{\beta,\Delta}}.$$

By the standard multiplicative Chernoff-Hoeffding bound [14, 88]:

$$\begin{aligned}
\Pr\left[\ |Q_t| > (1+\beta)\mathrm{E}\left[|Q_t|\right]\ \right] &\leq e^{-\frac{\beta^2}{4}\cdot \mathrm{E}[|Q_t|]} \leq e^{-\frac{\beta^2}{4}\cdot\frac{\tau\cdot\beta}{t_{\beta,\Delta}}}. \\
\Pr\left[\ |Q_t| < (1-\beta)\cdot\frac{\tau\cdot\beta}{t_{\beta,\Delta}}\right] &\leq \Pr\left[\ |Q_t| < (1-\beta)\cdot \mathrm{E}\left[|Q_t|\right]\ \right] \\
&\leq e^{-\frac{\beta^2}{2}\cdot\frac{\tau\cdot\beta}{t_{\beta,\Delta}}}.
\end{aligned}$$

To bound the probability of Part (2), note that if $t \notin \text{Heavy}(\mathbf{p})$, then by (3.16):

$$|P_t|\epsilon_t \quad < \quad \sum_{h\in P_t} p_h < \frac{\beta\Delta}{t_{\beta,\Delta}}.$$

Thus:

$$\mathrm{E}\left[|Q_t|\right] = \tau \cdot \frac{n\epsilon_t}{\Delta} \cdot \frac{|P_t|}{n} \leq \frac{\tau\cdot\beta}{t_{\beta,\Delta}}.$$

Again, by the Chernoff-Hoeffding bound:

$$\Pr\left[\ |Q_t| > (1+\beta)\cdot\frac{\tau\cdot\beta}{t_{\beta,\Delta}}\ \right] \leq e^{-\frac{\beta^2}{4}\cdot\frac{\tau\cdot\beta}{t_{\beta,\Delta}}}.$$

We now analyze algorithm VectorSummarization (under the noiseless numerical black-box): By the union bound over all scale parameters, with probability at least $1 - 2t_{\beta,\Delta} \cdot e^{-\frac{\beta^2}{4} \cdot \frac{\tau \cdot \beta}{t_{\beta,\Delta}}} = (1-\delta)$, the following holds for all $t \in [t_{\beta,\Delta}]$:

1. **For scales with significant data, we can obtain a robust estimator:** If $t \in \text{Heavy}(\mathbf{p})$, then:

$$(1-\beta) \cdot \frac{\tau \cdot \beta}{t_{\beta,\delta}} \leq (1-\beta)\text{E}\left[|Q_t|\right] \leq |Q_t| \leq (1+\beta)\text{E}\left[|Q_t|\right].$$

   Thus:

$$(1+\beta)^{-2} \leq \frac{\sum_{t \in \text{Heavy}(\mathbf{p})} \left(\sum_{s \in Q_t} q_s\right) \cdot \frac{n}{n_t}}{\sum_{t \in \text{Heavy}(\mathbf{p})} \left(\sum_{h \in P_t} p_h\right)} \leq (1+\beta)^2.$$

2. **For scales with insignificant data, the total overestimation is unlikely large:** If $i \notin \text{Heavy}(\mathbf{p})$, then:

$$\begin{aligned}
\sum_{t \notin \text{Heavy}(\mathbf{p})} \left(\sum_{s \in Q_t} q_s\right) \cdot \frac{n}{n_t} &\leq \sum_{t \notin \text{Heavy}(\mathbf{p})} (1+\beta)\frac{n}{n_t} \cdot |Q_t|\epsilon_t \\
&\leq \sum_{t \notin \text{Heavy}(\mathbf{p})} \beta(1+\beta)^2 \cdot \frac{\Delta}{t_{\beta,\Delta}} \\
&\leq \beta \cdot (1+\beta)^2\Delta.
\end{aligned}$$

Putting these together, we have:

$$(1+\beta)^{-2}\left(\sum_{j=1}^{n} p_j\right) - 2\beta\Delta \leq S_{\Delta,\beta,\tau} \leq (1+\beta)^2\left(\sum_{j=1}^{n} p_j\right) + \beta(1+\beta)^2\Delta.$$

Therefore, on the one hand, if $\sum_{j=1}^{n} p_j \geq \Delta$, then:

$$S_{\Delta,\beta,\tau} \geq (1-\beta)^2\Delta - 2\beta\Delta \geq (1-4\beta)\Delta.$$

On the other hand, if $\sum_{j=1}^{n} p_j < \frac{\Delta}{c}$, by $\beta = \frac{c-1}{7c}$, which is the same as

$\frac{1}{c} = 1 - 7\beta$, we have:

$$\begin{aligned} S_{\Delta,\beta,\tau} &\leq (1+\beta)^2 \cdot \frac{\Delta}{c} + \beta(1+\beta)^2\Delta \\ &= \Big((1+\beta)^2(1-7\beta) + \beta(1+\beta)^2\Big) \cdot \Delta \\ &= \Big(1 - 4\beta - 11\beta^2 - 6\beta^3\Big) \cdot \Delta \\ &< (1-4\beta)\Delta. \end{aligned}$$

The above inequalities imply that, with probability at least $(1-\delta)$, $[S_{\Delta,\beta,\tau} \geq (1-4\beta)\cdot\Delta]$ is a correct indicator for separating $\sum_{j=1}^{n} p_j \geq \Delta$ from $\sum_{j=1}^{n} p_j < \frac{\Delta}{c}$. Thus, we can use this indicator to solve VECTOR-SUMMARIZATION.

Dealing with Additive/Multiplicative Errors

In Algorithm DiscreteVectorSummarization, it is critical to assume that the numerical black-box is *noiseless*. However, we need to cope with the inherent additive and multiplicative errors in the numerical black-box of Definition 3.11 in order to solve SIGNIFICANTPAGERANK.

To overcome these errors, Borgs *et al.* [65] used a more "continuous" estimator, which can be viewed as a multi-scaled variation of the following standard estimator:

Proposition 3.17. Suppose q is a uniform random element from an array $\mathbf{p} = (p_1, ..., p_n) \in [0,1]^n$. Then, for any precision parameter $\epsilon > 0$:

$$\mathrm{E}\left[q \cdot \mathbf{I}[q \geq \epsilon]\right] \leq \frac{1}{n}\sum_{j=1}^{n} p_j \leq \mathrm{E}\left[q \cdot \mathbf{I}[q \geq \epsilon]\right] + \epsilon \tag{3.18}$$

The scheme of Borgs *et al.* [65] takes the following two steps:

1. Draw T i.i.d. uniform samples $Q = (q_1, ..., q_T)$ from $\mathbf{p} = (p_1, ..., p_n)$.
2. Apply the following multi-scale adaptation of Equation (3.18) as the estimator:

$$S_Q = \frac{n}{T}\sum_{t=1}^{T} \mathbf{I}[q_t \geq \epsilon_t], \text{ where } \forall t \in [T],\ \epsilon_t = \frac{t}{T}. \tag{3.19}$$

The estimator S_Q of Eqn. (3.19) is formulated according to the Riemann sum approximating the following well-known integral equation, which is the basis for Proposition 3.17:

Proposition 3.18. For any random variable $q \in [0, 1]$:

$$\mathrm{E}\left[q\right] = \int_0^1 \Pr\left[q \geq x\right] dx.$$

The parameter T will be set to $\widetilde{O}_n(1) \cdot \frac{n}{\Delta}$. For this estimator, the interval $[0, 1]$ is discretized into subintervals of length $\frac{1}{T}$. Note that Equation (3.19) uses the simpler $\mathbf{I}[q_t \geq \epsilon_t]$ rather than $q_t\mathbf{I}[q_t \geq \epsilon_t]$ as in Equation (3.18). The simplification is possible because each element in $\mathbf{p}$ has T chances to test against the multi-scale precisions $(\epsilon_t = \frac{t}{T})_{t\in[T]}$. Now, we have:

$$\mathrm{E}\left[S_Q\right] = \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{n}\mathbf{I}[p_j \geq \epsilon_t] = \frac{1}{T}\sum_{j=1}^{n}\sum_{t=1}^{T}\mathbf{I}\left[p_j \geq \frac{t}{T}\right] \tag{3.20}$$

Thus:

$$\sum_{j=1}^{n} p_j - \frac{1}{T} \leq \mathrm{E}\left[S_Q\right] \leq \sum_{j=1}^{n} p_j \tag{3.21}$$

However, we can only access $\mathbf{p}$ through the numerical black-box, which makes both additive and multiplicative errors. Borgs *et al.* (carefully) set the multiplicative parameter ρ according to the gap parameter c. To draw the t^{th} sample, they take a uniform random index $i_t \in [n]$, and obtain an approximate sample $q_t = \mathrm{Access}(\mathbf{p}, i_t, \epsilon_t, \rho)$. Let $Q = (q_1, ..., q_T)$ be these approximate samples selected. The numerical black-box access guarantees:

$$(1 - \rho)p_{i_t} - \epsilon_t \leq q_t \leq (1 + \rho)p_{i_t} + \epsilon_t.$$

In comparison, $S_{\Delta,\beta,\tau}$ of DiscreteVectorSummarization can be viewed as a crude approximation of S_Q which is robust against both additive and multiplicative errors of data in Q. The earlier proof that $S_{\Delta,\beta,\tau}$ is a robust estimator for VectorSummarization under the noiseless

numerical black-box model can be extended to the numerical black-box of Definition 3.11. Borgs *et al.* showed that for any gap parameter $c > 1$, it is sufficient to take $T = \widetilde{O}_n(1) \cdot (\frac{n}{\Delta})$ samples to solve VECTOR-SUMMARIZATION with high probability. See Section 4 of [65] for more details. The cost of estimator $S_{\Delta,\beta,\tau}$ is:

$$\sum_{t=1}^{T} \frac{T}{t} = O(T \log T) = \widetilde{O}_n(1) \cdot \left(\frac{n}{\Delta}\right).$$

3.5 Significant-PageRank Identification

Together, Theorem 3.13 and Theorem 3.10 (the second part) provide an $\widetilde{O}_n(1) \cdot (\frac{n}{\Delta})$-time algorithm for solving SIGNIFICANT-PAGERANK IDENTIFICATION for any weighted directed network. We view SIGNIFICANT-PAGERANK IDENTIFICATION as n independent VECTORSUMMARIZATION problems — see Proposition 3.6 — one for each column. For any $v \in V$, we use the local personalized PageRank algorithm of Theorem 3.10 as the Access() function of the "numerical black-box" associated with the v^{th} column $\mathbf{PPR}_{\mathbf{W},\alpha}[:, v]$ of $\mathbf{PPR}_{\mathbf{W},\alpha}$. By Theorem 3.10, Access($\mathbf{PPR}_{\mathbf{W},\alpha}[:, v], u, \epsilon, \rho$) takes $\widetilde{O}_n(1) \cdot \frac{1}{\epsilon}$ time, for any constant $\rho > 1$. So, in $\widetilde{O}_n(1) \cdot \frac{n}{\Delta}$ time, we can determine whether the PageRank of v is at least Δ or whether it is at most $\frac{\Delta}{c}$.

However, it is too expensive to process these columns independently. It would require $\widetilde{O}_n(1) \cdot (\frac{n^2}{\Delta})$ time, which is not even scalable.

To achieve super-scalability, we conduct VECTORSUMMARIZATION simultaneously on all columns. We use sparse vectors to maintain all intermediate steps as well as the final PageRank estimates. We can keep everything in sparse form, because the algorithm for Theorem 3.10 in fact produces a sparse vector $\widetilde{\mathbf{p}}_u$ that has $\widetilde{O}_n(1) \cdot \frac{1}{\epsilon}$ non-zeros. By Theorem 3.10, with probability $1 - \delta$:

$$(1 - \rho) \cdot \mathbf{p}_u[v] - \epsilon \leq \widetilde{\mathbf{p}}_u[v] \leq (1 + \rho) \cdot \mathbf{p}_u[v] + \epsilon, \quad \forall v \in V.$$

In the algorithm for SIGNIFICANT-PAGERANK IDENTIFICATION, the confidence parameter δ (when running VECTORSUMMARIZATION) is set as $\delta := \frac{1}{n^2}$. Then, with probability at least $1 - \frac{1}{n}$, VECTORSUMMARIZATION is effective on all columns of $\mathbf{PPR}_{\mathbf{W},\alpha}$ (by the union bound over

the n columns of $\mathbf{PPR}_{\mathbf{W},\alpha}$). In other words, the algorithm correctly identifies all columns of $\mathbf{PPR}_{\mathbf{W},\alpha}$ whose sum is at least Δ, without including any columns whose sum is less than $\frac{\Delta}{c}$.

The sparse vector $\widetilde{\mathbf{p}}_u$ provides simultaneous black-box access to the u^{th} entries of all columns of $\mathbf{PPR}_{\mathbf{W},\alpha}$. Moreover, $\widetilde{\mathbf{p}}_u$ has $O(\frac{1}{\epsilon})$ non-zero entries. Thus, by Proposition 2.5, each step of VectorSummarization takes $\widetilde{O}_n(1) \cdot \frac{1}{\epsilon}$ operations to simultaneously update all columns.

Because the algorithm for Theorem 3.10 retains sparse vectors, the sums for "insignificant" columns are implicitly set to 0. The time complexity for solving Significant-PageRank Identification is precisely the cost of VectorSummarization, as if we were running VectorSummarization on a single unknown abstract vector $\mathbf{p}$. Thus, the time complexity is $\widetilde{O}_n(1) \cdot \left(\frac{n}{\Delta} \cdot \log \frac{1}{\delta}\right) = \widetilde{O}_n(1) \cdot \frac{n}{\Delta}$.

4

Clustering: Local Exploration of Networks

Identifying groups of nodes with significant structural coherence is a fundamental problem in network analysis. Like document classification for Web search and data retrieval, understanding sub-structures of networks is essential for *community identification* and *network clustering.* Sub-structural analysis remains a conceptually challenging problem.

In this chapter, we will focus on a clustering approach based on *spectral graph theory*, which has led to a family of local clustering algorithms with provably-good quality guarantees. To effectively conduct sub-structural network analysis, we must meaningfully characterize coherent groups. A widely-used first step is to formulate a *clusterability measure* that models the level at which nodes in a group $S \subset V$ tend to *interact* among themselves in a network $G = (V, E, \mathbf{W})$.

Mathematically, a clusterability measure takes form:

$$\text{clusterability}_{\mathbf{W}} : 2^V \longrightarrow [0, 1].$$

Clusterability can then used as the *fitness score* for determining if a group is (potentially) a good cluster or community. For example, when $\text{clusterability}_{\mathbf{W}}$ is formulated so that more desirable groups have smaller measures, we classify $S \subset V$ as a *good cluster* in G, if $\text{clusterability}_{\mathbf{W}}(S)$ is smaller than a target threshold.

> *Like many* inverse problems *in machine learning, characterizing coherent groups is conceptually challenging. This is particularly true because different interpretations of network data lead to many competing clusterability measures.*

Among the clusterability measures proposed in the literature [186, 178, 318, 305, 261, 254, 232, 231, 2, 3, 90], some are based on physical intuition of interactions, while others are based on statistical properties. In addition to graph structures, different information and mathematical processes — such as diffusion, the spread of diseases, evolution of ideas, impacts of innovation — can lead to different rules for network interactions. If we believe that clusterability should reflect the level of interaction among nodes, then different clusterability measures may be needed for different processes. Thus, a central question for understanding the interplay between interaction processes and network structures is the following:

> *How should we determine which clusterability measure to use, when given an interaction model or a dynamic process for networks?*

In this chapter, we focus on *conductance* and its extensions. The *conductance* of a group is the ratio of its *external connection* to its *total connection*: For network $G = (V, E, \mathbf{W})$, let $d_i = \sum_{j \in V} w_{i,j}$ be the *weighted degree* of $i \in V$, and let $\mathbf{D_W}$ be the diagonal matrix of $[d_1, ..., d_n]$. For any $S \subset V$, let:

$$d_i^{external(S)} := \sum_{j \in \bar{S}} w_{i,j} \quad \text{and} \quad \text{vol}_{\mathbf{W}}(S) := \sum_{i \in S} d_i \tag{4.1}$$

Then:

$$\text{cut}_{\mathbf{W}}(S, \bar{S}) := \sum_{i \in S} d_i^{external(S)} = \sum_{u \in S, v \notin S} w_{u,v} \tag{4.2}$$

is the (external) connection involving nodes in S. For $S \subseteq V$, define:

$$\text{conductance}_{\mathbf{W}}(S) := \frac{\text{cut}_{\mathbf{W}}(S, \bar{S})}{\min(\text{vol}_{\mathbf{W}}(S), \text{vol}_{\mathbf{W}}(\bar{S}))} \tag{4.3}$$

Proposition 4.1. For any $S \subseteq V$ in a network $G = (V, E, \mathbf{W})$, $\text{conductance}_{\mathbf{W}}(S) = \text{conductance}_{\mathbf{W}}(\bar{S})$.

With this measure of clusterability, we say that a group of nodes is a *good cluster* if it has low conductance, i.e., if its internal connections are significantly richer than its external connections [318, 186]. Let $\phi_{\mathbf{W}}$ denote the conductance of the "best" cluster in G:

$$\phi_{\mathbf{W}} := \min_{S \in V} (\text{conductance}_{\mathbf{W}}(S)) \tag{4.4}$$

This graph-theoretical quantity is the central subject of following beautiful theorem [82] in spectral graph theory:

Theorem 4.2 (Cheeger's Inequality)**.** For any $G = (V, E, \mathbf{W})$, let $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ be the eigenvalues of the normalized Laplacian matrix of G:

$$\mathcal{L}_{\mathbf{W}} = \mathbf{D}_{\mathbf{W}}^{-1/2}(\mathbf{D}_{\mathbf{W}} - \mathbf{W})\mathbf{D}_{\mathbf{W}}^{-1/2} = \mathbf{I} - \mathbf{D}_{\mathbf{W}}^{-1/2}\mathbf{W}\mathbf{D}_{\mathbf{W}}^{-1/2}.$$

Then, $\lambda_1 = 0$ and λ_2 satisfies the following inequalities:

$$\frac{\phi_{\mathbf{W}}^2}{2} \leq \lambda_2 \leq 2 \cdot \phi_{\mathbf{W}} \tag{4.5}$$

In Section 4.6, we will consider a framework for understanding the interplay between a given dynamic process, such as diffusion and random walks, and its underlying network. This framework provides a clusterability characterization that captures conductance as a special case. In Section 4.7, we will study the conductance measure through this broader characterization, and prove Cheeger's inequality in this broader context. This proof will start our tour through spectral graph theory by first focusing on network interactions and clusterability. The parameterization of Cheeger's inequality systematically connects a family of basic clusterability measures with the spectral property of the matrix operator associated with the dynamic process. But first, we will focus on scalable local algorithms in the next five sections. We will use problems of identifying clusters with low conductance and of approximating personalized PageRank vectors as our examples.

4.1 Local Algorithms for Network Analysis

Local exploration of networks is a widely-used technique in Internet algorithms. For example, in distributed Web crawling, many processors conduct local exploration of this massive information network in order to index the vast Web contents [332]. In content delivery, processors run local mapping algorithms to build global Internet routing schemes [264]. The task of these local algorithms is to explore massive *hidden* networks. In this chapter, we will focus on the use of local algorithms for scalable network analysis.

> *Like sampling methods, the goal of* local algorithms *is to* find *a solution* without *examining the entire data.*

Unlike classical sampling approaches, local algorithms usually take "continuous" steps to explore input data, and use local details to form solutions. The combination of sampling and effective local algorithms — as illustrated by simulated annealing, and our example in Chapter 3 for Significant-PageRank Identification — provides a powerful framework for designing scalable algorithms.

Local Exploration of Networks

We will first define what it means to "locally explore" a network. We will use the framework introduced in [318] for characterizing *local algorithms* in network analysis. Similar formulations have also been used in distributed computing [321] and property testing [153, 288].

Definition 4.3 (Local Network-Analysis Algorithms)**.** In this framework, the underlying network $G = (V, E, \mathbf{W})$ is "hidden." The input to a network algorithm is a *starting node*, or a *starting subset* $S \subset V$ of G (which is much smaller than $|V|$). The network algorithm is *local*, if at each step, it only examines nodes connected to those it has seen before.

In other words, initially, a local algorithm knows nothing about the hidden network G, other than the starting data $S \subset V$. So during its first step, it can only access the neighbors of nodes in S. After

that, it can only access the neighbors of the nodes that it has so far accessed. For example, Facebook is largely a "private" network. A local algorithm for exploring Facebook from a given user, say "Christos Papadimitriou," can only access the user's Facebook friends, and subsequent friends of these friends, and so on.

This appears to be a restricted way to access a network. Then again, it is not: In terms of Definition 4.3, both breadth-first search (BFS) and depth-first search (DFS) are local explorations.

> *The impact of this restriction is only apparent, if the algorithm has to make decisions before accessing the entire network.*

Consequently, it is more subtle to capture the scalability of a local algorithm than a global one, because size and length are no longer apparent. To do so, we must address the following questions:

- How should we define the size of the problem, when local algorithms have to produce solutions with only partial access of input data?
- How long should a local algorithm run?
- How much data can it explore?

Example Problems

In network analysis, clustering is a quintessential problem for studying local algorithms [318]. Without looking at the whole network, a local clustering algorithm attempts to identify a cluster, satisfying a target clusterability condition, that either contains or is near a given node. In this section, we will consider the following local clustering problem:

Problem 4.4 (Local Clustering)**.** Given a node v of a network $G = (V, E, \mathbf{W})$ and a target conductance ϕ, find a cluster S containing v such that $\text{conductance}_{\mathbf{W}}(S) \leq \phi$.

As another example for illustrating local network-analysis algorithms, we will consider scalable approximation algorithms for personalized PageRank [177, 22, 65]. These scalable solutions — as stated in Theorem 3.10 — are crucially used in Chapter 3 for solving Significant-PageRank Identification.

Problem 4.5 (PPRApproximation)**.** Given $u \in V$ in a directed network $G = (V, E, \mathbf{W})$ and a precision parameter ϵ, compute a sparse vector $\widetilde{\mathbf{p}}_u$ with $O(\frac{1}{\epsilon})$ non-zeros such that:

$$\mathbf{p}_u[v] - \epsilon \leq \widetilde{\mathbf{p}}_u[v] \leq \mathbf{p}_u[v], \quad \forall v \in V \tag{4.6}$$

In the rest of this section, we will analyze the mathematical structures intrinsic to the solutions of these two local-computation problems. Performance benchmarks derived from these mathematical structures are then used to model the efficiency and scalability of their local algorithms.

We will also address the following subtle issue in the analysis of local algorithms. A local algorithm may not always find a solution from some starting nodes in the network. Such a possibility can arise because:

- there is no solution nearby, or
- the local algorithm failed to find a solution in the time allowed.

Thus, we need to determine how much time each local algorithm is allowed to search the network. We also need to analyze the statistical guarantee of the local algorithms, particularly regarding how often and under what condition, that the local algorithm succeeds.

Scalability of Local Clustering

For local clustering with starting node v, the size of the network data associated with the cluster identified by a local algorithm provides a natural benchmark for complexity analysis. We will use the following definition as the first-order estimate of this size.

Suppose $G = (V, E, \mathbf{W})$ is a weighted directed network. For $u \in V$, let $N^{out}(v) = \{v : (u, v) \in E\}$ denote u's "out" neighbors in G. Similarly, let $N^{in}(v) = \{v : (v, u) \in E\}$.

Definition 4.6 (Graph Volume)**.** For any non-empty $S \subseteq V$, the *graph volumes* of S in $G = (V, E, \mathbf{W})$ are:

$$\mathrm{vol}_G^{out}(S) := \sum_{v \in S} |N^{out}(v)| \quad \text{and} \quad \mathrm{vol}_G^{in}(S) := \sum_{v \in S} |N^{in}(v)|$$

Further, let $\mathrm{vol}_G(S) := (\mathrm{vol}_G^{out}(S) + \mathrm{vol}_G^{in}(S))/2$.

Note that typically $\mathrm{vol}_G(S) \neq \mathrm{vol}_{\mathbf{W}}(S)$ in weighted networks. For any undirected network G, $\mathrm{vol}_G(S) = \mathrm{vol}_G^{out}(S) = \mathrm{vol}_G^{in}(S)$, and if G is connected, then $\mathrm{vol}_G(S) \geq |S|$.

Following [318], efficient local clustering algorithms should be *output-sensitively scalable*: Whenever they returns a *non-empty cluster* $C \subset V$ in the hidden network $G = (V, E, \mathbf{W})$, the running time should be $\widetilde{O}_n(1) \cdot \mathrm{vol}_G(C)$. In other words, every successful run of scalable local clustering algorithms should be scalable with respect to the size of the output structure.

This condition suggests the following: A local clustering algorithm has three input parameters (v, ϕ, T), where:

1. $v \in V$ is the starting node for accessing the hidden network $G = (V, E, \mathbf{W})$,
2. ϕ is the target conductance, and
3. T is the time allowed for the algorithm to return the empty set.

Naturally, the algorithm can simply return the empty set, which is clearly not desirable for scalable local clustering. In Section 4.2, we will discuss a statistical guarantee that can be achieved for scalable local clustering, which offers a way to measure the inexact term, "usually," in the following statement:

> *A scalable local clustering algorithm may not find a cluster for some input nodes, but is usually successful.*

Scalability of Local Personalized PageRank Approximation

A local algorithm for personalized PageRank approximation has two input parameters, u and $\epsilon > 0$, where the second parameter, ϵ, defines the desired accuracy. Together, the personalized PageRank $\mathbf{p}_u$ and precision parameter ϵ define the following two natural complexity benchmarks:

1. the number of entries in $\mathbf{p}_u$ that are larger than ϵ, and
2. $\frac{1}{\epsilon}$, the worst-case output size for PPRApproximation.

We say a local personalized PageRank approximation algorithm is *scalable* if for all $u \in V$ and $\epsilon \in (0, 1]$, its running time is:

$$\widetilde{O}_n(1) \cdot \text{benchmark}(u, \epsilon).$$

Note that the second benchmark is at least as large as the first one. Thus, we say that a local personalized PageRank approximation algorithm is *baseline* scalable if it always runs in $\widetilde{O}_n(1) \cdot \frac{1}{\epsilon}$ time. As stated in Theorem 3.10, the local algorithms of [177, 22, 65] are all measured based on this baseline scalability benchmark.

Scalability of Locally Computable Properties

It has been a fruitful research direction to characterize graph-theoretical properties that are (approximately) computable by scalable local algorithms. For example, the scalable local clustering algorithm discussed in this Chapter can be extended to find a cluster containing a starting node that has target edge density [18]. In Section 8.1, we will give another example of how a scalable local algorithm can be used to identify communities containing a given member in a preference network. The local algorithms for personalized PageRank approximation have also been generalized for PageRank contributions [21] and for other families of personalized ranking vectors [93].

For general graph properties, modeling *local scalability* can be a delicate task.

- From the perspective of complexity theory, the local scalability of a graph property should capture the performance that local algorithms can achieve with respect to the size of the substructures that have the desired property.
- From the perspective of algorithm design, the local scalability should capture the usefulness of the local algorithms in "global" network optimization.

For example, in both local clustering and personalized PageRank approximation, we can partially justify the definition of local scalability: If we achieve this performance measure successfully, then we can obtain scalable optimization algorithms, respectively, for graph partitioning [318] and Significant-PageRank Identification [65].

Each of these optimization algorithms integrates a global sampling scheme with the local network-exploration procedure enabled by the corresponding scalable local algorithm.

4.2 Local Clustering and Random Walks

In this section, we will study local clustering, which has three input parameters: (i) $v \in V$, the starting node in a hidden network $G = (V, E, \mathbf{W})$, (ii) ϕ, the target conductance, and (iii) $T > 0$, the time allowed for a failed search.

The local clustering algorithm of [318] — called Nibble — uses the distributions of *short random walks* to identify local clusters. It achieves the following scalability and statistical guarantees:

Scalability: If $C = \mathsf{Nibble}(v, \phi, T)$ is non-empty, then (i) the running time is $\widetilde{O}_n(1) \cdot \mathrm{poly}(\frac{1}{\phi}) \cdot \mathrm{vol}_G(C)$, and (ii) $\mathrm{conductance}_{\mathbf{W}}(C) \leq \phi$.

Effective Termination: If $C = \mathsf{Nibble}(v, \phi, T)$ is empty, then the running time is $\widetilde{O}_n(1) \cdot T$.

Statistical Guarantee: For every subset $S \subset V$ with small conductance, $\mathsf{Nibble}(v, \phi, T)$ is *usually successful* for $v \in S$.

More specifically, for any $S \subset V$ (with $\frac{\mathrm{vol}_{\mathbf{W}}(S)}{\mathrm{vol}_{\mathbf{W}}(V)} \leq \frac{1}{2}$), consider the distribution where each $v \in S$ is chosen with probability proportionally to its degree d_v, i.e., according to $\boldsymbol{\psi}_S[v] = \frac{d_v}{\mathrm{vol}_{\mathbf{W}}(S)}$ for $v \in S$.

Then, with high probability, Nibble identifies a cluster C such that (i) $\text{conductance}_{\mathbf{W}}(C) \leq \phi$, and (ii) C has significant overlap with S.

In particularly, $\exists c = \widetilde{O}_n(1)$ such that the following holds with probability at least $\frac{1}{2}$, over the choice of v according distribution $\boldsymbol{\psi}_S$:

$\forall \phi \geq \sqrt{c \cdot \text{conductance}_{\mathbf{W}}(S)}$, $\exists b \in [0 : \log(\text{vol}_{\mathbf{W}}(S))]$ such that:

1. **Successful Identification**: $C = \mathsf{Nibble}(v, \phi, 2^b) \neq \emptyset$, and
2. **Significant Recovery**: $\text{vol}_{\mathbf{W}}(C \cap S) \geq 2^{b-1}$.

The quadratic gap between $\text{conductance}_{\mathbf{W}}(S)$ and the guarantee on $\text{conductance}_{\mathbf{W}}(C)$ is consistent with the quadratic gap exhibited in the Cheeger's inequality (Theorem 4.24). We will comment on the role of b below after we define algorithm Nibble.

Sweep with Eigenvectors

Several provably-good local clustering algorithms now exist [318, 22, 93, 24, 23]. All have roots in spectral graph theory, and can be viewed as a local version of the spectral partitioning scheme underlying the proof of Cheeger's inequality:

> *There exists a vector $\mathbf{v} \in \mathbb{R}^n$ that can be used in the following algorithm,* Sweep*, to identify a cluster with probably-good conductance in the input network.*

Algorithm: $\mathsf{Sweep}(G, \mathbf{v})$

Require: $G = (V, E, \mathbf{W})$ and $\mathbf{v} \in \mathbb{R}^{|V|}$

1: Let $\boldsymbol{\pi}$ be an ordering of V according to $\mathbf{v}$, i.e., $\forall k \in [n-1]$, $\mathbf{v}[\pi(k)] \geq \mathbf{v}[\pi(k+1)]$; let $S_k = \{\pi(1), ..., \pi(k)\}$.

2: Let $k^* = \text{argmin}_k \text{ conductance}_{\mathbf{W}}(S_k)$. Return S_{k^*}.

The next theorem — which follows immediately from the proof of Theorem 4.24 — illustrates the algorithmic essence of Cheeger's inequality that eigenvectors of (parameterized) graph Laplacians can be effectively used in Sweep to identify a quality cluster.

Theorem 4.7 (Cheeger Sweep)**.** For any network $G = (V, E, \mathbf{W})$, let $\mathbf{v}_2$ be the eigenvector of $\mathcal{L}_{\mathbf{W}}$ associated with its second smallest eigenvalue. Let $\mathbf{v} = \mathbf{D}_{\mathbf{W}}^{-\frac{1}{2}}\mathbf{v}_2$ and $S_{k^*} = \mathsf{Sweep}(G, \mathbf{v})$. Then:

$$\text{conductance}_{\mathbf{W}}(S_{k^*}) \leq 2\sqrt{\phi_{\mathbf{W}}} \tag{4.7}$$

The Robustness of Sweep

One does not have to use eigenvectors in Sweep in order to identify clusters with probably-good conductance. In fact, all scalable local clustering algorithms rely on the use of *sparse*, *locally constructed* "spectral vectors." For example, Mihail [47] proved the following theorem:

Theorem 4.8 (Rayleigh-Quotient and Cheeger's Inequality)**.** Let $G = (V, E, \mathbf{W})$ be a network, and $\widetilde{\mathbf{v}}$ be a vector perpendicular to $\mathbf{D}_{\mathbf{W}}^{\frac{1}{2}} \cdot \mathbf{1}$ (the eigenvector of $\mathcal{L}_{\mathbf{W}}$ corresponding to 0). Let $S_{k^*} = \mathsf{Sweep}(G, \mathbf{D}_{\mathbf{W}}^{-\frac{1}{2}} \cdot \widetilde{\mathbf{v}})$. Then:

$$\text{conductance}_{\mathbf{W}}(S_{k^*}) \leq \sqrt{2 \cdot \frac{\widetilde{\mathbf{v}}^T \cdot \mathcal{L}_{\mathbf{W}} \cdot \widetilde{\mathbf{v}}}{\widetilde{\mathbf{v}}^T \cdot \widetilde{\mathbf{v}}}} \tag{4.8}$$

Mihail's proof directly extends that of Cheeger's inequality (Theorem 4.24). A more elementary proof can also be found in [315].

Local Clustering: Sweeping Locally Constructed Vectors

Spielman and Teng [318] applied Sweep to a set of $\widetilde{O}_n(1)$ locally constructed sparse vectors based on the distributions of random-walks from a starting node v. They show that at least one of the Sweeps likely identifies a cluster with conductance $\widetilde{O}_n(1) \cdot \sqrt{\text{conductance}_{\mathbf{W}}(S)}$, and satisfies all conditions stated at the beginning of this section. The following process defines these sparse vectors: Let $\mathbf{q}^{(t)}$ be the t^{th} step

random-walk distribution starting from $\mathbf{1}_v$ according to transition matrix[1] $\mathbf{M} = \frac{1}{2}(\mathbf{W}\mathbf{D}^{-1} + \mathbf{I})$:

$$\mathbf{q}^{(0)} = \mathbf{1}_v \text{ and } \mathbf{q}^{(t)} = \mathbf{M} \cdot \mathbf{q}^{(t-1)} \tag{4.9}$$

Note that $\mathbf{q}^{(t)}$ can be locally computed from $\mathbf{q}^{(t-1)}$ by only examining neighbors of nodes in the support of $\mathbf{q}^{(t-1)}$. However, $\mathbf{q}^{(t)}$ can be dense even when $\mathbf{q}^{(t-1)}$ is sparse. Nibble maintains sparse vectors that approximate these distributions, which is the key to its scalability. For a given precision parameter $\epsilon > 0$, and a non-negative vector $\mathbf{q}$, let $\lfloor \mathbf{q} \rfloor_\epsilon$ be the truncation of $\mathbf{q}$ by setting every entry with $\mathbf{q}[u] < d_u\epsilon$ to 0. Then, instead of computing $\mathbf{q}^{(1)}, ...\mathbf{q}^{(t)}$, the local algorithm computes:

$$\mathbf{s}^{(0)} = \mathbf{1}_v \text{ and } \mathbf{s}^{(t)} = \lfloor \mathbf{M} \cdot \mathbf{s}^{(t-1)} \rfloor_\epsilon \tag{4.10}$$

Nibble then applies Sweep on vectors derived from $\{\mathbf{s}^{(1)}, ..., \mathbf{s}^{(t)}\}$ to identify a cluster. Below — to avoid becoming bogged down with technical details — we examine a variation of the Nibble. This variation simplifies the original algorithm in two places: (1) Instead of stating parameters precisely, particularly the ϵ for precision and L for the length of short random walks, we use $\widetilde{O}$ notation to state their relations with respect to the input parameters ϕ and T. (2) We use a simplified version implemented in Voevodski *et al.* [348] (for clustering protein networks), that does not test a condition of [318], whose purpose is to ensure Condition (2.b) for the significant recovery of the hidden S.

Algorithm: Nibble(G, v, ϕ, T)

1: Set $L = \widetilde{O}_n(1) \cdot \frac{1}{\phi^2}$; $\epsilon = \frac{1}{\widetilde{O}_n(1) \cdot T \cdot L} = \frac{\phi^2}{\widetilde{O}_n(1) \cdot T}$; $\mathcal{C} = \emptyset$; $\mathbf{r}^{(0)}, \mathbf{s}^{(0)} = \mathbf{1}_v$.
2: **for** $t = 1 : L$ **do**
3: $\quad \mathbf{s}^{(t)} = M \cdot \mathbf{r}^{(t-1)}$, $\mathbf{v}^{(t)} = \mathbf{D}_{\mathbf{W}}^{-1} \cdot \mathbf{s}^{(t)}$, and $\mathbf{r}^{(t)} = \lfloor \mathbf{s}^{(t)} \rfloor_\epsilon$
4: $\quad \mathcal{C} = \mathcal{C} \cup \text{Sweep}(G, \mathbf{v}^{(t)})$.
5: Return $\text{argmin}_{C \in \mathcal{C}}$ conductance$_{\mathbf{W}}(C)$.

In Section 4.3, we outline a proof of the following theorem, whose

[1]This is a "lazy" random walk that stays put with probability $\frac{1}{2}$, and takes a random step with probability $\frac{1}{2}$. Note that the laziness increases the chance the walk stays at the current node.

statement is slightly weaker than the main theorem of [318] discussed at the beginning of this section.

Theorem 4.9 (Simplified Nibble)**.** For any v, ϕ and T:

(1) SCALABILITY: $C = \mathsf{Nibble}(v, \phi, T)$ runs in time $\widetilde{O}_n(1) \cdot \frac{T}{\phi^4}$ and $\text{conductance}_{\mathbf{W}}(C) \leq \phi$ whenever $C \neq \emptyset$.

(2) STATISTICAL GUARANTEES: For some $c = \widetilde{O}_n(1)$, each subset $S \subset V$ (with $\text{vol}_{\mathbf{W}}(S) \leq \frac{\text{vol}_{\mathbf{W}}(V)}{2}$) contains $S_g \subseteq S$ with $\text{vol}_{\mathbf{W}}(S_g) \geq \frac{\text{vol}_{\mathbf{W}}(S)}{2}$ satisfying the following condition: $\forall \phi \geq \sqrt{c \cdot \text{conductance}_{\mathbf{W}}(S)}$, $\forall v \in S_g$, $\exists b \in [0 : \log(\text{vol}_{\mathbf{W}}(S))]$ such that $C = \mathsf{Nibble}(v, \phi, 2^b) \neq \emptyset$.

In the theorem, b plays a delicate role in regulating the trade-off between scalability and statistical guarantees. For any $v \in S_g$, $C = \mathsf{Nibble}(v, \phi, \text{vol}_{\mathbf{W}}(S))$ always successfully identifies a good cluster, i.e., $\text{conductance}_{\mathbf{W}}(C) \leq \phi$. However, when C is too small — i.e., $\text{vol}_{\mathbf{W}}(C) \ll \text{vol}_{\mathbf{W}}(S)$ — this local clustering process is not scalable. The theorem states that in such a case, there exists a smaller setting of b such that $\mathsf{Nibble}(v, \phi, 2^b)$ also successfully identifies a good cluster. Applying the binary-search principle, we can scalably determine the smallest $b \in [0 : \log(\text{vol}_{\mathbf{W}}(S))]$ such that $\mathsf{Nibble}(v, \phi, 2^b) \neq \emptyset$.

In [22], Andersen, Chung, and Lang developed a faster and more elegant solution to local clustering. Instead of sweeping multiple random-walk distribution vectors, their algorithm — PPR-Nibble — uses an approximation $\widetilde{\mathbf{p}}_v$ of the personalized PageRank vector $\mathbf{p}_v$ of the starting node v to assemble random-walk distributions into a single vector. While performance analysis of PPR-Nibble is much cleaner for unweighted graphs, PPR-Nibble applies to all symmetric networks. Here is a simplified version of PPR-Nibble implemented in [348].

Algorithm: PPR-Nibble(G, v, ϕ, T)

1: Set $\alpha = \Theta(\frac{\phi^2}{\log \text{vol}_{\mathbf{W}}(V)})$ and $\epsilon = \frac{1}{\widetilde{O}_n(1) \cdot T \cdot \log \text{vol}_{\mathbf{W}}(V)}$
2: $\widetilde{\mathbf{p}}_v = \mathsf{PushBasedPPRApproximation}(G, \alpha, \epsilon, v)$.
3: $\mathbf{v} = \mathbf{D}_{\mathbf{W}}^{-1} \cdot \widetilde{\mathbf{p}}_v$,
4: Return $\mathsf{Sweep}(G, \mathbf{v}_t, \phi')$.

In the algorithm, PushBasedPPRApproximation(G, α, ϵ, v) approximates the personalized PageRank vector $\mathbf{p}_v$. See Section 4.4 for the algorithm and the approximation guarantee. PPR-Nibble(G, v, ϕ, T) runs in $\widetilde{O}_n(1) \cdot \frac{T}{\phi^2}$ time and likely identifies a cluster C with $\text{conductance}_{\mathbf{W}}(C) \leq O(\sqrt{\text{conductance}_{\mathbf{W}}(S) \cdot \log \text{vol}_{\mathbf{W}}(S)})$ when v is drawn randomly according to $\boldsymbol{\psi}_S$.

4.3 Performance Analysis of Local Clustering

The mathematical foundation of local clustering is based on the following result of Lovász and Simonovits [239, 240]:

> *If the distributions of short random walks from a node do not quickly converge to its stationary, then these distributions can be used to identify a cluster with small conductance.*

Let $\boldsymbol{\psi}$ be the stationary distribution of lazy random walks — given by transition matrix $\mathbf{M} = \frac{1}{2}(\mathbf{W}\mathbf{D}^{-1} + \mathbf{I})$ — on undirected network $G = (V, E, \mathbf{W})$. It is well known that:

$$\boldsymbol{\psi} = (d_1, ..., d_n) / \sum_u d_u \tag{4.11}$$

Recall that $\text{vol}_{\mathbf{W}}(V) = \sum_u d_u$. Then, for $\mathbf{q}^{(t)}$ defined by Eqn. (4.9), we have $\lim_{t\to\infty} \mathbf{D}_{\mathbf{W}}^{-1}\mathbf{q}^{(t)} = \mathbf{1}/\text{vol}_{\mathbf{W}}(V)$. For intermediate t, the quantity $\frac{\mathbf{q}^{(t)}[u]}{d_u} \cdot \text{vol}_{\mathbf{W}}(V)$ measures how far $\mathbf{q}^{(t)}$ is from the stationary distribution at $u \in V$. For example, if this quantity is more than 1, then u has more probability mass in the random-walk distribution than in the stationary distribution $\boldsymbol{\psi}$.

With a fixed tie-breaking rule — say by the labels of nodes in V — the Sweep of vector $\mathbf{D}_{\mathbf{W}}^{-1}\mathbf{q}^{(t)}$ uses the ordering $\boldsymbol{\pi}^{(t)}$ that satisfies:

$$\frac{\mathbf{q}^{(t)}[\boldsymbol{\pi}^{(t)}[i]]}{d_{\boldsymbol{\pi}^{(t)}[i]}} \geq \frac{\mathbf{q}^{(t)}[\boldsymbol{\pi}^{(t)}[i+1]]}{d_{\boldsymbol{\pi}^{(t)}[i+1]}}, \quad \forall i \in [n-1].$$

It defines a family of clusters of the form: $S_k^{(t)} = \{\boldsymbol{\pi}^{(t)}[1], ..., \boldsymbol{\pi}^{(t)}[k]\}$. In other words, $S_k^{(t)}$ consists of the k most favored destinations of the t-step random-walks, when normalized by their stationary probabilities.

By definition, (i) $\boldsymbol{\psi}(V) = \mathbf{q}^{(t)}(V) = 1$ and (ii) for all $k \in [n-1]$:

$$\frac{\mathbf{q}^{(t)}(S_k^{(t)})}{\boldsymbol{\psi}(S_k^{(t)})} \geq \frac{\mathbf{q}^{(t)}(S_{k+1}^{(t)})}{\boldsymbol{\psi}(S_{k+1}^{(t)})}.$$

Suppose the distributions of the random walks fail to converge sufficiently after t steps. Then, there must be a $k_0 \in [n-1]$ such that $\forall k \leq k_0$, $S_k^{(t)}$ attracts much more probability density than its fair share at the stationary, which is $\boldsymbol{\psi}(S_k^{(t)}) = \mathrm{vol}_{\mathbf{W}}(S_k^{(t)})/\mathrm{vol}_{\mathbf{W}}(V)$.

Lovász and Simonovits' result is a local analogue to Cheeger's inequality. Instead of characterizing eigenvalues by conductances, Lovász and Simonovits quantify the convergence speed of random-walks in terms of the conductances of the subsets defined by the sweeps of $\mathbf{D}_{\mathbf{W}}^{-1}\mathbf{q}^{(1)}$, ..., $\mathbf{D}_{\mathbf{W}}^{-1}\mathbf{q}^{(t)}$.

Theorem 4.10 (Lovász and Simonovits)**.** For any $G = (V, E, \mathbf{W})$, $v \in V$, and $T > 1$, if conductance$(S_k^{(t)}) \geq \phi$, $\forall t < T$ and $\forall k \in [1:n)$, then for every subset $U \subseteq V$:

$$\left| \sum_{u \in U} \left(\mathbf{q}^{(T)}[u] - \boldsymbol{\psi}[u] \right) \right| \leq \sqrt{\min(\mathrm{vol}_{\mathbf{W}}(U), \mathrm{vol}_{\mathbf{W}}(V-U)} \left(1 - \frac{\phi^2}{2}\right)^T.$$

Proof. (Highlights) Lovász and Simonovits proved that the convergence of the random-walk distributions to the stationary can be captured by the evolution of a family of piece-wise linear functions $\Pi^{(t)}(x)$ satisfying:

$$\Pi^{(t)}\left(\mathrm{vol}_{\mathbf{W}}(S_k^t)\right) := \mathbf{q}^{(t)}(S_k^{(t)}) \tag{4.12}$$

Let $W = \mathrm{vol}_{\mathbf{W}}(V)$ be the total weights. By the definition of $\boldsymbol{\pi}^{(t)}$, the piece-wise linear function $\Pi^{(t)}(x)$ is concave and $\Pi^{(t)}(W) = 1$. At the stationary, $\lim_{t\to\infty} \Pi^{(t)}(x)$ is a linear function of slope $1/W$. In general, $\Pi^{(t)}(x) \leq \Pi^{(t-1)}(x)$, $\forall x \in [0, W]$.

Lovász and Simonovits [239, 240] showed that not only $\Pi^{(t)}(x)$ monotonically converges to $\lim_{t\to\infty} \Pi^{(t)}(x)$, but convergence is rapid

if the conductances of all $S_k^{(t)}$ are large enough. In particular, they proved that if conductance$(S_k^{(t)}) \geq \phi, \forall k \in [n-1]$, then for $x \leq W/2$:

$$\Pi^{(t)} \leq \frac{1}{2}\left(\Pi^{(t-1)}(x - 2\phi x) + \Pi^{(t-1)}(x + 2\phi x)\right) \tag{4.13}$$

and symmetrically for $W/2 < x \leq W$:

$$\Pi^{(t)} \leq \frac{1}{2}\left(\Pi^{(t-1)}\left(x - 2\phi(\Delta - x)\right) + \Pi^{(t-1)}\left(x + 2\phi(\Delta - x)\right)\right) \tag{4.14}$$

Thus, under the condition conductance$(S_k^{(t)}) \geq \phi, \forall k \in [n-1]$, these two inequalities imply that $\Pi^{(t)}(x)$ drops significantly below $\Pi^{(t-1)}(x)$. One can then repeatedly apply this step to establish the theorem. □

Good Starting Nodes for Local Clustering

The proof of Theorem 4.9 then contains these two main components:

1. **Characterization of Good Starting Nodes**: For each subset $S \subset V$ with conductance$_{\mathbf{W}}(S) \leq \phi^2$, characterize the subset $S_g \subset S$ — according to Theorem 4.9 — that are good starting nodes of Nibble.

2. **Robustness Analysis**: Prove that the truncated distribution vector $\mathbf{s}^{(t)}$ of Nibble remains effective for Sweep in place of $\mathbf{q}^{(t)}$.

Let $\boldsymbol{\psi}_S$ be the distribution such that $\boldsymbol{\psi}_S[u] = 0$ for $v \notin S$ and $\boldsymbol{\psi}_S[v] = d_v/\text{vol}_{\mathbf{W}}(S)$ for $v \in S$. Note that:

> *If the lazy random walk starts at a random node chosen according to $\boldsymbol{\psi}_S$, then with probability* $1 -$ conductance$_{\mathbf{W}}(S)/2$*, the node after a single step remains in S.*

In fact, for any integer T, the probability that the entire T-step lazy random walk stays in S is $1 - T \cdot \text{conductance}_{\mathbf{W}}(S)/2$. Thus, the probability that any T-step random walk escapes S is at most:

$$T \cdot \text{conductance}_{\mathbf{W}}(S)/2 \tag{4.15}$$

Definition 4.11 (T-Step-Random-Walk Core of S)**.** For an integer T, let $\mathrm{Core}_T^{RW}(S)$ be the set of all nodes v of S satisfying the following condition: The probability that a T-step random-walk starting at v terminating outside S is at most $T \cdot \mathrm{conductance}_{\mathbf{W}}(S)$.

By Equation (4.15), the expected escaping probability is at most $T \cdot \mathrm{conductance}_{\mathbf{W}}(S)/2$. Thus, by increasing the escaping tolerance to $T \cdot \mathrm{conductance}_{\mathbf{W}}(S)$, Definition 4.11 guarantees that $\mathrm{Core}_T^{RW}(S)$ is not too small with respect to S. The following proposition can be proved by an averaging argument:

Proposition 4.12 (Richness of RW-Cores)**.** For all $S \subset V$ and T:

$$\mathrm{vol}_{\mathbf{W}}\left(\mathrm{Core}_T^{RW}(S)\right) \geq \frac{\mathrm{vol}_{\mathbf{W}}(S)}{2}.$$

Nibble uses the following intuition. At stationary, the probability density of S is $\boldsymbol{\psi}(S) = \frac{\mathrm{vol}_{\mathbf{W}}(S)}{\mathrm{vol}_{\mathbf{W}}(V)}$. If $(1 - T \cdot \mathrm{conductance}_{\mathbf{W}}(S))$ is significantly larger than $\boldsymbol{\psi}(S)$, then for every $v \in \mathrm{Core}_T^{RW}(S)$, the random-walks starting from v will not converge rapidly because:

$$\left|\mathbf{q}^{(T)}(S) - \boldsymbol{\psi}(S)\right| \geq \left|(1 - T \cdot \mathrm{conductance}_{\mathbf{W}}(S)) - \frac{\mathrm{vol}_{\mathbf{W}}(S)}{\mathrm{vol}_{\mathbf{W}}(V)}\right| \quad (4.16)$$

In particular, by Lovász-Simonovits' theorem, if

$$\left|(1 - T \cdot \mathrm{conductance}_{\mathbf{W}}(S)) - \frac{\mathrm{vol}_{\mathbf{W}}(S)}{\mathrm{vol}_{\mathbf{W}}(V)}\right| \geq \sqrt{\min(\mathrm{vol}_{\mathbf{W}}(U), \mathrm{vol}_{\mathbf{W}}(V-U)}\left(1 - \frac{\phi^2}{2}\right)^T \quad (4.17)$$

then for every $v \in \mathrm{Core}_T^{RW}(S)$, by sweeping the distributions of the T-step random-walks from v, the following holds:

$$\min_{t \in [T], k \in [n-1]} \mathrm{conductance}_{\mathbf{W}}\left(S_k^{(t)}\right) \leq \phi \quad (4.18)$$

Then, condition (4.17) can be satisfied with a proper choice of constants in:

$$\begin{aligned} T &= \Theta(1/\text{conductance}_{\mathbf{W}}(S)) \\ \phi &= \Theta(\sqrt{\text{conductance}_{\mathbf{W}}(S) \cdot \log \text{vol}_{\mathbf{W}}(S)}). \end{aligned}$$

Consequently, by sweeping according to these precise random-walk distributions $(\mathbf{q}^{(1)}, ..., \mathbf{q}^{(T)})$, Theorem 4.10 guarantees that a cluster with conductance at most ϕ can be identified.

Of course Nibble cannot afford to work with precise random-walk distributions $(\mathbf{q}^{(1)}, ..., \mathbf{q}^{(T)})$. Most of the technical analyses of [318] are for bounding the impact of the truncation.

4.4 Scalable Local Computation of Personalized PageRank

Local approximation of personalized PageRank is a basic problem in network analysis. For example, as shown in Chapter 3, SIGNIFICANT-PAGERANK IDENTIFICATION can be reduced to the problem of approximating personalized PageRank. Scalable local personalized PageRank approximation also leads to a scalable local clustering algorithm [22].

In this section, we will discuss the elegant construction of Jeh-Widom [177] and Andersen-Chung-Lang [22] for locally approximating personalized PageRank. The goal of these algorithms — when given a node v of a graph $G = (V, E, \mathbf{W})$ and a precision parameter ϵ — is to compute a sparse vector $\widetilde{\mathbf{p}}_u$ with $O(1/\epsilon)$ non-zero entries such that:

$$\forall v \in V, \quad \mathbf{p}_v[v] - \epsilon \leq \widetilde{\mathbf{p}}_v[v] \leq \mathbf{p}_v[v].$$

Recall from Eqn. (3.3) that the (personalized) PageRank are governed by the following linear equation:

$$\mathbf{p}_{\mathbf{s}} = \alpha \cdot \mathbf{s} + (1 - \alpha) \cdot \mathbf{M} \cdot \mathbf{p}_{\mathbf{s}} \tag{4.19}$$

where $\mathbf{s}$ is any *starting distribution*, and $\mathbf{M} = \mathbf{W}^T \cdot (\mathbf{D}^{out})^{-1}$. When $\mathbf{s} = \mathbf{1}_u$, we have $\mathbf{p}_{\mathbf{s}} = \mathbf{p}_u$. When $\mathbf{s} = \mathbf{1}$, we have $\mathbf{p}_{\mathbf{s}} = \mathbf{PageRank}_{\mathbf{W},\alpha}$.

Eqn. (4.19) can be solved by the following power series:

$$\mathbf{p}_{\mathbf{s}} = \alpha \cdot \mathbf{s} + \alpha \cdot \sum_{t=1}^{\infty} (1 - \alpha)^t \cdot \mathbf{M}^t \cdot \mathbf{s} \tag{4.20}$$

which implies $\mathbf{p}_{(\mathbf{M}\cdot\mathbf{s})} = \mathbf{M}\cdot\mathbf{p}_\mathbf{s}$. This power series leads to the following key equation for taking *local* steps in order to approximate personalized PageRank:

$$\mathbf{p}_\mathbf{s} = \alpha\cdot\mathbf{s} + (1-\alpha)\mathbf{M}\mathbf{p}_\mathbf{s} = \alpha\cdot\mathbf{s} + (1-\alpha)\mathbf{p}_{(\mathbf{M}\cdot\mathbf{s})} \tag{4.21}$$

For a given $\mathbf{s}$, we can then iteratively update two non-negative vectors $(\widetilde{\mathbf{p}}, \mathbf{r})$, which initially are $\widetilde{\mathbf{p}} = \mathbf{0}$ and $\mathbf{r} = \mathbf{s}$, and maintain the invariant:

$$\widetilde{\mathbf{p}} = \mathbf{p}_{(\mathbf{s}-\mathbf{r})}.$$

The idea behind the algorithms of Jeh-Widom and Andersen-Chung-Lang is the following: We can repeatedly use a local operation to "move" probabilities from $\mathbf{r}$ to $\widetilde{\mathbf{p}}$ to eventually reduce the 1-norm of $\mathbf{r}$ to below ϵ. This operation is called $\mathsf{Push}(v)$, which locally updates $\widetilde{\mathbf{p}}$ and $\mathbf{r}$. $\mathsf{Push}(v)$ only involves the entries associated with v and its neighbors $N^{out}(v) = \{w : (v,w) \in E\}$.

Algorithm: $\mathsf{Push}(G, \alpha, \widetilde{\mathbf{p}}, \mathbf{r}, v,)$

1: Set $\widetilde{\mathbf{p}}[v] = \widetilde{\mathbf{p}}[v] + \alpha\cdot\mathbf{r}[v]$; set $\mathbf{r}[v] = (1-\alpha)\cdot\mathbf{M}[v,v]\cdot\mathbf{r}[v]$.
2: For all $w \in N^{out}(v)$, set $\mathbf{r}[w] = \mathbf{r}[w] + (1-\alpha)\cdot\mathbf{M}[w,v]\cdot\mathbf{r}[v]$.
3: Return $(\widetilde{\mathbf{p}}, \mathbf{r})$.

Proposition 4.13 (Push)**.** $\mathsf{Push}(G, \alpha, \widetilde{\mathbf{p}}, \mathbf{r}, v)$ preserves the invariant $\widetilde{\mathbf{p}} = \mathbf{p}_{(\mathbf{s}-\mathbf{r})}$. Moreover, the operation is local to v and runs in $O(|N^{out}(v)|)$ time. The Push moves an amount of $\alpha\cdot\mathbf{r}[v]$ from $\mathbf{r}$ to $\widetilde{\mathbf{p}}$.

Proof. Let $\Delta\widetilde{\mathbf{p}}$ and $\Delta\mathbf{r}$ denote the changes in $\widetilde{\mathbf{p}}$ and $\mathbf{s}-\mathbf{r}$, respectively, after the Push operation. We have:

$$\Delta\widetilde{\mathbf{p}} = (\alpha\cdot\mathbf{r}[v])\cdot\mathbf{1}_v \text{ and } \Delta\mathbf{r} = \mathbf{r}[v]\cdot\mathbf{1}_v - (1-\alpha)\mathbf{r}[v]\cdot\mathbf{M}\cdot\mathbf{1}_v.$$

Consequently, by substituting $(\Delta\widetilde{\mathbf{p}}, \Delta\mathbf{r})$ on the left-hand side of the following equation, we have:

$$\alpha\cdot\Delta\mathbf{r} + (1-\alpha)\cdot\mathbf{M}\cdot\Delta\widetilde{\mathbf{p}} = (\alpha\cdot\mathbf{r}[v])\cdot\mathbf{1}_v = \Delta\widetilde{\mathbf{p}}.$$

By Eqn. (4.19), $\Delta\widetilde{\mathbf{p}} = \mathbf{p}_{(\Delta\mathbf{r})}$, and hence $\mathsf{Push}(G, \alpha, \widetilde{\mathbf{p}}, \mathbf{r}, v)$ preserves the invariant $\widetilde{\mathbf{p}} = \mathbf{p}_{(\mathbf{s}-\mathbf{r})}$. □

The algorithms of [177, 22] repeatedly apply the following: If there is a node $v \in V$ with a large enough entry in $\mathbf{r}$, then apply Push to v.

Algorithm: PushBasedPPRApproximation(G, α, ϵ, u)

Set $\widetilde{\mathbf{p}} = \mathbf{0}$ and set $\mathbf{r} = \mathbf{1}_u$.
while $\exists v \in V$ such that $\mathbf{r}[v] \geq \epsilon$ **do**
    $(\widetilde{\mathbf{p}}, \mathbf{r}) = \mathsf{Push}(G, \alpha, \widetilde{\mathbf{p}}, \mathbf{r}, v)$
Return $(\widetilde{\mathbf{p}}, \mathbf{r})$.

When the approximation quality of this algorithm is examined via *backward-error analysis*, its performance becomes more transparent.

Definition 4.14 (Approximate PPR with ϵ-Backward Errors)**.** $\widetilde{\mathbf{p}}$ is an approximation of $\mathbf{p_s}$ with ϵ-backward error if there exists a non-negative vector $\mathbf{r}$ such that $\widetilde{\mathbf{p}} = \mathbf{p}_{(\mathbf{s}-\mathbf{r})}$ and $||\mathbf{r}||_{\max} \leq \epsilon$.

We now analyze the quality and sparsity of the approximation.

Theorem 4.15. Suppose $u \in V$ is a node in $G = (V, E, \mathbf{W})$. For any $\alpha, \epsilon \in (0, 1)$, $(\widetilde{\mathbf{p}}, \mathbf{r}) = \mathsf{PushBasedPPRApproximation}(G, \alpha, \epsilon, u)$ terminates after making at most $\frac{1}{\alpha\epsilon}$ calls to Push(), to obtain an approximation $\widetilde{\mathbf{p}}$ of personalized PageRank $\mathbf{p}_{\mathbf{1}_u}$ with ϵ-backward error. In other words, $\widetilde{\mathbf{p}} = \mathbf{p}_{(\mathbf{1}_u - \mathbf{r})}$ and $||\mathbf{r}||_{\max} \leq \epsilon$. Moreover, $\mathrm{nnz}(\widetilde{\mathbf{p}}) \leq \frac{1}{\alpha\epsilon}$.

Proof. Each call to Push increases $||\widetilde{\mathbf{p}}||_1$ by at least $\alpha\epsilon$. Because $||\widetilde{\mathbf{p}}||_1 \leq 1$ throughout the algorithm, Push is called at most $\frac{1}{\alpha\epsilon}$ time, and hence $\mathrm{nnz}(\widetilde{\mathbf{p}}) \leq \frac{1}{\alpha\epsilon}$. By the condition of the **while** loop, when the algorithm terminates, every entry in $\mathbf{r}$ is at most ϵ, so $||\mathbf{r}||_{\max} \leq \epsilon$. □

The running time of $\mathsf{Push}(G, \alpha, \widetilde{\mathbf{p}}, \mathbf{r}, v)$ is $\Theta(|N^{out}(v)|)$. The time complexity of $\mathsf{PushBasedPPRApproximation}(G, \alpha, \epsilon, u)$ may be much more expensive than $O(\frac{1}{\alpha\epsilon})$, when the maximum out-degree is large.

In the context of designing local clustering algorithm for undirected and unweighted networks, Andersen, Chung, and Lang [22] introduced

the following slightly weaker notion of PPR approximation: $\widetilde{\mathbf{p}}$ is an approximation of $\mathbf{p_s}$ with *ϵ-normalized backward error* if there exists non-negative vector $\mathbf{r}$ such that (i) $\widetilde{\mathbf{p}} = \mathbf{p}_{(\mathbf{s}-\mathbf{r})}$ and (ii) $\forall v \in V$, $\mathbf{r}[v] \leq \epsilon \cdot d_v$, where $d_v = |N(v)|$ is the degree of v in the unweighted graph G. Such an approximation can be computed by the following natural variation of PushedBasedPPRApproximation:

Algorithm: ACL-PushBasedPPRApproximation$(G, \alpha, \epsilon, \mathbf{s})$

1: Set $\widetilde{\mathbf{p}} = \mathbf{0}$ and set $\mathbf{r} = \mathbf{s}$.
2: **while** $\exists v \in V$ such that $\mathbf{r}[v] \geq \epsilon \cdot d_v$ **do**
3: $\quad (\widetilde{\mathbf{p}}, \mathbf{r}) = \mathsf{Push}(G, \alpha, \widetilde{\mathbf{p}}, \mathbf{r}, v)$
4: Return $(\widetilde{\mathbf{p}}, \mathbf{r})$.

Andersen, Chung, and Lang [22] gave an elegant proof showing that ACL-PushBasedPPRApproximation is a scalable local algorithm for ϵ-normalized backward errors.

Theorem 4.16. Suppose $G = (V, E)$ is an undirected, unweighted graph and $\mathbf{s}$ is a probability vector. For any $\alpha, \epsilon \in (0, 1)$, $(\widetilde{\mathbf{p}}, \mathbf{r}) =$ ACL-PushBasedPPRApproximation$(G, \alpha, \epsilon, \mathbf{s})$ runs in time $O(\frac{1}{\alpha\epsilon})$ to obtain an approximation $\widetilde{\mathbf{p}}$ of personalized PageRank $\mathbf{p_s}$ with ϵ-normalized backward error. Moreover, $\mathrm{nnz}(\widetilde{\mathbf{p}}) \leq \mathrm{vol}_{\mathbf{W}}(\mathrm{support}(\widetilde{\mathbf{p}})) \leq O\left(\frac{1}{(1-\alpha)\epsilon}\right)$.

Proof. Suppose the algorithm terminates after T steps. For $t \in [T]$, let v_t be the node Pushed at step t, which increases $||\widetilde{\mathbf{p}}||_1$ by at least $\alpha \cdot (\epsilon d_{v_t})$. After T steps, $\sum_{t=1}^{T} \alpha \cdot (\epsilon d_{v_t}) \leq ||\widetilde{\mathbf{p}}||_1 \leq 1$, which implies:

$$\sum_{t=1}^{T} d_{v_t} \leq \frac{1}{\alpha\epsilon} \tag{4.22}$$

Because the t^{th} Push takes $O(|N(v_t)|) = O(d_{v_t})$ time, the complexity of ACL-PushBasedPPRApproximation$(G, \alpha, \epsilon, \mathbf{s})$ is:

$$O\left(\sum_{t=1}^{T} d_{v_t}\right) = O\left(\frac{1}{\alpha\epsilon}\right).$$

We also have, $\text{vol}_{\mathbf{W}}(\text{support}(\widetilde{\mathbf{p}})) \leq \sum_t^T d_{v_t} \leq \frac{1}{\alpha\epsilon}$. The tighter bound of $\text{vol}_{\mathbf{W}}(\text{support}(\widetilde{\mathbf{p}}))$ stated in the Theorem follows from the observation that when a node is last Pushed, $(1-\alpha)\epsilon d_{v_t}$ amount of probability is retained by $\mathbf{r}$, and $||\mathbf{r}||_1 \leq 1$ throughout the algorithm. □

For unweighted networks, Theorem 4.16 neatly transfers the dependence on vertex degrees from running-time to approximation accuracy. However, the linear dependence on the maximum vertex degree in some form seems unavoidable. It remains open if one can locally approximate personalized PageRank in time $\widetilde{O}_n(1) \cdot \frac{1}{\epsilon}$ either for achieving backward error of ϵ, or for ensuring $\mathbf{p}_u[v] - \epsilon \leq \widetilde{\mathbf{p}}_u[v] \leq \mathbf{p}_u[v]$, as originally desired in PPRApproximation (Problem 4.5).

Needed for Significant-PageRank Identification, Borgs *et al.* [65] gave an $\widetilde{O}_n(1) \cdot \frac{1}{\epsilon \cdot \rho^2}$ time algorithm for the following relaxation of PPRApproximation.

Problem 4.17 (BicriteriaPPRApproximation)**.** Given a node u in a network $G = (V, E, \mathbf{W})$ and parameters $\epsilon, \rho \in (0, 1)$, compute a sparse vector $\widetilde{\mathbf{p}}_u$ with $O(1/\epsilon)$ non-zero entries (with the factor in the big-O notation depending polynomially on $1/\rho$) such that:

$$(1-\rho) \cdot \mathbf{p}_u[v] - \epsilon \leq \widetilde{\mathbf{p}}_u[v] \leq (1+\rho) \cdot \mathbf{p}_u[v] + \epsilon, \ \forall v \in V.$$

The algorithm of [65] uses the following probabilistic view of personalized PageRank. Recall $\mathbf{p}_u$ is the stationary distribution of the Markov process starting at u that either (1) takes a random-walk step (with probability $(1-\alpha)$), or (2) restarts at u (with probability α). Let a *run* of this process be a random walk between two consecutive restarts.

Proposition 4.18 (Probabilistic View of PPR)**.** For any $u, v \in V$, $\mathbf{p}_u[v] = \mathbf{PPR}_{\mathbf{W},\alpha}[u, v]$ is equal to the probability that a run of random walk starting at u passes by v immediately before it restarts.

Proof. It follows directly from Eqn. (4.25). □

Based on this view the following local algorithm uses "short" runs of random walks — starting at u — to sample this probability distribution. By short runs, we mean that random-walks are limited to $T = \log_{1/(1-\alpha)} \frac{2}{\alpha\epsilon}$ steps. For a confidence parameter δ, the algorithm makes $R = \frac{8}{\epsilon \cdot \rho^2} \cdot \ln(n/\delta)$ number of runs to obtain a robust estimate of $\mathbf{p}_v$. Any run that tries go beyond T steps will be ignored.

Algorithm: BicriteriaPPRApproximation$(G, \alpha, \epsilon, \rho, \delta, u)$

1: Set $T = \log_{1/(1-\alpha)} \frac{2}{\alpha\epsilon}$ and $R = \frac{8}{\epsilon \cdot \rho^2} \cdot \ln(n/\delta)$.
2: Set $\widetilde{\mathbf{p}} = \mathbf{0}$.
3: **for** $h = 1 : R$ **do**
4: Take a run of random walks up to T steps, and v was the last node it visits before the restart, then set $\widetilde{\mathbf{p}}[v] = \widetilde{\mathbf{p}}[v] + 1/R$.
5: Return $\widetilde{\mathbf{p}}$.

Theorem 4.19. For any network $G = (V, E, \mathbf{W})$ with n nodes, and $\epsilon, \rho, \delta \in (0, 1)$, with probability at least $1-\delta$ and in time $\widetilde{O}_n(1) \cdot \frac{1}{\epsilon \cdot \rho^2} \cdot \log \frac{1}{\epsilon}$ BicriteriaPPRApproximation$(G, \alpha, \epsilon, \rho, \delta, u)$ computes a sparse vector $\widetilde{\mathbf{p}}_u$ such that $\mathrm{nnz}(\widetilde{\mathbf{p}}_u) \leq \frac{8}{\epsilon \cdot \rho^2} \cdot \ln(n/\delta)$, and:

$$\forall v \in V, \quad (1-\rho) \cdot \mathbf{p}_u[v] - \epsilon \leq \widetilde{\mathbf{p}}_u[v] \leq (1+\rho) \cdot \mathbf{p}_u[v] + \epsilon \qquad (4.23)$$

Proof. By Proposition 4.18, Algorithm BicriteriaPPRApproximation introduces two types of errors:

Truncation Error: When the length of the random walk is limited to T, some loss of probability is introduced into the algorithm. But, the loss at each node is bounded by $\sum_{t=1}^{T} (1-\alpha)^t \leq (1-\alpha)^T/\alpha = \epsilon/2$.

Sampling Error: To bound the error of the second type, we classify $v \in V$ based on whether or not $\mathbf{p}_u[v] \geq \epsilon$:

1. If $\mathbf{p}_u[v] \geq \epsilon$, then the expected number of runs of random-walks that visit v before restart is at most $\mathbf{p}_u[v] \cdot R$ and at least:

$$(\mathbf{p}_u[v] - \epsilon/2) \cdot R \geq (\epsilon/2) \cdot R = \frac{4}{\rho^2} \cdot \ln(n/\delta)$$

The standard (multiplicative) Chernoff-Hoeffding bound implies that with probability at least $1 - \delta/n$:

$$(1 - \rho) \cdot (\mathbf{p}_u[v] - \epsilon/2) \leq \widetilde{\mathbf{p}}_u[v] \leq (1 + \rho) \cdot \mathbf{p}_u[v].$$

2. If $\mathbf{p}_u[v] < \epsilon$, the expected number of runs that visit u before restart is at most:

$$\mathbf{p}_u[v] \cdot R \leq \epsilon \cdot R = \frac{8}{\rho^2} \cdot \ln(n/\delta).$$

Thus, with probability at least $1 - \delta/n$:

$$\mathbf{p}_u[v] - \epsilon \leq 0 \leq \widetilde{\mathbf{p}}_u[v] \leq (1 + \rho) \cdot \epsilon \leq (1 + \rho)\mathbf{p}_u[v] + \epsilon.$$

The guarantee given by Eqn. (4.23) of the theorem then follows from the union bound. □

4.5 Discussions: Local Exploration of Networks

The task of a scalable local algorithm is to explore a *hidden* network. However, its objective is not just to collect information about networks. For most applications, the goal of the local algorithm is to efficiently identify a substructure that has a target property — such as a good cluster — before exploring the entire network. Therefore, the design must address the following key algorithmic question:

> *In which order should the local algorithm explore the nodes and edges of a network?*

The two most commonly used graph exploration techniques are breadth-first search (BFS) and depth-first search (DFS). However, the strength of each in the traditional exploration setting is also its limitation in the local exploration setting.

For example, one expects most social networks to have low diameters. Thus, running BFS on such a network quickly exposes too many nodes, which could either force the BFS to run out of time, or force it to make complex ordering decisions of nodes in the same level set. For such networks, BFS level sets are ineffective guides for "vicinity."

With weighted networks, BSF will naturally explore nodes in the order of "shortest-path" distances from the starting node. Such an ordering is likely ineffective in low-diameters networks.

In contrast, DFS usually maintains a much smaller set of "exposed" nodes than BFS, which is a needed property for scalable local exploration. With a backtracking mechanism, the search can travel far away from the starting node, and then return to its vicinity. However, the long-range search could be a distraction from efficient local network exploration.

The local clustering algorithm Nibble of [318] uses an exploration strategy that is based on the following intuition: The nodes in the target cluster are most likely to appear in short random walks from the starting node. Therefore, this likelihood could be used to design a guide for vicinity: The more likely that a node v appears in a short random walk from starting node u, the *nearer* v is to u. This vicinity guide could be effective for defining "neighborhoods" or "candidate clusters" — such as in Sweep — near a starting node.

Algorithmically, a local algorithm faces more challenges because it does not have any information of this vicinity guide in advance. It needs to efficiently build one as part of its local exploration.

- ***Random-Walk Distributions***: Nibble adaptively uses the scaled lazy random-walk distributions as vicinity guides. By varying the length of the random walks, Nibble constructs several approximated guides: For $t = 1 : \widetilde{O}_n(1)$, Nibble orders nodes of the network from "near" to "far" by sorting an approximation of $\mathbf{D}_{\mathbf{W}}^{-1} \cdot \mathbf{M}^t \cdot \mathbf{1}_u$ for the t^{th}-step exploration, and uses this ordering to identify nodes in the local neighborhoods of u as candidates for the local cluster.

- ***Personalized PageRank***: PPR-Nibble uses an approximation of (the scaled) personalized PageRank, $\mathbf{D}_{\mathbf{W}}^{-1}\mathbf{p}_u$, as the vicinity guide for local exploration and the formation of candidate clusters near a starting node u. Thus, instead of using multiple vicinity guides as in Nibble, PPR-Nibble uses a single guide to identify local clusters.

Mathematically, the local algorithm should use a vicinity guide that is provably effective for a particular network-analysis task. Because the local algorithm usually applies approximation to achieve efficiency, the mathematical analysis of local algorithms usually consists of the following two steps:

- **Effectiveness**: An effective vicinity guide exists.
- **Robustness**: The approximate vicinity guide computed by the local algorithm remains sufficiently effective.

For Nibble, Lovász-Simonovits' theorem [239, 240] provides the main tool for the effectiveness analysis. This theorem establishes the effectiveness of Sweep for identifying clusters from the distributions of short random-walks. For PPR-Nibble, Andersen, Chung, and Lang [22] extended Lovász-Simonovits' theorem to personalized PageRank.

The robustness analysis of Nibble relies on the fact that the truncated distributions remain sufficiently accurate for effective Sweep [318]. The robustness of PPR-Nibble is established by a similar analysis [22].

Sampling vs Matrix Powering

We conclude this section with a brief discussion of random-walk approximation. Two basic techniques for building a local vicinity guide are sampling and matrix power. The sampling approach — as in BicriteriaPPRApproximation — uses short random-walks to gather statistical information of random-walk distributions. Its main advantage is that short random walks can pass through high-degree nodes efficiently. However, sampling introduces some additional errors.

The matrix-power approach, as in Nibble and PushBasedPPRApproximation, calculates the random-walk distribution by following the transition matrix of the random walk. To be efficient, the algorithm needs to determine when to ignore small entries, and in which order to emulate the power of the matrix. While mathematically more appealing, this approach could be limited by the high computation cost around large-degree nodes.

4.6 Interplay Between Dynamic Processes and Networks

What network interaction model do we assume when using conductance as the measure of clusterability? What is the impact of other dynamic processes on network centrality and clusterability?

Mathematically, conductance is related with the random-walk Markov process over a network $G = (V, E, \mathbf{W})$. For any $S \subseteq V$, suppose this Markov process starts at a random node in S, drawn according to nodes' stationary probability. Then, $\text{conductance}_{\mathbf{W}}(S)$ is equal to the probability that a single random-walk step leaves S. Thus, when choosing conductance as the measure of clusterability, one may implicitly assume that a standard random walk is taking place on the network, which induces interactions among nodes.

It is conceivable that other dynamic processes on a network, such as the spread of information, ideas, or epidemics [114, 286, 201], define different interactions among nodes [146, 84]. A random-walk goes from a node to one of its neighbors. In contrast, the spread of epidemics or information is a stochastic process that may infect or influence multiple neighbors of the node [114, 286, 201]. Consequently, for a group of nodes in the network, the dynamic process may impact its clusterability. The network process may also impact the centrality of nodes. However, it remains challenging to characterize the impact that these complex dynamic processes have on centrality and clusterability [146, 84].

Below, we discuss a framework of [146] for studying the *interplay* between a dynamic process and the graph structure on which this process unfolds. Ghosh *et al.* [146] consider a simple family of dynamic processes, which parameterizes the random-walk Markov process. Together, the specifics of the dynamic process and "static" network data $\mathbf{W}$ are then used to define a family of clusterability measures. This family extends the traditional conductance measure. Ghosh *et al.*'s work provides a method to quantify the impact of this interplay on network centrality and clusterability.

In Section 8.1, we will discuss another recent work [84], which focuses on the interplay between social-influence and network centrality.

A Family of Graphical Dynamic Models

In a dynamic process over a network $G = (V, E, \mathbf{W})$, we use a temporal variable $\theta_u^{(t)}$ to capture the level of participation of a node $u \in V$ in the dynamic process. From a starting vector $\boldsymbol{\theta}^{(0)}$, the evolution of the dynamic process is specified by a temporal vector:

$$\boldsymbol{\theta}^{(t)} = \left(\theta_1^{(t)}, ..., \theta_n^{(t)}\right).$$

In the *graphical dynamic model*, we further assume that the evolution of $\theta_u^{(t)}$ depends on its own state and the states of its network neighbors:

$$N(u) = \{v : (u, v) \in E\}.$$

In other words, its *derivative* $\frac{d\theta_u}{dt}$ depends $\{\theta_v^{(t)} : v \in N(u)\} \cup \{\theta_u^{(t)}\}$. These dependencies capture the underlying interactions among nodes in the network. In the simplest form of the *graphical dynamic model*, the interactions among nodes are assumed to be *linear*. In a *continuous time* framework, the following partial differential equation defines the evolution of the dynamic process:

$$\frac{d\boldsymbol{\theta}}{dt} = -\mathcal{L} \cdot \boldsymbol{\theta} \tag{4.24}$$

The linear operator $\mathcal{L}$ specifies the level of influence among network neighbors during the dynamic process. Analytically, one can express the evolution of a linear graphical dynamic process by a power series:

$$\boldsymbol{\theta}^{(t)} = e^{-\mathcal{L}t} \cdot \boldsymbol{\theta}^{(0)} = \sum_{k=0}^{\infty} \frac{(-t)^k}{k!} \cdot \mathcal{L}^k \cdot \boldsymbol{\theta}^{(0)} \tag{4.25}$$

Some readers may find it more intuitive when t takes *discrete integer values*, i.e. $t = 0, 1, 2, \cdots$ and consider Eqn. (4.24) in its first-order difference form: $\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)} = -\mathcal{L} \cdot \boldsymbol{\theta}^{(t)}$. This iterative form of this discrete dynamic process is then: $\boldsymbol{\theta}^{(t+1)} = (\mathbf{I} - \mathcal{L}) \cdot \boldsymbol{\theta}^{(t)}$. We will refer to $\mathbf{M} = (\mathbf{I} - \mathcal{L})$ as the *interaction matrix*. For example, when $\mathcal{L} = (\mathbf{D_W} - \mathbf{W}) \cdot \mathbf{D_W^{-1}}$, the discrete dynamic process becomes:

$$\boldsymbol{\theta}^{(t+1)} = (\mathbf{I} - \mathcal{L}) \cdot \boldsymbol{\theta}^{(t)} = \left(\mathbf{W} \cdot \mathbf{D_W^{-1}}\right) \cdot \boldsymbol{\theta}^{(t)} \tag{4.26}$$

where the interaction matrix $\mathbf{M_W} = \mathbf{W} \cdot \mathbf{D_W^{-1}}$ is the *transition matrix* that models the discrete random walks on the network defined by $\mathbf{W}$.

The graphical dynamic models considered in [146] is linear. Each of its operators $\mathcal{L}$ incorporates some dynamics-specific parameters into the random-walk Markov process given by the static network data $\mathbf{W}$.

Definition 4.20 (Parameterized Laplacian Dynamics)**.** Suppose $G = (V, E, \mathbf{W})$ is a network. The operator $\mathcal{L}_{\mathbf{W},\mathbf{R},\mathbf{T}}$ is defined by two dynamics-specific parameters: $\mathbf{R} = \text{diag}([\rho_1, ..., \rho_n])$ and $\mathbf{T} = \text{diag}([\tau_1, ..., \tau_n])$, (where $\mathbf{T}$ is scaled: $\forall u \in V$, $\tau_u \geq 1$):

1. **Interaction Matrix**: $\mathbf{M}_{\bar{\mathbf{W}}} = \bar{\mathbf{W}} \cdot \mathbf{D}_{\bar{\mathbf{W}}}^{-1}$, where $\bar{\mathbf{W}} = \mathbf{R} \cdot \mathbf{W} \cdot \mathbf{R}$.
2. **Operator**:

$$\mathcal{L}_{\mathbf{W},\mathbf{R},\mathbf{T}} = (\mathbf{I} - \mathbf{M}_{\bar{\mathbf{W}}}) \cdot \mathbf{T}^{-1} \tag{4.27}$$

We first examine a few graphical dynamic processes that can be captured by the parameterized Laplacian. These examples also explain the role of parameters $\mathbf{R} = \text{diag}([\rho_1, ..., \rho_n])$ and $\mathbf{T} = \text{diag}([\tau_1, ..., \tau_n])$ in the dynamic processes.

Normalized Laplacians and Symmetrization of Random Walks: In the base case when $\mathbf{R} = \mathbf{T} = \mathbf{I}$:

$$\mathcal{L}_{\mathbf{W},\mathbf{I},\mathbf{I}} = (\mathbf{D}_{\mathbf{W}} - \mathbf{W}) \cdot \mathbf{D}_{\mathbf{W}}^{-1} = \mathbf{I} - \mathbf{W} \cdot \mathbf{D}_{\mathbf{W}}^{-1} \tag{4.28}$$

Here, $\mathbf{W} \cdot \mathbf{D}_{\mathbf{W}}^{-1}$ is the transition matrix of the standard random walks on G, and $\mathcal{L}_{\mathbf{W},\mathbf{I},\mathbf{I}}$ defines the dynamic process associated with random walks. Note that $\mathcal{L}_{\mathbf{W},\mathbf{I},\mathbf{I}}$ has two well-known similar forms:[2]

$$\mathbf{D}_{\mathbf{W}}^{-\frac{1}{2}} \mathcal{L}_{\mathbf{W},\mathbf{I},\mathbf{I}} \mathbf{D}_{\mathbf{W}}^{\frac{1}{2}} = \mathbf{I} - \mathbf{D}_{\mathbf{W}}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}_{\mathbf{W}}^{-\frac{1}{2}} \tag{4.29}$$

$$\mathbf{D}_{\mathbf{W}}^{-1} \mathcal{L}_{\mathbf{W},\mathbf{I},\mathbf{I}} \mathbf{D}_{\mathbf{W}} = \mathbf{I} - \mathbf{D}_{\mathbf{W}}^{-1} \mathbf{W} \tag{4.30}$$

This first matrix — known as the *normalized Laplacian* [95, 305, 185] — is the symmetrization of $\mathcal{L}_{\mathbf{W},\mathbf{I},\mathbf{I}}$ under similar transformation. This second matrix is associated with — as pointed out in [146] — the dynamic

[2] Two matrices $\mathbf{M}_1$ and $\mathbf{M}_2$ are *similar* if there exists a non-singular matrix $\mathbf{B}$ such that $\mathbf{M}_1 = \mathbf{B} \cdot \mathbf{M}_2 \cdot \mathbf{B}^{-1}$.

process called the *consensus process* on the network: The stationary distribution of any consensus process is the uniform distribution.

Note that eigenvalues are preserved under the similarity matrix transformation. Thus, although they appear to be different, linear dynamics with similar operators are essentially the same [146].

Semi-Markov Random Walks, Lazy Walks, and Diffusions: When $\mathbf{T}$ varies and $\mathbf{R} = \mathbf{I}$:

$$\mathcal{L}_{\mathbf{W},\mathbf{I},\mathbf{T}} = (\mathbf{D}_{\mathbf{W}} - \mathbf{W}) \cdot \mathbf{D}_{\mathbf{W}}^{-1} \cdot \mathbf{T}^{-1}.$$

With this operator, Eqn. (4.25) defines a process, known as the *continuous time Markov process*. Although the "holding times" $[\tau_1, ..., \tau_n]$ are no longer uniform, this exponential "Brownian" dynamics process remains Markovian.

There are two different discrete-time processes related with this model:

- *Semi-Markov random walks*: To provide some intuition about this dynamic process, consider the following scenario. Each node u in the network $G = (V, E, \mathbf{W})$ has its own "entropy" [76], η_u, which requires a prudent meanderer to pause for $\tau_u = g(\eta_u)$ of time to process the neighborhood of u before taking a "fair" random step. Here $g(\eta)$ denotes the amount of time for the meanderer to understand a site with entropy η.

- *Lazy Walks*: At $v \in V$, with probability $1 - \frac{1}{\tau_v}$, the meanderer stays put at v, and with probability $\frac{1}{\tau_v}$, the meanderer takes a random walk from v.

In both discrete processes, "holding times" in $\mathbf{T}$ play roles that impact the "realization" of a path. Although they appear to be similar, these two discrete processes have subtle differences. For example, unlike semi-random walks, lazy walks are always Markovian.

The parameterization of $\mathbf{T}$ enables us to study various forms of random-walks under a single network. For example, when τ_u is inversely proportional to d_u, i.e., $\mathbf{T} = d_{\max} \cdot \mathbf{D}_{\mathbf{W}}^{-1}$:

$$\mathcal{L}_{\mathbf{W},\mathbf{I},\mathbf{T}} = (\mathbf{D}_{\mathbf{W}} - \mathbf{W}) \cdot \mathbf{D}_{\mathbf{W}}^{-1} \cdot \mathbf{T}^{-1} = \frac{1}{d_{\max}} \cdot (\mathbf{D}_{\mathbf{W}} - \mathbf{W}).$$

Thus, the *Laplacian matrix* $\mathbf{L_W} = (\mathbf{D_W} - \mathbf{W})$ is intrinsically connected with the operator of this dynamic process, which is known as the (heat) *diffusion process* [92].

Affinity Scaling — Biased Random Walks and Replicators: The symmetric diagonal scaling of $\mathbf{W}$ by $\mathbf{R}$, $\bar{\mathbf{W}} = \mathbf{R} \cdot \mathbf{W} \cdot \mathbf{R}$ changes the affinity weight between u and v from $w_{u,v}$ to $\bar{w}_{u,v} = (\rho_u \rho_v) \cdot w_{u,v}$, thus altering the interaction matrix and the distribution of random walks. The data in $\mathbf{R}$ provides a parameterization for modeling a richer family of interactions, whose intensity depends both on the topology of the underlying network and the features of nodes.

For example, suppose Facebook had decided to display each user's PageRank, which in turn spurred the degree of interaction among "friends" to be proportional to the product of their PageRank values (i.e., popular friends are more likely to interact!). Then, the resulting interaction can be modeled by a new matrix $\bar{\mathbf{W}} = \mathbf{R} \cdot \mathbf{W} \cdot \mathbf{R}$, where ρ_u is the PageRank of u. As another example, suppose due to the attention span, interaction efficiency between each pair of friends u and v is inversely proportional to $\sqrt{d_u d_v}$. In this case, $\rho_u = d_u^{-\frac{1}{2}}$, and the interaction matrix becomes:

$$\bar{\mathbf{W}} = \mathbf{D}_{\mathbf{W}}^{-\frac{1}{2}} \cdot \mathbf{W} \cdot \mathbf{D}_{\mathbf{W}}^{-\frac{1}{2}}.$$

In the literature, the matrix $\mathbf{D}_{\mathbf{W}}^{-\frac{1}{2}} \cdot \mathbf{W} \cdot \mathbf{D}_{\mathbf{W}}^{-\frac{1}{2}}$ is sometimes referred to as the *unbiased affinity matrix* of G, and its Laplacian is referred to as the *unbiased Laplacian* of G.

Another example is to regulate the interaction matrix of $\mathbf{W}$ by its top eigenvalue vector. Suppose $\lambda_{\max}$ is the largest eigenvalue of $\mathbf{W}$ and $\mathbf{Wv} = \lambda_{\max} \cdot \mathbf{v}$. Let $\mathbf{R} = \mathrm{diag}(\mathbf{v})$. Then, as observed in [308, 146], the symmetrization of $\mathcal{L}_{\mathbf{W},\mathbf{R},\mathbf{I}}$ is $\mathbf{I} - \frac{1}{\lambda_{\max}}\mathbf{W}$. This operator is known as the *replicator matrix* for modeling the growth of some epidemics [145].

The dynamics-specific parameter $\mathbf{R}$ also enables us to combine information from different networks. For example, suppose V denotes the researchers in theoretical computer science, and $G_{TCS} = (V, E, \mathbf{W})$ is a network where $w_{u,v}$ denotes the number of times u's work cites v's work. Suppose $V_S \subset V$ is a group of lucky researchers in V invited to attend programs at *the Simons Institute for the Theory of Computing*

(http://simons.berkeley.edu) in 2016. Suppose $G_S = (V_S, E_S, \mathbf{W}_S)$ is the graph where $\mathbf{W}_S[u, v]$ represents the number of days that $u, v \in V_S$ attend the same venues at Simons Institute. If the chance of interaction among these TCS researchers during the Simons' programs is proportional to the "significance" of their work, then one may approximately model this interaction by first computing for each $u \in V_S$, the PageRank p_u of u in the citation network G_{TCS}, and then defining the interaction matrix as $\bar{\mathbf{W}}_S = \mathbf{R} \cdot \mathbf{W}_S \cdot \mathbf{R}$, where $\rho_u = p_u$.

Parameterized Conductances

Although the parameterized dynamic process as defined by $\mathcal{L}_{\mathbf{W},\mathbf{R},\mathbf{T}}$ remains a Markov process, the separation of *static network data* $\mathbf{W}$ and the *dynamics-specific parameters* $(\mathbf{R}, \mathbf{T})$) helps to highlight the interplay between them. The work of [146] has inspired generalization to a broader family of dynamic processes, that includes viral marketing and social influence [84]. For clusterability, the framework of Ghosh *et al.* leads to a family of parameterized conductances [146], which incorporates the impact of $\mathbf{R}$ and $\mathbf{T}$. Let $\bar{\mathbf{W}} = \mathbf{R} \cdot \mathbf{W} \cdot \mathbf{R}$, and $\bar{d}_u = \sum_{v \in V} \bar{w}_{u,v}$ be the degree of u in network $\bar{G} = (V, E, \bar{\mathbf{W}})$. Let:

$$\text{cut}_{(\mathbf{W},\mathbf{R},\mathbf{T})}(S, \bar{S}) = \text{cut}_{\bar{\mathbf{W}}}(S, \bar{S}) = \sum_{u \in S, v \notin S} \bar{w}_{u,v} \tag{4.31}$$

$$\text{vol}_{(\mathbf{W},\mathbf{R},\mathbf{T})}(S) = \sum_{u \in S} \bar{d}_u \tau_u = \sum_{u \in S} \left(\sum_{v \in V} \bar{w}_{u,v} \right) \tau_u \tag{4.32}$$

The factor of τ_u in Eqn. (4.32) reflects the temporal impact to the connections involving S: the larger the delay factor τ_u is, the more present the continuous time Markov process is at node u.

Definition 4.21 (Parameterized Conductance)**.** For an operator $\mathcal{L}_{\mathbf{W},\mathbf{R},\mathbf{T}}$ with parameters $\mathbf{R}, \mathbf{T}$ and a network $G = (V, E, \mathbf{W})$:

$$\text{conductance}_{(\mathbf{W},\mathbf{R},\mathbf{T})}(S) = \frac{\text{cut}_{\mathbf{W},\mathbf{R},\mathbf{T}}(S, \bar{S})}{\min\left(\text{vol}_{(\mathbf{W},\mathbf{R},\mathbf{T})}(S), \text{vol}_{(\mathbf{W},\mathbf{R},\mathbf{T})}(\bar{S})\right)}.$$

Interpolating Two Popular Clusterability Measures

The two most commonly used clusterability measures in theoretical computer science are conductance (Eqn. (4.3)) and cut-ratio [12, 315]:

$$\text{cut-ratio}_{\mathbf{W}}(S) = \frac{\text{cut}_{\mathbf{W}}(S, \bar{S})}{\min(|S|, |\bar{S}|)} \tag{4.33}$$

These two clusterability measures are correlated. But clearly, they are different measures of clusterability, except when every node in V has the same degree, in which case, $\text{conductance}_{\mathbf{W}}(S)$ is proportional to $\text{cut-ratio}_{\mathbf{W}}(S)$, as $\text{vol}_{\mathbf{W}}(S)$ is proportional to $|S|$.

> *When I taught spectral graph theory, students often asked me about the relation between these two popular clusterability measures, particularly about their "spectral-theoretical" differences. For a long time, I could only diplomatically say that "they are related graph-theoretical quantities."*

Now I will say that the parameterized conductances smoothly interpolate between them!

Proposition 4.22 (Interpolating Conductance and Cut-Ratio)**.** For any network $G = (V, E, \mathbf{W})$ and $S \subset V$:

$$\begin{aligned} \text{conductance}_{\mathbf{W}}(S) &= \text{conductance}_{\mathbf{W},\mathbf{I},\mathbf{I}}(S). \\ \text{cut-ratio}_{\mathbf{W}} &= \frac{1}{d_{\max}} \cdot \text{conductance}_{\mathbf{W},\mathbf{I},d_{max}\cdot\mathbf{D}_{\mathbf{W}}^{-1}}(S). \end{aligned}$$

Thus, conductance and cut-ratio reflect off a network under similar, but, nevertheless, two different dynamic processes. In the next section, we will discuss an extension of the classical Cheeger's inequality, which illustrates the connection between parameterized conductances and the spectral properties of their corresponding linear dynamic operators.

Parameterized Continuous-Time Kernel PageRank

Chung [92] defined a family of centrality ranking vectors that are analogous to personalized PageRanks. For any probability vector $\boldsymbol{\theta}$, the

heat-kernel PageRank of a *starting distribution* $\boldsymbol{\theta}$ at parameter $t \geq 0$ is given by:

$$\begin{aligned}\boldsymbol{\rho}_{\boldsymbol{\theta}}^{(t)} &:= e^{-(\mathbf{I}-\mathbf{M}_{\mathbf{W}})t} \cdot \boldsymbol{\theta} \\ &= \sum_{k=0}^{\infty} \frac{(-t)^k}{k!} \cdot (\mathbf{I} - \mathbf{M}_{\mathbf{W}})^k \cdot \boldsymbol{\theta} \\ &= e^{-t} \sum_{k=0}^{\infty} \frac{t^k}{k!} \cdot \mathbf{M}_{\mathbf{W}}^k \cdot \boldsymbol{\theta}.\end{aligned}$$

Note that this definition is based on Eqn. (4.25). In contrast, for a restart constant $\alpha > 0$, the personalized PageRank vector associated with a starting distribution $\boldsymbol{\theta}$ is the solution of:

$$\mathbf{p}_{\alpha,\boldsymbol{\theta}} = \alpha \cdot \boldsymbol{\theta} + (1-\alpha) \cdot \mathbf{M}_{\mathbf{W}} \cdot \ \mathbf{p}_{\alpha,\boldsymbol{\theta}} = \alpha \sum_{k=0}^{\infty} (1-\alpha)^k \cdot \mathbf{M}_{\mathbf{W}}^k \cdot \boldsymbol{\theta}.$$

In other words, the personalized PageRank uses a power series of geometric sum. In contrast, the heat-kernel PageRank takes an exponential sum.

Following Chung [92], we use Eqn. (4.25) to define a time-dependent numerical ranking vector, analogous to the personalized PageRank, to incorporate the dynamics-specific parameters.

Definition 4.23 (Continuous-Time Kernel PageRank)**.** The *continuous-time kernel PageRank* with respect to a starting distribution $\boldsymbol{\theta}$ and parameter $t \geq 1$ of $\mathcal{L}_{\mathbf{W},\mathbf{R},\mathbf{T}}$ is given by:

$$\mathbf{p}_{\mathbf{W},\mathbf{R},\mathbf{T},\boldsymbol{\theta}}^{(t)} = e^{-\mathcal{L}_{\mathbf{W},\mathbf{R},\mathbf{T}} \cdot t} \cdot \boldsymbol{\theta} = e^{-\mathbf{T}^{-1}t} \sum_{k=0}^{\infty} \frac{(-t)^k}{k!} \cdot \left(\mathbf{M}_{\bar{\mathbf{W}}} \mathbf{T}^{-1}\right)^k \cdot \boldsymbol{\theta}.$$

As in its (heat-kernel) PageRank counterparts, each entry in $\mathbf{p}_{\mathbf{W},\mathbf{R},\mathbf{T},\boldsymbol{\theta}}^{(t)}$ sums over paths through the underlying network.

To understand the contribution from network paths to ranking vectors, let's consider the case when $\mathbf{R} = \mathbf{I}$. The random walk with transition matrix $\mathbf{M}_{\mathbf{W}}$ assigns a probabilistic density to each path. $\mathbf{T}$ has no impact on paths' probability densities; however, the delays it introduces

at each node in a path alter the significance of the path: As $\mathbf{T}$ increases, the delays stretch the paths and weaken their contributions. In general, $\mathbf{R}$ changes the probability densities of the paths as the random walk now follows the transition matrix $\mathbf{M}_{\bar{\mathbf{W}}}$ rather than $\mathbf{M}_{\mathbf{W}}$. Thus, $\mathbf{R}$ and $\mathbf{T}$ provide two independent parameterizations: In the dynamic process, $\mathbf{W}$ and $\mathbf{R}$ regulate the distribution of multi-step interactions, while $\mathbf{T}$ controls the time needed for interaction to be realized on each path. Together, they define the significance of every path to the interaction among nodes.

The continuous-time kernel PageRank reflects the levels of nodes' activity in this family of dynamic processes [146, 62]. Like its (heat-kernel) PageRank counterparts, these ranking vectors provide rich information about the network. They can also be used to design efficient local algorithms for identifying clusters of nodes with more intensive internal interactions than external interactions during the dynamic process [146, 227].

4.7 Cheeger's Inequality and its Parameterization

> *Is conductance the right clusterability measure for modeling the interactions under in standard random walks? Is parameterized conductance the right clusterability measure for capturing the interplay between the network defined by* $\mathbf{W}$ *and the dynamic process associated with operator* $\mathcal{L}_{\mathbf{W},\mathbf{R},\mathbf{T}}$*?*

Although we are still seeking precise conceptual frameworks for answering to these questions, the classical Cheeger's inequality provides a partial mathematical justification for these clusterability measures. Below, we prove Cheeger's inequality in its parameterized form. We also use the proof of this theorem to provide our readers with a quick tour of spectral graph theory from a broader angle for modeling interactions among nodes in a network. In Section 8.1, we will return to the topic of the interplay between dynamic processes and network structures. We will discuss a recent result of [84], for analyzing the interplay between social-influence processes and network centrality.

Theorem 4.24 (Parameterized Cheeger Inequality)**.** For any $G = (V, E, \mathbf{W})$ and $\mathbf{R}$ and $\mathbf{T}$, let $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ be the eigenvalues of $\mathcal{L}_{\mathbf{W},\mathbf{R},\mathbf{T}}$. Then $\lambda_1 = 0$ and λ_2 satisfies the following inequalities:

$$\frac{\phi^2_{\mathbf{W},\mathbf{R},\mathbf{T}}}{2} \leq \lambda_2 \leq 2 \cdot \phi_{\mathbf{W},\mathbf{R},\mathbf{T}} \tag{4.34}$$

where $\phi_{\mathbf{W},\mathbf{R},\mathbf{T}} = \min_{S \in V} \left(\text{conductance}_{(\mathbf{W},\mathbf{R},\mathbf{T})}(S)\right)$.

Proof. Without loss of generality, we assume $\mathbf{R} = \mathbf{I}$. The following proof extends directly to other $\mathbf{R}$. Suppose $\mathbf{T} = \text{diag}[\tau_1, ..., \tau_n]$. For $i \in [n]$, let $\mathbf{v}_i$ be the eigenvector of $\mathcal{L}_{\mathbf{W},\mathbf{I},\mathbf{T}}$ associated with eigenvalue λ_i. Then $\mathbf{v}_1 = \mathbf{T}^{\frac{1}{2}} \cdot \mathbf{D}_{\mathbf{W}}^{\frac{1}{2}} \cdot \mathbf{1}$. Let $\boldsymbol{g} = \mathbf{D}_{\mathbf{W}}^{-\frac{1}{2}} \cdot \mathbf{T}^{-\frac{1}{2}} \cdot \mathbf{v}_2$. Because $\mathbf{v}_2 \perp \mathbf{v}_1$, we have, $\sum_v g_v \cdot (d_v \tau_v) = 0$. Thus, with symmetrization of $\mathcal{L}_{\mathbf{W},\mathbf{I},\mathbf{T}}$, we have:

$$\begin{aligned} \lambda_2 &= \frac{\mathbf{v}_2^T \cdot \mathbf{T}^{-\frac{1}{2}} \cdot \mathbf{D}_{\mathbf{W}}^{-\frac{1}{2}} \cdot \mathcal{L}_{\mathbf{W},\mathbf{I},\mathbf{T}} \cdot \mathbf{D}_{\mathbf{W}}^{\frac{1}{2}} \cdot \mathbf{T}^{\frac{1}{2}} \cdot \mathbf{v}_2}{\mathbf{v}_2^T \cdot \mathbf{v}_2} \\ &= \frac{\boldsymbol{g}^T \cdot (\mathbf{D}_{\mathbf{W}} - \mathbf{W}) \cdot \boldsymbol{g}}{\mathbf{v}_2^T \cdot \mathbf{v}_2} = \frac{\sum_{u,v \in V} (g_u - g_v)^2 w_{u,v}}{\sum_v g_v^2 d_v \tau_v}. \end{aligned}$$

We now show the last quantity is at least $\frac{\phi^2_{\mathbf{W},\mathbf{I},\mathbf{T}}}{2}$, which provides the same lower bound for λ_2. We will use the standard approach for proving Cheeger's inequality such as the one presented in [95]. Let $\boldsymbol{\pi}$ be the monotonic ordering of V according to $\boldsymbol{g}$, meaning:

$$g_{\boldsymbol{\pi}[1]} \geq g_{\boldsymbol{\pi}[2]} \geq \cdots \geq g_{\boldsymbol{\pi}[n]}.$$

For $k \in [n]$, let $S_k := \{\boldsymbol{\pi}[1], \cdots, \boldsymbol{\pi}[k]\}$. We only need to consider sets S_k with $\text{vol}_{\mathbf{W},\mathbf{I},\mathbf{T}}(S_k) \leq \frac{\text{vol}_{\mathbf{W},\mathbf{I},\mathbf{T}}(V)}{2}$, (because $\text{conductance}_{\mathbf{W},\mathbf{I},\mathbf{T}}(S_k) = \text{conductance}_{\mathbf{W},\mathbf{I},\mathbf{T}}(\bar{S_k})$). Let $r = \max\{k : \text{vol}_{\mathbf{W},\mathbf{I},\mathbf{T}}(S_k) \leq \frac{\text{vol}_{\mathbf{W},\mathbf{I},\mathbf{T}}(V)}{2}\}$. Following [95], let:

$$h_v = \begin{cases} g_v - g_{\boldsymbol{\pi}[r]} & \text{if } g_v \geq g_{\boldsymbol{\pi}[r]} \\ 0 & \text{otherwise} \end{cases} \textbf{ and } l_v = \begin{cases} g_{\boldsymbol{\pi}[r]} - g_v & \text{if } g_v \leq g_{\boldsymbol{\pi}[r]} \\ 0 & \text{otherwise} \end{cases}$$

Then, $g_v - g_{\boldsymbol{\pi}[r]} = h_v - l_v$, $h_v \cdot l_v = 0$, and $\forall u, v$, $(h_u - h_v)(l_v - l_u) \geq 0$: If both g_v and g_u on the same side of $g_{\boldsymbol{\pi}[r]}$ then this product is 0; otherwise $h_u \geq h_v$ iff $l_u \leq l_v$. Thus, this product is positive. Because:

$$\sum_{v \in V} (g_v - g_{\boldsymbol{\pi}[r]})^2 d_v \tau_v = \sum_{v \in V} g_v^2 d_v \tau_v + g_{\boldsymbol{\pi}[r]}^2 d_v \tau_v \geq \sum_{v \in V} g_v^2 d_v \tau_v,$$

where the equation follows from $\sum_v g_v d_v \tau_v = 0$, we have:

$$\begin{aligned}
\lambda_2 &= \frac{\sum_{u,v \in V} (g_u - g_{\boldsymbol{\pi}[r]} + g_{\boldsymbol{\pi}[r]} - g_v)^2 w_{u,v}}{\sum_v g_v^2 d_v \tau_v} \\
&\geq \frac{\sum_{u,v \in V} ((h_u - l_u) - (h_v - l_v))^2 w_{u,v}}{\sum_v (h_v - l_v)^2 d_v \tau_v} \\
&\geq \frac{\sum_{u,v \in V} (h_u - h_v)^2 w_{u,v} + (l_u - l_v)^2 w_{u,v}}{\sum_v (h_v^2 + l_v^2) d_v \tau_v} \\
&\geq \min \left\{ \frac{\sum (h_u - h_v)^2 w_{u,v}}{\sum_v h_v^2 d_v \tau_v}, \frac{\sum (l_u - l_v)^2 w_{u,v}}{\sum_v l_v^2 d_v \tau_v} \right\}.
\end{aligned}$$

Without loss of generality, we assume the first ratio is no more than the second ratio, and focus on the vertices $\{\boldsymbol{\pi}[1], ..., \boldsymbol{\pi}[r]\}$ in the "high half" of $\boldsymbol{\pi}$ in the analysis below. By the Cauchy-Schwartz inequality:

$$\begin{aligned}
\lambda_2 &\geq \frac{\sum_{u,v} (h_u - h_v)^2 w_{u,v}}{\sum_v h_v^2 d_v \tau_v} \cdot \frac{\sum_{u,v} (h_u + h_v)^2 w_{u,v}}{\sum_{u,v} (h_u + h_v)^2 w_{u,v}} \\
&\geq \frac{\left(\sum_{u,v} (h_u^2 - h_v^2) w_{u,v} \right)^2}{\left(\sum_v h_v^2 d_v \tau_v \right) \left(\sum_{u,v} (h_u + h_v)^2 w_{u,v} \right)}
\end{aligned} \tag{4.35}$$

Because $\tau_u \geq 1$ for all $u \in V$, we have:

$$\begin{aligned}
\sum_{u,v} (h_u + h_v)^2 w_{u,v} &\leq \sum_{u,v} 2(h_u^2 + h_v^2) w_{u,v} \\
&= 2 \sum_{u \in V} h_u^2 d_u \leq 2 \sum_{u \in V} h_u^2 d_u \tau_u.
\end{aligned}$$

Thus, the denominator of (4.35) is at most

$$2 \left(\sum_{u \in V} h_u^2 d_u \tau_u \right)^2.$$

We now bound the numerator of (4.35) by considering $\forall k \in [r]$, $S_k = \{\boldsymbol{\pi}[1], \cdots, \boldsymbol{\pi}[k]\}$. For convenience, let $S_0 = \emptyset$. First note that:

$$\mathrm{vol}_{\mathbf{W},\mathbf{I}.\mathbf{T}}(S_k) - \mathrm{vol}_{\mathbf{W},\mathbf{I},\mathbf{T}}(S_{k-1}) = d_{\boldsymbol{\pi}[k]}\tau_{\boldsymbol{\pi}[k]} \tag{4.36}$$

By the definition of $\phi_{\mathbf{W},\mathbf{I},\mathbf{T}}$, for all k, $\mathrm{conductance}_{\mathbf{W},\mathbf{I},\mathbf{T}}(S_k) \geq \phi_{\mathbf{W},\mathbf{I},\mathbf{T}}$. Since $\mathrm{vol}_{\mathbf{W},\mathbf{I},\mathbf{T}}(S_k) \leq \mathrm{vol}_{\mathbf{W},\mathbf{I},\mathbf{T}}(\bar{S_k})$ for all $k \in [r]$, we have:

$$\begin{aligned} \mathrm{cut}_{\mathbf{W}}(S_k, \bar{S_k}) &= \mathrm{conductance}_{\mathbf{W},\mathbf{I},\mathbf{T}}(S_k) \cdot \mathrm{vol}_{\mathbf{W},\mathbf{I},\mathbf{T}}(S_k) \\ &\geq \phi_{\mathbf{W},\mathbf{I},\mathbf{T}} \cdot \mathrm{vol}_{\mathbf{W},\mathbf{I},\mathbf{T}}(S_k) \end{aligned} \tag{4.37}$$

Therefore, expressing the summation of differences as a telescoping series, and then by terms involving consecutive nodes $(\boldsymbol{\pi}[k], \boldsymbol{\pi}[k+1])$:

$$\begin{aligned} &\left(\sum_{u,v}(h_u^2 - h_v^2)w_{u,v}\right)^2 \\ &= \left(\sum_{i<j}\left(\sum_{k=0}^{j-i-1} h^2_{\boldsymbol{\pi}[i+k]} - h^2_{\boldsymbol{\pi}[i+k+1]}\right) w_{\boldsymbol{\pi}[i],\boldsymbol{\pi}[j]}\right)^2 \\ &= \left(\sum_{k=1}^{r-1}\left(h^2_{\boldsymbol{\pi}[k]} - h^2_{\boldsymbol{\pi}[k+1]}\right) \cdot \mathrm{cut}_{\mathbf{W}}(S_i, \bar{S_i})\right)^2 \\ &\geq \left(\sum_{i=1}^{r-1}\left(h^2_{\boldsymbol{\pi}[k]} - h^2_{\boldsymbol{\pi}[k+1]}\right) \cdot \phi_{\mathbf{W},\mathbf{I},\mathbf{T}} \cdot \mathrm{vol}_{\mathbf{W},\mathbf{I},\mathbf{T}}(S_k)\right)^2 \\ &= \phi^2_{\mathbf{W},\mathbf{I},\mathbf{T}} \cdot \left(\sum_{k=1}^{r} h^2_{\boldsymbol{\pi}[k]} \cdot (\mathrm{vol}_{\mathbf{W},\mathbf{I},\mathbf{T}}(S_k) - \mathrm{vol}_{\mathbf{W},\mathbf{I},\mathbf{T}}(S_{k-1}))\right)^2 \\ &= \phi^2_{\mathbf{W},\mathbf{I},\mathbf{T}} \cdot \left(\sum_{v \in V} h_v^2 \cdot d_v\tau_v\right)^2. \end{aligned}$$

The last inequality holds because $h_v = 0$, $\forall v \geq r$. Consequently, we have $\lambda_2 \geq \frac{\phi^2_{\mathbf{W},\mathbf{I},\mathbf{T}}}{2}$, as stated in the theorem. The right hand side of the theorem follows from the same argument for the standard Cheeger's Inequality: In particular, $\forall S \subset V$, we can use a test vector based on

the characteristic vector $\mathbf{1}_S$ of S to prove:

$$\begin{aligned}\lambda_2 &\leq \frac{\mathrm{cut}_{\mathbf{W},\mathbf{R},\mathbf{T}}(S,\bar{S}) \cdot \mathrm{vol}_{(\mathbf{W},\mathbf{R},\mathbf{T})}(V)}{\mathrm{vol}_{(\mathbf{W},\mathbf{R},\mathbf{T})}(S) \cdot \mathrm{vol}_{(\mathbf{W},\mathbf{R},\mathbf{T})}(\bar{S})} \\ &\leq 2 \cdot \frac{\mathrm{cut}_{\mathbf{W},\mathbf{R},\mathbf{T}}(S,\bar{S})}{\min(\mathrm{vol}_{(\mathbf{W},\mathbf{R},\mathbf{T})}(S), \mathrm{vol}_{(\mathbf{W},\mathbf{R},\mathbf{T})}(\bar{S}))}.\end{aligned}$$

The last inequality follows from the fact that:

$$\mathrm{vol}_{(\mathbf{W},\mathbf{R},\mathbf{T})}(V) \leq 2 \cdot \max(\mathrm{vol}_{(\mathbf{W},\mathbf{R},\mathbf{T})}(S), \mathrm{vol}_{(\mathbf{W},\mathbf{R},\mathbf{T})}(\bar{S})), \quad \forall S \subset V.$$

□

5

Partitioning: Geometric Techniques for Data Analysis

In many applications, graphs have additional geometric structures. Sometimes, the graphs — such as networks of wireless or mobile devices, structures of protein folding, nearest neighborhood graphs of data points, and meshes in computer graphics and numerical simulation — are directly derived from geometric data. At other times, one first obtains graphs or matrices and then derives the underlying geometric structures, such as latent semantic structures of term-document matrices, metrics of effective resistances, embeddings from graph drawing, and layouts of circuits. For these graphs, networks, and matrices, we can use their valuable underlying geometric structures to understand them. For example, we can apply geometric intuition to formulate algebraic programs to model solution concepts. We can drive geometric characterizations of structural measures, such as clusterability and dimensionality. We can also apply geometric techniques to develop efficient algorithms.

For example, a basic and widely used family of geometric graphs is the k-NNGs: Suppose $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ is a point set in $\mathbb{R}^d$ and $k > 0$ is a positive integer. For each $\mathbf{p}_i \in P$, let $N_k(\mathbf{p}_i)$ be the set of k points in P that are closest to $\mathbf{p}_i$ (where ties are broken arbitrarily). Then,

the *k-nearest neighborhood graph* (k-NNG) of P [279, 252] is either the directed graph with edge set:

$$E = \{(i,j) : \mathbf{p}_j \in N_k(\mathbf{p}_i)\},$$

or the undirected graph with edge set:

$$E = \{(i,j) : \mathbf{p}_i \in N_k(\mathbf{p}_j) \text{ or } \mathbf{p}_j \in N_k(\mathbf{p}_i)\}.$$

The notion of $N_k(\mathbf{p}_i)$ and k-NNG can be directly extended to point set P in any metric space $\mathcal{M}$. Consider also the family of networks that arise in data mining and information retrieval: Suppose $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ is a set of unit d-dimensional feature vectors (See Section 2.2). Then, the *cosine similarity* can be used to define a weighted graph $G = (V, E, \mathbf{W})$ with $V = [n]$ and affinities $w_{i,j} = \mathbf{p}_i^T \mathbf{p}_j$.

Once these geometric graphs over data points have been formed, we can use their combinatorial/algebraic measures to study the structural properties of these data points [279, 38]. For example, we can define the following data clusterability measure based on the conductance of groups in k-NNGs.

Definition 5.1 (Data Clusterability by k-NNGs)**.** Suppose $P = (\mathbf{p}_1, ..., \mathbf{p}_n)$ is set of data points in a metric space $\mathcal{M}$. For $S \subset P$, let clusterability$_{k\text{-NNG}}(S)$ be the fraction of S's k-NNG relations with $\bar{S} = P - S$. In other words:

$$\text{clusterability}_{k\text{-NNG}}(S) := \frac{|(\mathbf{p}, \mathbf{q}) : \mathbf{p} \in S \ \& \ \mathbf{q} \in \bar{S} \cap N_k(\mathbf{p})|}{k \cdot |S|}.$$

If clusterability$_{k\text{-NNG}}(S) \ll 1$, then members of S essentially have all their k-nearest neighbors in the cluster. In this chapter, we will illustrate a collection of powerful geometric techniques for designing scalable algorithms. We will focus on two basic problems: (1) the computation of k-NNGs, and (2) the identification of a geometric cluster S with clusterability$_{k\text{-NNG}}(S) = o(1)$, according to n. Both scalable algorithms work with point sets in $\mathbb{R}^d$, for any constant d (e.g., $d = 50$).

This collection of scalable geometric techniques is based on integrating *geometric partitioning* with *geometric sampling*. We will discuss two

basic concepts in geometric partitioning: *centerpoints* and *geometric separators*. We then illustrate their usefulness in a simple and scalable *geometric divide-and-conquer* algorithm for constructing k-NNGs and for identifying geometric clusters.

For geometric data in fixed dimensions, we will prove the following strong clusterability theorem regarding $\text{clusterability}_{k\text{-NNG}}$ function. The techniques to prove this theorem has led to several scalable algorithms for geometric data in fixed dimensions.

Theorem 5.2 (Clusterability of Low-Dimensional Data)**.** For any fixed integer $d \geq 1$ and for any pairs of integers $k, t > 1$, every point set $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \subset \mathbb{R}^d$ can be partitioned into t clusters $P_1, ..., P_t$ such that for all $i \in [t]$, (i) $|P_i| \in [\frac{n}{2t}, \frac{2n}{t}]$, and (ii):

$$\text{clusterability}_{k\text{-NNG}}(P_i) = O\left(\left(\frac{k \cdot t}{n}\right)^{1/d}\right).$$

Theorem 5.2 shows that for any fixed dimension d, for any $k \cdot t = o(n)$, every data point set in $\mathbb{R}^d$ has a "balanced" t-way clustering in which the $\text{clusterability}_{k\text{-NNG}}$ of each cluster is $o(1)$ according to n. In other words, essentially all k-NNG relations are internal to the clusters. It would be wonderful that such strong clusterability statement holds for data points derived from real-world social/information networks. Thus, Theorem 5.2 illustrates that either the dimensionality of real-world social/information networks is usually intrinsically high, or we have to go far beyond the NNG-framework when we study these networks by embedding them in a low dimensional space.

In Section 5.8, we will conclude this chapter by connecting spectral graph theory with the geometry of graphs [315]. Spectral methods usually analyze input networks by first deriving geometric embeddings from their adjacency/random-walk/Laplacian matrices. We will show that the geometric techniques for proving Theorem 5.2 can also be used to establish remarkable bound on the eigenvalues for Laplacian matrices. Together with Cheeger's inequality, these bounds can be used to prove that spectral graph-partitioning methods work well for a large class of practical graphs [315].

5.1 Centerpoints and Regression Depth

For one-dimensional data, medians and their approximations have been proven to be statistically robust. Suppose $P = \{p_1, ..., p_n\}$ is a set of n real numbers. For $\delta \in (0, 1/2]$, $c \in \mathbb{R}$ is a *δ-median* of P if $\max|P^-|, |P^+| \leq (1-\delta)n$, where $P^- = \{p_i : p_i < c\}$ and $P^+ = \{p_i : p_i > c\}$. A $(1/2)$-median of P is known simply as a *median.*

Centerpoints are a high-dimensional generalization of medians.

Definition 5.3 (Centerpoints)**.** Suppose $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ is a point set in $\mathbb{R}^d$. For $\delta \in (0, 1/2]$, a point $\mathbf{c} \in \mathbb{R}^d$ is a *δ-centerpoint* of P if for all unit vectors $\mathbf{z} \in \mathbb{R}^d$, the projection $\mathbf{z}^T\mathbf{c}$ is a *δ-median* of the projections, $\mathbf{z}^T \cdot P = \{\mathbf{z}^T\mathbf{p}_1, \ldots, \mathbf{z}^T\mathbf{p}_n\}$.

Every d-dimensional hyperplane $\mathbf{h}$ divides P into three subsets: $P^+ = \mathbf{h}^+ \cap P$, $P^- = \mathbf{h}^- \cap P$, and $P \cap \mathbf{h}$. Let the *splitting ratio* of $\mathbf{h}$ over P — denoted by $\delta_{\mathbf{h}}(P)$ — be $\delta_{\mathbf{h}}(P) := \frac{\max\left(|P^+|, |P^-|\right)}{|P|}$. Definition 5.3 can be restated as: $\mathbf{c} \in \mathbb{R}^d$ is a δ-centerpoint of P if the splitting ratio of every hyperplane $\mathbf{h}$ passing through $\mathbf{c}$ is at most $(1-\delta)$.

Theorem 5.4. Every point set $P \subset \mathbb{R}^d$ has a $\frac{1}{d+1}$-centerpoint.

Proof. (Sketch) To focus on the algorithmic nature of this theorem, we will sketch a proof to show that, for $\delta \leq \frac{1}{d+1}$, the set of all δ-centerpoints is convex. Consequently, one can apply linear programming to compute a centerpoint. The proof goes as follows: Let $\mathcal{H}_{\mathcal{P},\delta}$ denote the set of all closed halfspaces that contains at least $\lceil(1-\delta)|P|\rceil$ points of P. Consider any set of $(d+1)$ halfspaces from $\mathcal{H}_{\mathcal{P},\delta}$. If $\delta \leq \frac{1}{d+1}$, because each of their complements — which are open halfspaces — contains less than $\frac{|P|}{d+1}$ points from P, then there must be a point in P that is common to these $(d+1)$ halfspaces.

Thus, following Helly's classical theorem from convex geometry — stated in Theorem 5.5 below — the intersection of halfspaces from $\mathcal{H}_{\mathcal{P},\delta}$ is not empty for $\delta \leq \frac{1}{d+1}$. This intersection precisely defines the set of all δ-centerpoints of P. □

Theorem 5.5 (Helly)**.** Suppose $\mathcal{K}$ is a family of at least $d+1$ convex sets in $\mathbb{R}^d$, and $\mathcal{K}$ is finite or each member of $\mathcal{K}$ is compact. Then, if each $d+1$ members of $\mathcal{K}$ have a common point, there must be a point common to all members of $\mathcal{K}$.

Extending the basic fact that every one-dimensional point set has a median, Theorem 5.4 shows that every d-dimensional point set has a $\frac{1}{d+1}$-centerpoint. However, unlike medians, the set of $\frac{1}{d+1}$-centerpoints may not intersect the point set itself [106]. To construct the linear program for δ-centerpoints of P, it is sufficient to consider the intersection of "boundary" halfspaces from $\mathcal{H}_{\mathcal{P},\delta}$. The bounding hyperplanes of these halfspaces contain at least d points of P. There are $O(n^d)$ such halfspaces. Thus, the set of δ-centerpoints of P is defined by a d-dimensional linear program with $O(n^d)$ inequalities.

Using Megiddo's linear programming algorithm [247], we can compute a δ-centerpoint in $O(n^d)$ time. Although this algorithm is not scalable, in the next section, we will show that it can be made scalable with the help of sampling.

Regression Depth: As proved in [17], centerpoints are strongly related to the concept of regression depth introduced by Rousseeuw and Hubert in robust statistics [172]:

Definition 5.6 (Regression Depth)**.** For a point set P in $\mathbb{R}^d$, the *regression depth* of a hyperplane $\mathbf{h}$ is equal to the minimum number of points of P intersected by the hyperplane as it continuously "rotates" from its initial position to a vertical position.[1]

---

[1] A hyperplane in $\mathbb{R}^d$ is vertical if its normal vector is perpendicular to $\mathbf{1}_d$.

Statistically, the regression depth of a hyperplane $\mathbf{h}$ — which represents a linear classifier of the point set — measures the smallest classification-label changes that can make $\mathbf{h}$ a "nonfit" classifer [172]. Thus, regression depth provides a key quality measure for robust linear regression. Like medians, hyperplanes with high regression depth are robust in various error models [172]. Amenta, Bern, Eppstein, and Teng proved the following theorem [17] by applying Brouwer's fixed-point theorem [75] to continuously map a $\frac{1}{d+1}$-centerpoint into a "deep" hyperplane.

Theorem 5.7 (Regression Depth)**.** Every set P of n points in $\mathbb{R}^d$ has a hyperplane with regression depth at least $\frac{n}{d+1}$.

If one applies the point-hyperplane duality, the following also holds:

Corollary 5.8 (Arrangement Depth)**.** The arrangement of any n hyperplanes in $\mathbb{R}^d$ has a cell whose points cannot escape to infinity without crossing at least $\frac{n}{d+1}$ hyperplanes.

5.2 Scalable Algorithms for Centerpoints

Unlike medians, it remains an open question whether scalable algorithms for computing centerpoints in higher dimensions exist. Below, we discuss two approximation algorithms for centerpoints. Both are super-scalable and extremely simple. The first one uses straightforward sampling with linear programming. It produces an approximate $\frac{1}{d+1}$-centerpoint with high probability. Although this algorithm is super-scalable, it is highly impractical — it needs to solve a large constant sized linear program. The second algorithm applies an evolutionary algorithm (EA). We will show that it converges rapidly to a $\frac{1}{d^2}$-centerpoint and is "practically" scalable [148]. The first algorithm is built on the following basic concept of geometric approximation:

Definition 5.9 (Geometric Samples). For a point set $P \subset \mathbb{R}^d$, $S \subset P$ is an ϵ-*sample* of P if for all halfspaces H in $\mathbb{R}^d$:

$$\left| \frac{|H \cap S|}{|S|} - \frac{|H \cap P|}{|P|} \right| \leq \epsilon \tag{5.1}$$

Proposition 5.10 (Centerpoint Approximation by Sampling). If S an ϵ-*sample* of P and $\mathbf{c}_S$ is a δ-centerpoint of S, then $\mathbf{c}_S$ is a $(\delta - \epsilon)$-centerpoint of P.

An Application of VC Sampling Theory

Suppose $\epsilon \in (0, \delta)$. The celebrated Vapnik-Chervonenkis theory [342] uses dimensions to characterize the effectiveness of sampling.

Theorem 5.11 (Vapnik and Chervonenkis). $\exists$ a constant c such that for any point set $P \subset \mathbb{R}^d$, for any $\epsilon, \delta \in (0, 1)$, if S is a set of $c \cdot \frac{d}{\epsilon^2} \left(\log \frac{d}{\epsilon^2 \delta} \right)$ uniform and independent samples from P, then:

$$\Pr[S \text{ is an } \epsilon\text{-sample of } P] \geq 1 - \delta.$$

This theorem immediately leads to the following scalable algorithm for approximating centerpoints.

Algorithm: CenterpointsBySampling(P, ϵ, δ)

1: Select S of $k = c \cdot \frac{d}{\epsilon^2} \left(\log \frac{d}{\epsilon^2 \delta} \right)$ uniform and independent samples from P.
2: Return a centerpoint $\mathbf{c}_S$ of S, using the Linear Programming algorithm for centerpoints discussed in the last section.

Corollary 5.12 (Theoretically Scalable Computation of Centerpoints)**.** For parameters $0 < \epsilon < \frac{1}{d+1}$ and $\delta \in (0,1)$, with probability at least $(1-\delta)$, CenterpointsBySampling(P, ϵ, δ) returns a $(\frac{1}{d+1} - \epsilon)$-centerpoint of P in time:

$$2^{O(d)} \left(\frac{d}{\epsilon^2} \cdot \log \frac{d}{\epsilon^2 \delta} \right)^d.$$

In theory, when d is fixed, the sampling algorithm is super-scalable. Its complexity does not depend on the size of the input! However, this algorithm is not scalable in practice, due to its exponential dependence on dimension d. Even in three dimensions, we needs about 1000 sample points, which means that the algorithm needs to solve a linear program with millions of inequalities.

Below, we will discuss an extremely simple evolutionary algorithm [148] for approximating centerpoints, and a rigorous analysis — drawn from the work of Clarkson *et al.* [96] — on its quality. This simple construction is one of the illustrative examples of practical and provably-good scalable data-analysis algorithms.

An Evolutionary Algorithm for Approximated Median

To provide better intuition and motivation of the evolutionary algorithm, we first focus on one dimension and analyze the following algorithm for approximating medians.

Algorithm: Middle-of-Three Evolution(P, h)

Require: a set of numbers $P = \{p_1, ..., p_n\}$ and an integer h

1: Select a population S of 3^h *i.i.d* uniform samples from P.
2: **while** S has more than one element **do**
3: Randomly partition the elements of S into groups of three, and eliminate from each group the smallest and the largest element from S.
4: Return the survivor.

Theorem 5.13 (Middle-of-Three Evolution)**.** For any $\delta \in (0,1)$ and $\epsilon \in (0,1/2)$, for any totally ordered set P, if

$$3^h \geq O\left(\left(\frac{1}{\epsilon}\right)^{3.45} \cdot \left(\log \frac{1}{\delta}\right)^{1.59}\right)$$

then with probability at least $1-\delta$, Middle-of-Three Evolution(P,h) returns a $(1/2-\epsilon)$-median.

Proof. In this proof, we will use MoT as the abbreviation of Middle-of-Three Evolution. MoT only uses comparisons. Thus, only the relative ranks (not the values of P) matters. So, without loss of generality, we assume that P is a permutation of the set $\{1/n, 2/n, ..., 1\}$. MoT is symmetric with respect to the rankings. Thus, the expected value of the survivor of the Middle-of-Three Evolution is exactly $n/2$.

We now analyze the concentration of this evolution. Let $p_h(x)$ be the probability that the survivor of $\mathsf{MoT}(P,h)$ has value at most x. Then, $p_0(x) = \frac{\lfloor x \cdot n \rfloor}{n} \leq x$. Moreover, $\mathsf{MoT}(P,h) \leq x$ if and only if either two or all three of evolution's last three survivors are at most x. Thus:

$$\begin{aligned} p_h(x) &= \binom{3}{2}(p_{h-1}(x))^2(1-p_{h-1}(x)) + (p_{h-1}(x))^3 \\ &= 3(p_{h-1}(x))^2 - 2(p_{h-1}(x))^3 \end{aligned}$$

Suppose $x = \frac{1}{2} - \epsilon$. We divide the analysis into two cases: (i) $\epsilon \in [\frac{1}{4}, \frac{1}{2}]$ (implying $x \leq \frac{1}{4}$) and (ii) $\epsilon < \frac{1}{4}$. In Case (i), by induction, we have:

$$\begin{aligned} p_h(x) &= 3(p_{h-1}(x))^2 - 2(p_{h-1}(x))^3 \leq 3(p_{h-1}(x))^2 \\ &\leq 3 \cdot 3^2 \cdots 3^{2^{h-1}} (p_0(x))^{2^h} \leq 3^{-1}(3x)^{2^h} \leq 3^{-1}\left(\frac{3}{4}\right)^{2^h}. \end{aligned}$$

In Case (ii), because $2\epsilon^3 \leq \frac{\epsilon}{8}$, by $p_1(\frac{1}{2}-\epsilon) = \frac{1}{2} - \frac{3\epsilon}{2} + 2\epsilon^3$, we have $p_1(\frac{1}{2}-\epsilon) \leq \frac{1}{2} - \frac{11\epsilon}{8}$. Setting $t = \lceil \log_{\frac{11}{8}}(\frac{1}{4\epsilon}) \rceil$, by induction, we have $p_t(\frac{1}{2}-\epsilon) \leq \frac{1}{4}$, and therefore:

$$p_h(x) = p_{h-t}(p_t(x)) \leq p_{h-t}\left(\frac{1}{4}\right) \leq 3^{-1}\left(\frac{3}{4}\right)^{2^{h-t}}.$$

Thus, $\mathsf{MoT}(P,h) \leq \frac{1}{2} - \epsilon$ with probability at least $1 - \frac{\delta}{2}$, as long as:

$$3^h \geq \max\left(3^{\lceil \log_{\frac{11}{8}}(\frac{1}{4\epsilon})\rceil}, 1\right)\left(\log_{\frac{4}{3}}\frac{2}{3\delta}\right)^{\log_2 3} \geq O\left(\left(\frac{1}{\epsilon}\right)^{3.45}\left(\log\frac{1}{\delta}\right)^{1.59}\right).$$

The theorem then follows from the symmetry of MoT. □

An Evolutionary Algorithm for Centerpoints

To extend Middle-of-Three Evolution to higher dimensions, we first generalize the middle-of-three operator. We use the following interpretation of this operator: The middle of any three numbers $x, y, z \in \mathbb{R}$ lies in the interval (i.e., the convex hull) defined by the other two points. Clearly, if three points in $\mathbb{R}^2$ are not co-linear, then they do not have this type of division. But for any four two-dimensional points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4$, there are two basic cases:

1. All four points are extreme points of their convex hull, which forms a convex quadrilateral. Then, the two diagonals of this convex quadrilateral must intersect.

2. Otherwise, it must be the case that one of the points, say $\mathbf{p}_1$ lies in the convex hull (i.e., the triangle) defined by the other three points.

In both cases, we find a partition of points in $\{\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4\}$ into two subsets, such that the convex hulls of the two subsets intersect. By returning this intersection point, we generalize the middle-of-three operator to two dimensions. In fact, the following Radon's classical theorem [106, 336] enables us to extend the middle-of-three operator to any dimension.

Theorem 5.14 (Radon)**.** Every point set $Q \subset \mathbb{R}^d$ with $|Q| \geq d+2$ can be partitioned into two subsets (Q_1, Q_2) such that the convex hulls of Q_1 and Q_2 have a common point.

Proof. Suppose $Q = \{\mathbf{q}_1, ..., \mathbf{q}_n\}$ with $n \geq d+2$. Consider the system of $d+1$ homogeneous linear equations:

$$\sum_{i=1}^{n} \alpha_i = 0 \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i \mathbf{q}_i = \mathbf{0} \tag{5.2}$$

Since $n \geq d+2$, the system has a nonzero solution $\alpha_1, ..., \alpha_n$. Define:

$$I_+ = \{i : \alpha_i \geq 0\} \quad \text{and} \quad I_- = \{i : \alpha_i < 0\}$$
$$Q_1 = \{\mathbf{q}_i : i \in I_+\} \quad \text{and} \quad Q_2 = \{\mathbf{q}_i : i \in I_-\}$$

Then, letting $c = \sum_{i \in I_+} \alpha_i > 0$, it must be the case that $\sum_{j \in I_-} \alpha_j = -c$ and $\sum_{i \in I_+} (\alpha_i / c)\mathbf{q}_i = \sum_{j \in I_-} (-\alpha_j / c)\mathbf{q}_j$. Thus, the partition (Q_1, Q_2) of Q has the desired property. □

Definition 5.15 (Radon points)**.** Let Q be a set of points in $\mathbb{R}^d$. A point $\mathbf{q} \in \mathbb{R}^d$ is a *Radon point* of Q if Q can be partitioned into two disjointed subsets Q_1 and Q_2, such that $\mathbf{q}$ is in the intersection of the convex hulls of Q_1 and Q_2.

Note that a Radon point can be found in $O(d^3)$ time by Gaussian elimination. The following algorithm — RadonEvolution — naturally extends Middle-of-Three Evolution to high dimensions.

Algorithm: RadonEvolution(P, d, h)

1: Select a population S of $(d+2)^h$ *i.i.d.* uniform samples from P.
2: **while** S has more than one element **do**
3: Randomly partition the elements of S into groups of $(d+2)$, and replace points from each group by one of its Radon points.
4: Return the point $\mathbf{c}$ left in S.

To analyze the following RadonEvolution, we need to show that for any $\mathbf{z} \in \mathbb{R}^d$, $\mathbf{z}^T\mathbf{c}$ is an approximate median of $\mathbf{z}^T \cdot P$. We will first focus on a given projection vector $\mathbf{z}$.

Lemma 5.16. Let $\mathbf{c} = \mathsf{RadonEvolution}(P, d, h)$. Let $\beta_d = 1/\binom{d+2}{2}$. For any $\beta < \beta_d$ and for any unit vector $\mathbf{z} \in \mathbb{R}^d$, the probability that $\mathbf{z}^T\mathbf{c}$ is not a β-median of $\mathbf{z}^T \cdot P$ is at most $\beta_d(\frac{\beta}{\beta_d})^{2^h}$.

Proof. We will follow Theorem 5.13 by assuming (without loss of generality) that $\mathbf{z}^T \cdot P = \{1/n, 2/n, ..., 1\}$. For any positive integer h, let $p_h(x)$ be the probability that $\mathbf{z}^T \cdot \mathsf{RadonEvolution}(P, d, h)$ is less than x. Again, $p_0(x) \leq x$. Note that $\mathbf{z}^T \cdot \mathsf{RadonEvolution}(P, d, h) \leq x$ only if least two of its $d + 2$ inputs have $\mathbf{z}$-projection at most x. $\mathsf{RadonEvolution}(P, d, h)$ takes inputs from the outputs of $d + 2$ independent calls to $\mathsf{RadonEvolution}(P, d, h - 1)$. Thus:

$$\begin{aligned} p_h(x) &\leq \binom{d+2}{2}(p_{h-1}(x))^2 \\ &\leq \frac{1}{\beta_d} \cdot \frac{1}{\beta_d^2} \cdots \frac{1}{\beta_d^{2^{h-1}}} \cdot p_0(x)^{2^h} = \beta_d \left(\frac{x}{\beta_d}\right)^{2^h}. \end{aligned}$$

□

We now use Lemma 5.16 and Theorem 5.11 (Vapnik-Chervonenkis theory) to bound the number of generations h needed for $\mathbf{c} = \mathsf{RadonEvolution}(P, d, h)$ to obtain a β-centerpoint of P for any $\beta < \beta_d$. Let $\epsilon = \frac{\beta_d - \beta}{2}$. By Theorem 5.11, with high probability, a $\frac{\beta+\beta_d}{2}$-centerpoint of the set of $O\left(\frac{d}{\epsilon^2}\left(\log\frac{d}{\epsilon^2\delta}\right)\right)$ uniform and independent samples of P is also a β-centerpoint for P.

The linear program for the $\frac{\beta+\beta_d}{2}$-centerpoints of the sample has only $O\left(\frac{d}{\epsilon^2}\left(\log\frac{d}{\epsilon^2\delta}\right)\right)^d$ constraints. Thus, to certify $\mathbf{c}$ is a β-centerpoint of P, we only need to verify the projections defined by these constraints. Applying union bound to Lemma 5.16, running $\mathsf{RadonEvolution}$ with $h = O(\log d + \log\log\frac{1}{\epsilon\delta})$ is sufficient for obtaining a β-centerpoint of P with probability $1 - \delta$. This amount to an initial population of $(d + 2)^{O(h)} = (d\log\frac{1}{\epsilon\delta})^{O(\log d)}$. Therefore:

Theorem 5.17 (Scalable Radon Evolution)**.** For $\beta < \beta_d$ and any point set $P \in \mathbb{R}^d$, with probability $1 - \delta$, RadonEvolution computes a β-centerpoint of P, using $(d \log \frac{1}{(\beta_d - \beta)\delta})^{O(\log d)}$ time.

In RadonEvolution, population declines rapidly — it requires $(d+2)$ points to produce a new point. Clarkson *et al.* [96] showed that we can use a layered DAG rather than a $(d + 2)$-way tree to "repopulate" the data points during the evolution: The evolution starts with a random population S. At each step, it forms $|S|$ sets of (d+2) random data points from the current population, and uses their Radon points as the new population. A careful analysis further eliminates the sub-exponential dependence of $\log d$. As shown in [96], this variation of Radon evolution can obtain a β-centerpoint (for any $\beta < \beta_d$) with probability $1 - \delta$ in time polynomial in d, $\log \frac{1}{\beta_d - \beta}$, and $\log \frac{1}{\delta}$.

5.3 Geometric Separators

Like medians, centerpoint is an important concept for robust statistics [17, 172]. They are also useful for developing divide-and-conquer geometric algorithms. In this section, we review a fundamental concept called *sphere separators* [252] and analyze a scalable algorithm for finding them. This algorithm uses scalable centerpoint approximation as a key subroutine. Geometric separators are themselves the fundamental building blocks for divide-and-conquer. In the next section, we apply geometric separators to design a simple scalable algorithm for constructing the nearest neighborhood graphs of geometric data.

Neighborhood Systems of Geometric Data

In data analysis, such as image and document search, we need to identify points close to an input data point. Explicitly or implicitly, we need to use some notion of neighborhoods. In this section, we will analyze the most basic geometric neighborhoods: *balls* in Euclidean space.

The results below can be extended to balls in other norms.

Definition 5.18 (Neighborhood Systems)**.** A *neighborhood system in d dimensions* is a set $\Gamma = \{B_1, \ldots, B_n\}$ of closed balls in $\mathbb{R}^d$.

Balls arise naturally in geometric algorithms. They are also common in applications such as computer graphics, computer vision, image processing, protein folding, wireless networks, and mobile computing. In these applications, the neighborhood systems satisfy certain locality properties. We will focus on the following locality measure:

Definition 5.19 (Ply: A Locality Measure)**.** The *ply* of a d-dimensional neighborhood system $\Gamma = \{B_1, \ldots, B_n\}$, denoted by $\text{Ply}(\Gamma)$, is the smallest integer k such that no point in $\mathbb{R}^d$ lies in the interior of more than k balls of Γ.

For example, k-NNGs can be naturally defined by balls around each point. Suppose $P = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$ is a point set in $\mathbb{R}^d$. Let $B_i^{(k)}$ be the ball centered at $\mathbf{p}_i$ with radius equal to the distance from $\mathbf{p}_i$ to its k^{th} nearest neighbor in P. This is the largest ball whose interior contains at most k points from P (including $\mathbf{p}_i$ itself). The k-NNG of P can be defined based on which points are in $B_i^{(k)}$, $\forall i \in [n]$. We refer to:

$$\Gamma_k(P) = \left\{B_1^{(k)}, \ldots, B_n^{(k)}\right\}$$

as the k^{th} *neighborhood system* of P. These neighborhood systems enjoy the following locality property [252]:

Proposition 5.20 (Locality of Nearest Neighborhoods)**.** For any d, $\exists$ a constant τ_d (known as the d-dimensional kissing number [99]), such that for any d-dimensional point set P, $\text{Ply}(\Gamma_k(P)) \leq \tau_d k$.

Sphere Separators

Each $(d-1)$-sphere S partitions a d-dimensional neighborhood system $\Gamma = \{B_1, \ldots, B_n\}$ into three subsets:

- $\Gamma_{exterior}(S)$: the set of balls of Γ that lie in the exterior of S.
- $\Gamma_{interior}(S)$: the set of balls of Γ that lie in the interior of S.
- $\Gamma_{intersect}(S)$: the set of balls of Γ that intersect S.

Proposition 5.21 (Sphere Separators)**.** For any d-dimensional neighborhood system Γ and any $(d-1)$-sphere S, the partition $\Gamma = \Gamma_{exterior}(S) \cup \Gamma_{interior}(S) \cup \Gamma_{intersect}(S)$ has the property:

$$\forall B_i \in \Gamma_{exterior}(S), \forall B_j \in \Gamma_{interior}(S), B_i \cap B_j \neq \emptyset.$$

Miller *et al.* [252] proved the following geometric theorem:

Theorem 5.22 (Sphere-Separators)**.** For any d-dimensional neighborhood system $\Gamma = \{B_1, \ldots, B_n\}$ and for any $0 < \delta \leq \frac{1}{d+2}$, there exists a sphere S such that:

- **Balance of Partition**: $|\Gamma_{exterior}(S)|, |\Gamma_{interior}(S)| \leq (1-\delta)n$.
- **Sub-linearity of Cut**: $|\Gamma_{interior}(S)| \leq 2d \cdot \text{Ply}(\Gamma)^{\frac{1}{d}} n^{1-\frac{1}{d}}$.

Algorithmically, the proof below yields a (randomized) scalable reduction from the problem of finding high-quality sphere separators to the problem of (approximately) computing centerpoints.

Theorem 5.23 (Scalable Reduction: Sphere Separators to Centerpoints)**.** For any $\delta \in (0, \frac{1}{d+2})$, after a single call to computing a $(d+1)$-dimensional δ-centerpoint, one can find a separating sphere S that guarantees $|\Gamma_{interior}(S)|, |\Gamma_{exterior}(S)| \leq (1-\delta)n$ and $|\Gamma_{intersect}(S)| \leq 2d \cdot \text{Ply}(\Gamma)^{\frac{1}{d}} n^{1-\frac{1}{d}}$ in $O(d)$ time, with high probability.

Proof. (Sketch for Theorem 5.22 and Theorem 5.23)

Step 1. *Conformal Mapping*[2] — *lifting data from* $\mathbb{R}^d$ *to* S^d: Given a neighborhood system $\Gamma = \{B_1, ..., B_n\}$, the first step is to lift it from $\mathbb{R}^d$ to the unit d-sphere S^d in $\mathbb{R}^{d+1}$. We first use *stereographic projection* to be denoted by Π: We view a point $\mathbf{x} \in \mathbb{R}^d$ as a point of $\mathbb{R}^{d+1}$ of form $[\mathbf{x}, 0]$, by appending 0 as its final coordinate. Then, $\Pi(\mathbf{x})$ is simply the intersection of S^d with the line in $\mathbb{R}^{d+1}$ connecting $\mathbf{x}$ to the north pole of S^d, $[0, 0, \ldots, 0, 1]^T$.

$\Pi(\mathbf{x})$ and its inverse are respectively given by the following formula:

$$\begin{aligned} \Pi(\mathbf{x}) &= \frac{1}{\mathbf{x}^T\mathbf{x}+1}\begin{pmatrix} 2\mathbf{x} \\ \mathbf{x}^T\mathbf{x}-1 \end{pmatrix}, \quad \forall \mathbf{x} \in \mathbb{R}^d \\ \Pi^{-1}(\mathbf{u}) &= \frac{\mathbf{u}[1:d]}{1-\mathbf{u}[d+1]}, \quad \forall \mathbf{u} \in S^d. \end{aligned}$$

Stereographic projection can be more generally defined as the following: For any vector $\mathbf{u} \in \mathbb{R}^{d+1} \setminus \{\mathbf{0}\}$, let $\mathbf{h_u}$ be the d-dimensional hyperplane passing through $\mathbf{0}$ with normal vector $\mathbf{u}/||\mathbf{u}||$. Let $\Pi_{\mathbf{u}}$ be the stereographic projection that maps $\mathbf{h_u}$ to S^d and let $\Pi_{\mathbf{u}}^{-1}$ be its inverse. For example, the stereographic projection Π above is $\Pi_{\mathbf{1}_{d+1}}$. Note that $\Pi_{\mathbf{u}}^{-1} \circ \Pi(\Gamma)$ is a d-dimensional neighborhood system on $\mathbf{h_u}$.

We will use the fact that the stereographic projection is *sphere-preserving* [252] — it maps every ball — including the "degenerate balls", halfspace — of $\mathbb{R}^d$ to a spherical cap on S^d. Thus, Π induces a *neighborhood system* of spherical caps $\Pi(\Gamma) = \{S_1, ..., S_n\}$, where $S_i = \Pi(B_i)$ is the image of B_i under the stereographic projection Π. Let $\mathrm{Ply}(\{S_1, ..., S_n\})$ be the smallest integer k such that no point in S^d lies in the interior of more than k spherical caps from $\{S_1, ..., S_n\}$.

[2] To the memory of William Thurston (1946 – 2012): After I explained to Bill that Dan (Spielman) and I successfully used conformal mapping in our spectral analysis of planar graphs and finite-element meshes [315], I cheerfully proclaimed, "Lifting $\mathbb{R}^d$ to the unit sphere (in $\mathbb{R}^{d+1}$) is brilliant!" as I thanked Bill again for the idea that I learned from him during our collaboration [252, 253] that has been so useful to my research. He took a long pause, looking pensive. "It's helpful for dealing with a concept that I don't understand," he finally said. "A concept? a Math concept? a Math concept you don't understand?" I shouted out. "Infinity," he replied quietly.

Proposition 5.24. For all neighborhood system Γ in $\mathbb{R}^d$ and for all stereographic projection $\Pi_{\mathbf{u}}$ from $\mathbb{R}^d$ to S_d, $\text{Ply}(\Gamma) = \text{Ply}(\Pi_{\mathbf{u}}(\Gamma))$. Furthermore, $\text{Ply}(\Gamma) = \text{Ply}(\Pi_{\mathbf{u}}^{-1} \circ \Pi(\Gamma))$.

Step 2. *Conformally Map Centerpoints*: Let $\mathbf{p}_i$ be the center of ball B_i in $\mathbb{R}^d$, and let $\mathbf{q}_i = \Pi(\mathbf{p}_i)$ be the stereographic image of $\mathbf{p}_i$ on S^d. For any $\delta \leq \frac{1}{d+2}$, by Theorem 5.4, the set of $(d+1)$-dimensional points $\Pi(P) = \{\mathbf{q}_1, ..., \mathbf{q}_n\}$ has a δ-centerpoint, located in the interior of the unit sphere S^d. Suppose $\mathbf{c}$ is a δ-centerpoint of $\Pi(P)$.

Miller *et al.* (Lemma 2.3.3.1 [252]) discovered the following remarkably simple formula to *conformally map a centerpoint to the center of the unit sphere* S^d. For $\alpha > 0$, let D_α be the dilation map of $\mathbb{R}^{d+1}$ by a factor of α. Bellow $||\mathbf{x}||$ denotes the Euclidean norm of $\mathbf{x}$.

Theorem 5.25 (Conformally Map Centerpoints)**.** For any $P \subset \mathbb{R}^d$, let $\mathbf{c}$ be a δ-centerpoint of $\Pi(P)$. Let:

$$\Phi = \Pi_{\mathbf{c}} \circ D_{\sqrt{(1-||\mathbf{c}||)/(1+||\mathbf{c}||)}} \circ \Pi_{\mathbf{c}}^{-1} \circ \Pi$$

Then, $\mathbf{0}$, the center of S^d, is a δ-centerpoint of $\Phi(P)$.

Step 3. *Random Projection by Great Circle*: The last step of the algorithm is simple. It selects a random *great circle* of S_d to define the separating sphere S. Recall that a great circle of S_d is where S_d intersects with a hyperplane passing through the center of S_d.

Here, it is more convenient to use the *dual representation* of great circles. Every vector $\mathbf{u}$ on S_d defines a great circle, namely, the great circle of the hyperplane containing $\mathbf{0}$ with normal vector $\mathbf{u}$. Note that $\mathbf{u}$ and $-\mathbf{u}$ defines the same great circle. Let $\text{GC}_{\mathbf{u}}$ denote the great circle associated with $\mathbf{u}$. To generate a random great circle, it is sufficient to generate a random point on S_d: For example, we can first generate $(d+1)$ i.i.d normally distributed numbers $q_1, ..., q_{d+1}$ (from $\mathcal{N}(0,1)$, and setting $\mathbf{u} = \mathbf{q}/||\mathbf{q}||$ where $\mathbf{q} = [q_1, ..., q_{d+1}]^T$. When we notate in

Matlab, we set $\mathbf{q} = randn(d+1); \mathbf{u} = \mathbf{q}/\text{norm}(\mathbf{q})$. Then, in the last step of the algorithm, we set the separating sphere as:

$$\begin{aligned} S &= \left(\Pi_{\mathbf{c}} \circ D_{\sqrt{(1-||\mathbf{c}||)/(1+||\mathbf{c}||)}} \circ \Pi_{\mathbf{c}}^{-1} \circ \Pi\right)^{-1} (\text{GC}_{\mathbf{u}}) \\ &= \Pi^{-1} \circ \Pi_{\mathbf{c}} \circ D_{\sqrt{(1+||\mathbf{c}||)/(1-||\mathbf{c}||)}} \circ \Pi_{\mathbf{c}}^{-1}(\text{GC}_{\mathbf{u}}). \end{aligned}$$

The following is a pseudo-code for the sphere-separator algorithm [252].

Algorithm: SphereSeparator(Γ)

Require: a k-ply system $\Gamma = \{B_1, ..., B_n\}$ in $\mathbb{R}^d$ with centers $P = \{\mathbf{p}_1, ..., \mathbf{p}_n\}$.

1: **Finding Centerpoint** : Compute a centerpoint $\mathbf{c}$ of $\Pi(P) = \{\Pi(\mathbf{p}_1), \ldots, \Pi(\mathbf{p}_n)\}$.

2: **Random Projection**: Compute a random point $\mathbf{u}$ on the sphere S_d by generating a d-dimensional Gaussian vector $\mathbf{q}$ and setting $\mathbf{u} = \mathbf{q}/||\mathbf{q}||$.

3: **Return the Pre-image of the Random Great Circle**:

$$S = \Pi^{-1} \circ \Pi_{\mathbf{c}} \circ D_{\sqrt{(1+||\mathbf{c}||)/(1-||\mathbf{c}||)}} \circ \Pi_{\mathbf{c}}^{-1}(\text{GC}_{\mathbf{u}})$$

4: **Partition by Sphere**: Return S (and the partition Γ by S: $\Gamma_{interior}(S)$, $\Gamma_{exterior}(S)$, and $\Gamma_{intersect}(S)$).

Step 4. *Analysis*: Each hemisphere of $\text{GC}_{\mathbf{u}}$ contains at most $(1-\delta)n$ points of $\Phi(P)$, since $\mathbf{0}$ is a δ-centerpoint of $\Phi(P)$. The pre-image of $\text{GC}_{\mathbf{u}}$ is S. So, the interior and exterior of S correspond to the two hemispheres of $\text{GC}_{\mathbf{u}}$, respectively. Thus, $|\Gamma_{interior}(S)|, |\Gamma_{exterior}(S)| \leq (1-\delta)n$ as desired. Let $C_i = \Phi(B_i)$ be the spherical cap image of B_i. By Proposition 5.24, $\text{Ply}(\Phi(\Gamma)) = \text{Ply}(\Gamma)$, where $\Phi(\Gamma) = \{C_1, ..., C_n\}$. Note that C_i intersects $\text{GC}_{\mathbf{u}}$ if and only if B_i intersects S. Thus:

$$\text{E}\left[|\Gamma_{intersect}(S)|\right] = \text{E}_{\mathbf{u}}\left[|\{i : \text{GC}_{\mathbf{u}} \cap C_i \neq \emptyset\}|\right].$$

There are most $2\text{Ply}(\Phi(\Gamma))$ caps that contains a hemisphere. Because $\text{Ply}(\Phi(\Gamma)) \leq n$, we have $\text{Ply}(\Phi(\Gamma)) \leq \text{Ply}(\Phi(\Gamma))^{\frac{1}{d}} n^{1-\frac{1}{d}}$. Without loss of generality, we assume $\Phi(\Gamma)$ does not contain such spherical caps. Then, the area of the base of C_i is $V_d r_i^d$, where r_i denotes the *base radius* of

C_i, and V_d denotes the volume of a unit d-dimensional ball. Let A_d be the surface area of a unit d-sphere in $\mathbb{R}^{d+1}$. It is well known that $A_{d-1} = dV_d$. The base of C_i has less area than the area of C_i. Thus:

$$\sum_i^n V_d r_i^d \leq \sum_i^n \text{Area}(C_i) \leq \text{Ply}(\Phi(\Gamma))A_d = \text{Ply}(\Gamma) \cdot A_d. \tag{5.2}$$

The following basic geometric property will be useful for bounding $\mathrm{E}_{\mathbf{u}}[|\{i : \mathrm{GC}_{\mathbf{u}} \cap C_i \neq \emptyset\}|]$.

Proposition 5.26 (Duality)**.** For any positive integer d, for any $\mathbf{u}, \mathbf{v} \in S_d$, $\mathbf{v} \in S_d \cap \mathrm{GC}_{\mathbf{u}}$ if and only if $\mathbf{u} \in S_d \cap \mathrm{GC}_{\mathbf{v}}$.

This duality implies that $\mathrm{GC}_{\mathbf{u}} \cap C_i \neq \emptyset$ iff $\mathbf{u}$ does not lie in the dual-image of C_i, denoted by $C_i^{\perp}$, which is a "belt-shaped" area of S_d that lies between a pair of parallel hyperplanes of distance $2r_i$ symmetric to the center: Let $\text{Area}(C_i^{\perp})$ denote the surface areas of $C_i^{\perp}$. Then:

$$\begin{aligned} \mathrm{E}_{\mathbf{u}}[|\{i : \mathrm{GC}_{\mathbf{u}} \cap C_i \neq \emptyset\}|] &= \mathrm{E}_{\mathbf{u}}\left[\left|\{i : \mathbf{u} \in C_i^{\perp}\}\right|\right] \\ &= \frac{1}{A_d}\left(\sum_{i=1}^n \text{Area}(C_i^{\perp})\right) \leq \sum_{i=1}^n \frac{2A_{d-1}}{A_d} r_i. \end{aligned} \tag{5.3}$$

The right-hand side of Eqn. (5.3) achieves maximum possible value under constraint (5.2), when all $\{r_i\}_{i \in V}$ are equal and largest possible. By Eqn. (5.2), in which case:

$$r_1 = \cdots = r_n \leq (\text{Ply}(\Gamma) \cdot A_d / V_d)^{\frac{1}{d}} \cdot n^{-\frac{1}{d}}$$

implying the right-hand side of Eqn. (5.3) is at most:

$$\left(\frac{2A_{d-1}}{A_d}\right) \cdot \left(\frac{A_d}{V_d}\right)^{\frac{1}{d}} \text{Ply}(\Gamma)^{\frac{1}{d}} n^{1-\frac{1}{d}} \leq 2d \cdot \text{Ply}(\Gamma)^{\frac{1}{d}} n^{1-\frac{1}{d}}.$$

The last inequality follows from $A_{d-1} = dV_d$. □

5.4 Discussion: Projection of High-Dimensional Data — Random vs Spectral

Random Projection

The sphere-separator algorithm uses random projection — a powerful tool for data analysis — to partition input geometric data. The conformal mapping of Theorem 5.22 reduces the partitioning of Γ to the partition of $\Phi(\Gamma)$ on the unit sphere S_d. Thus, the key algorithmic step of SphereSeparator(Γ) is a one-dimensional random projection. It uses a random vector $\mathbf{u}$ on S_d to define the partition of $\Phi(\Gamma)$.

By projecting the images of Γ's centers — $\{\mathbf{q}_1, ..., \mathbf{q}_n\} := \Phi(\{\mathbf{p}_1, ..., \mathbf{p}_n\})$ — onto $\mathbf{u}$, we partition $\Phi(\Gamma)$ into two sets:

$$\begin{aligned} \Phi^+_{\mathbf{u}}(\Gamma) &= \{C_i : \mathbf{u}^T \cdot \mathbf{q}_i \geq 0\} \\ \Phi^-_{\mathbf{u}}(\Gamma) &= \{C_i : \mathbf{u}^T \cdot \mathbf{q}_i < 0\}. \end{aligned}$$

Let $\Phi^{\text{intersect}}_{\mathbf{u}}(\Gamma) = \{C_i : C_i \cap \text{GC}_{\mathbf{u}} \neq \emptyset\}$. We can then define

$$\begin{aligned} \Phi(\Gamma_{interior}(\text{GC}_{\mathbf{u}})) &= \Phi^+_{\mathbf{u}}(\Gamma) - \Phi^{\text{intersect}}_{\mathbf{u}}(\Gamma) \\ \Phi(\Gamma_{exterior}(\text{GC}_{\mathbf{u}})) &= \Phi^-_{\mathbf{u}}(\Gamma) - \Phi^{\text{intersect}}_{\mathbf{u}}(\Gamma) \\ \Phi(\Gamma_{intersect}(\text{GC}_{\mathbf{u}})) &= \Phi^{\text{intersect}}_{\mathbf{u}}(\Gamma). \end{aligned}$$

The fact that $\mathbf{0}$ is a δ-centerpoint of $\Phi(P)$ guarantees that:

$$|\Phi(\Gamma_{interior}(\text{GC}_u u))|, |\Phi(\Gamma_{exterior}(\text{GC}_u u))| \leq (1-\delta)n.$$

The fact Φ is a conformal map guarantees that $S = \Phi^{-1}(\text{GC}_{\mathbf{u}})$ is a sphere in $\mathbb{R}^d$.

Thus, the sphere separator is induced by a one-dimensional random projection of the conformally mapped data.

> *Even without any optimization over the sphere S^d, random projection guarantees the expected size of $\Phi^{\text{intersect}}_{\mathbf{u}}(\Gamma)$ is at most $2d \cdot \text{Ply}(\Gamma)^{\frac{1}{d}} n^{1-\frac{1}{d}}$.*

Spectral Projection

The spectral partitioning method of Theorem 4.2 also uses a one-dimensional projection. It views the normalized Laplacian matrix $\mathcal{L}_{\mathbf{W}}$

as geometric data, whose column vectors form a set of n-dimensional data points.

The spectral partition is formulated by a one-dimensional projection of $\mathcal{L}_{\mathbf{W}}$, given by the eigenvector of $\mathcal{L}_{\mathbf{W}}$ associated with its second smallest eigenvalue. The quality of the partition, guaranteed by Cheeger's inequality, is characterized by this eigenvalue.

Theorem 4.8 further shows that other vectors can be effective for projection as well. The quality of the partition that they define can be characterized their Rayleigh quotients.

> *Although they appear to be very different, spectral partitioning and geometric partitioning are beautifully connected [315].*

This connection will be the subject of Section 5.8.

Dimension Reduction

Projection is widely used in data analysis. Just as sampling makes big data smaller, projection reduces the dimensionality of data. Random projection, in particular, provides a powerful tool for preserving important geometric quantities — such as distances and inner products — as guaranteed by Johnson-Lindenstrauss's celebrated theorem [180].

For a positive integer k and d, let $\mathbf{R}_{k,d}$ be a $k \times d$ random matrix where each entry is an i.i.d. standard Gaussian variable.

Theorem 5.27 (Johnson-Lindenstrauss)**.** There exists a constant c such that for any point set $P = \{\mathbf{p}_1, ..., \mathbf{p}_n\}$ in $\mathbb{R}^d$, $\epsilon \in (0, 1/2)$, and $\delta \in (0, 1)$, if $k = c \cdot \epsilon^{-2} \log \frac{n}{\delta}$, then with probability at least $1 - \delta$:

$$(1-\epsilon)||\mathbf{p}_i - \mathbf{p}_j|| \leq ||\mathbf{q}_i - \mathbf{q}_j|| \leq (1+\epsilon)||\mathbf{p}_i - \mathbf{p}_j||, \quad \forall i, j \in [n] \quad (5.4)$$

where $\mathbf{q}_i = \frac{1}{\sqrt{k}}\mathbf{R}_{k,d} \cdot \mathbf{p}_i$. If further $||\mathbf{p}_i|| \leq 1$, $\forall i$, then:

$$|\mathbf{p}_i^T \mathbf{p}_j - \mathbf{q}_i^T \mathbf{q}_j| \leq \epsilon, \quad \forall i, j \in [n] \quad (5.5)$$

Theorem 5.27 implies that the two data models for defining network affinities discussed in Section 2.2 have the following properties:

Proposition 5.28 (Low-Rank Feature Model)**.** For any non-negative $n \times n$ symmetric matrix $\mathbf{W}$, $\exists$ a feature space $\mathcal{F} = (V, \mathbf{F})$ where $\mathbf{F} \in \mathbb{R}^{n \times k}$ with $k = O(\epsilon^{-2} \log n)$ such that:

$$(1-\epsilon)w_{i,j} - \epsilon \leq \mathbf{F}[i,:]^T \cdot \mathbf{F}[j,:] \leq (1+\epsilon)w_{i,j} + \epsilon, \quad \forall i \neq j \in [n],$$

Proposition 5.29 (Low-Rank Metric Model)**.** Suppose a network $G = (V, E, \mathbf{W})$ is defined by a geometric embedding $\{\mathbf{x}_v \in \mathbb{R}^n : v \in V\}$ of G, i.e.:

$$w_{u,v} = (||\mathbf{x}_u - \mathbf{x}_v||_2)^{-\alpha}, \text{ for a positive } \alpha > 0.$$

Then, there exists a k-dimensional embedding $\{\mathbf{y}_v : v \in V\} \subset \mathbb{R}^k$ of G with $k = O(\epsilon^{-2} \log n)$, such that:

$$(1-\alpha\epsilon) \cdot w_{u,v} = (||\mathbf{y}_u - \mathbf{y}_v||_2)^{-\alpha} \leq (1+\alpha\epsilon)w_{u,v}.$$

A comprehensive survey of random projections can be found in [343]. Note that both spectral and geometric partitioning use much more aggressive dimension reduction than the low-distortion Johnson-Lindenstrauss projection. Both partitioning methods reduce high-dimensional data all the way to one dimension.

When working with search and information retrieval applications, one should preserve the distances or inner products between all data points. Such a strong geometric guarantee naturally has its cost — it requires $\Omega(\epsilon^{-2} \log n / \log \frac{1}{\epsilon})$ dimensions [10].

However, for many optimization problems, such as graph partitioning, the objective is not to approximately preserve the distances or inner products among all data points. Thus, much more aggressive dimension reduction is often possible.

5.5 Scalable Geometric Divide-and-Conquer

We will now illustrate the use of sphere separators in scalable divide-and-conquer algorithms for geometric data. Our main example is k-NNG construction.

Suppose $P = \{\mathbf{p}_1, ..., \mathbf{p}_n\} \subset \mathbb{R}^d$ and integer $k > 0$ are the input. We will analyze the following simple k-NNG algorithm, that also computes the neighborhood system

$$\Gamma_k(P) = \{B_1^{(k)}, \ldots, B_n^{(k)}\}.$$

Algorithm: k-NNGBySphereSeparator(P, k)

1: **Base Case**: If $|P| \leq k+1$, return $((B_1, ..., B_{|P|}), K_{|P|})$, where each $B_i = \mathbb{R}^d$, and $K_{|P|}$ is a clique of $|P|$ nodes.
2: **Divide**: P into two arbitrary "equal-sized" subsets P_0 and P_1.
3: **Conquer**: $\forall i \in \{0, 1\}$, $(\Gamma_i, G_i) =$ k-NNGBySphereSeparator(P_i, k).
4: **Combine**: Build a scalable binary-search data structure for point location in Γ_0 and Γ_1 using geometric separators (see below).
5: **for** $i \in \{0, 1\}$ **do**
6: Update balls in Γ_i and edges in G_i by locating points in $P - P_i$ in the structure for Γ_i (see below). Call the updated neighborhood system $\bar{\Gamma}_i$ and the updated graph $\bar{G}_i$.
7: Return $(\bar{\Gamma}_0 \cup \bar{\Gamma}_1, \bar{G}_0 \cup \bar{G}_1)$.

The base case, divide, and conquer steps 1-3 are straightforward. In the "combine" steps 4-6, we solve the following point-location problem.

Definition 5.30 (Point Location in Neighborhood Systems)**.** Given a neighborhood system Γ, construct a data structure to answer the following query: Output all balls in Γ that contain a query point $\mathbf{p}$.

Partitioning by spheres offers a natural framework for *binary search* in high dimensions: Each sphere can be represented with $O(d)$ space, and it takes only $O(d)$ time to determine whether a point in $\mathbb{R}^d$ is in the interior or the exterior of a sphere. We will now use sphere separators to construct a binary-search data structure for the point location problem given in Definition 5.30.

This data structure is a binary tree. Its root stores a sphere separator $S_\emptyset$ for answering the first query of the binary search. $S_\emptyset$ guarantees $|\Gamma_{exterior}(S_\emptyset)|, |\Gamma_{interior}(S_\emptyset)| \leq (1 - \delta)n$ and $|\Gamma_{intersect}(S_\emptyset)| =$

$O(\text{Ply}(\Gamma)^{\frac{1}{d}} n^{1-\frac{1}{d}})$. To recursively build the binary-search structure, let $\Gamma_0 = \Gamma_{exterior}(S_\emptyset) \cup \Gamma_{intersect}(S_\emptyset)$ and $\Gamma_1 = \Gamma_{interior}(S_\emptyset) \cup \Gamma_{intersect}(S_\emptyset)$. We then recursively build binary search structures for Γ_0 and Γ_1, respectively, starting with their sphere separators S_0 and S_1. The recursion stops when the subset is smaller than $\Theta(\text{Ply}(\Gamma))$. The height of this binary search tree of sphere separators is $O(\frac{\log n}{\delta})$.

To locate all balls that cover a query point $\mathbf{p} \in \mathbb{R}^d$, we first check $\mathbf{p}$ against $S_\emptyset$. If $\mathbf{p}$ is in the exterior of $S_\emptyset$, we follow the binary search to the subtree for Γ_0. If $\mathbf{p}$ is inside or on $S_\emptyset$, we follow the subtree for Γ_1. When the search reaches a leaf, we check $\mathbf{p}$ against all balls associated with the leaf and outputs all those that contain $\mathbf{p}$. As the height of the search tree is $O(\frac{\log n}{\delta})$ and each leaf contains $O(\text{Ply}(\Gamma))$ balls, the query time of this data structure is $O(\frac{\log n}{\delta}) + \text{Ply}(\Gamma))$. The time and space to build this binary search structure is upper bounded by:

$$T(n) \leq \begin{cases} 1 & \text{if } n = \Theta(\text{Ply}(\Gamma)) \\ T\left(\delta n + O\left(\text{Ply}(\Gamma)^{\frac{1}{d}} n^{1-\frac{1}{d}}\right)\right) + T((1-\delta)n) + O(n) & \text{otherwise.} \end{cases}$$

$$S(n) \leq \begin{cases} O(dn) & \text{if } n = \Theta(k) \\ S\left(\delta n + O(\text{Ply}(\Gamma)^{\frac{1}{d}} n^{1-\frac{1}{d}})\right) + S((1-\delta)n) + O(d) & \text{otherwise} \end{cases}$$

For constant d and $\text{Ply}(\Gamma) = o(n)$, $T(n) = O(n \log n)$ and $S(n) = O(n)$.

Theorem 5.31 (Scalable High-Dimensional Binary Search)**.** For any fixed d, and neighborhood system $\Gamma \subset \mathbb{R}^d$, we can build a binary search tree, using $O(n)$ space in $O(n \log n)$ time, to answer any point location query regarding Γ in query time $O(\log n + \text{Ply}(\Gamma))$.

Thus, the data structure in Step 4 of k-NNGBySphereSeparator can be constructed in $O(n \log n)$ time. Because $\text{Ply}(\Gamma_i) = O(k)$, for $i \in \{0, 1\}$, the query time is bounded $O(\log n + k)$. In Step 6, to update balls in Γ_i and edges in G_i, we make $n/2$ queries, one for each point in $\mathbf{u} \in P - P_i$, in order to identify every point $\mathbf{q} \in P_i$ whose ball $B_\mathbf{q}$ in Γ_i contains $\mathbf{u}$. There are at most $O(k)$ such points.

For each such point $\mathbf{q} \in P_i$, let $N_{G_i}(\mathbf{q})$ denote $\mathbf{q}$'s neighbors in G_i. There are two cases:

1. If $|N_{G_i}(\mathbf{q})| = k$ (the typical case), then we replace $\mathbf{q}$'s neighbors with the k closest points from $N_{G_i}(\mathbf{q}) \cup \{\mathbf{u}\}$. We also update the radius of $\mathbf{q}$'s ball in Γ_i. The new radius is equal to the distance between $\mathbf{q}$ and its k^{th} nearest neighbors in $N_{G_i}(\mathbf{q}) \cup \{\mathbf{u}\}$.

2. If $|N_{G_i}(\mathbf{q})| < k$ (near the end of the divide-and-conquer), we simply add $\mathbf{u}$ to the neighbors of $\mathbf{q}$.

Thus, Step 6 can be implemented in $O(n(\log n + k))$ time. Putting it all together, the time that k-NNGBySphereSeparator(P, k) needs is upper bounded by the following recurrence:

$$T_{kNNG}(n) \leq \begin{cases} 1 & \text{if } n \leq k \\ 2 \cdot T_{kNNG}(n/2) + O(n(k + \log n)) & \text{otherwise} \end{cases}$$

Thus, k-NNGBySphereSeparator(P, k) runs in $O(n(k+\log n) \log n)$ time.

Discussion

Using a careful probabilistic analysis, Frieze *et al.* [139] proved that a variation of k-NNGBySphereSeparator can be implemented in random $O(n(k + \log n))$ time, matching the optimal time bound achieved by Vaidya [338]. Although Vaidya's algorithm is deterministic, it uses a more complex data structure of k-NNGBySphereSeparator. Based on binary search, Frieze *et al.*'s algorithm is also suitable for parallel implementation, providing the first optimal parallel algorithm for constructing k-NNGs. It runs in random $O(\log n)$ parallel time, with $O(n(k + \log n))$ total work. Late, Bern *et al.* [51] and Callahan and Kosaraju [78] developed optimal deterministic parallel k-NNG algorithms based on quadtrees and their high-dimensional generalization.

5.6 Graph Partitioning: Vertex and Edge Separators

The study of sphere separators and geometric partitioning is part of a larger research program commonly known as *graph partitioning*. It addresses the fundamental problem of dividing a graph into two or more non-overlapping subgraphs. In addition to geometric applications, graph partitioning is widely used in VLSI layout, parallel computing,

numerical methods, protein folding, data clustering, and combinatorial optimization. One can divide a graph either by removing a subset of vertices or by removing a subset of edges.

Vertex Separators

Central to the first approach is the concept of *vertex separators*, popularized by Lipton-Tarjan's *Planar Separator Theorem* [236].

Definition 5.32 (Vertex Separators)**.** Let $G = (V, E)$ be an undirected, unweighed graph. For a constant $\alpha \in [1/2, 1)$, $C \subset V$ is an *α-vertex separator* of G if the removal of C (and all edges incident to C) leaves G with no connected component of size more than $\alpha|V|$.

Lipton and Tarjan [236] proved the following remarkable theorem.

Theorem 5.33 (Planar Separators)**.** Every n-vertex planar graph has a $\frac{2}{3}$-vertex separator of size $O(\sqrt{n})$.

The existence of vertex separators turns out to be a broad phenomenon in graph theory: As proved by Gilbert, Hutchinson, and Tarjan [147], every graph whose genus is bounded by g has a $\frac{2}{3}$-vertex separator of size $O(\sqrt{gn})$. Moreover, as proved by Alon, Seymour, and Thomas [13], every graph that excludes h-clique as graph minors has a $\frac{2}{3}$-vertex separator of size $O(h^{\frac{3}{2}}\sqrt{n})$.

Theorem 5.22 can be viewed as a geometric generalization of Lipton-Tarjan's Separator Theorem [236]. In fact, we can directly derive the Lipton-Tarjan's theorem from Theorem 5.22 as the following:

Proof. (A geometric proof of Theorem 5.33) We call a set of two-dimensional disks $\Gamma = (D_1, \ldots, D_n)$ with disjointed interiors a *disk packing*. Thus, each disk packing is a 1-ply neighborhood system in two dimensions. Let the *nerve* of a disk packing Γ, nerve(Γ), be the graph $G = (V, E)$, where $V = [n]$, and E contains exactly the pairs

of (i, j) that D_i and D_j touch. Clearly, nerve(Γ) is a planar graph. Remarkably, Koebe [213] proved that the converse is true.

Theorem 5.34 (Koebe). Every triangulated planar graph G has a disk packing $\Gamma = (D_1, \ldots, D_n)$, whose nerve($\Gamma$) is isomorphic to G.

By Theorem 5.22, $\exists$ a circle S in $\mathbb{R}^2$ such that $|\Gamma_{intersect}(S)| \leq 2\sqrt{n}$ and $|\Gamma_{interior}(S)|, |\Gamma_{exterior}(S)| \leq 3n/4$. Thus, $\Gamma_{intersect}(S)$ induces a $(3/4)$-vertex separator of G of size $2\sqrt{n}$. By applying this argument to the larger of the two partitions (if necessary), one can then obtain a $(\frac{2}{3})$-vertex separator of size $O(\sqrt{n})$, as stated in Theorem 5.33. □

The sphere-separator theorem, from Koebe's viewpoint, generalizes Lipton-Tarjan's separator theorem from two-dimensional planar graphs to higher dimensional geometric graphs. Graphs in three and higher dimensions arise in scientific simulation and computer graphics. Dimensionality reduction of input data also produces three and higher dimensional graphs. Theorem 5.22 suggests that dimensional reduction may help to reveal geometric clusters of data along its principle dimensions.

Definition 5.35 (Intersection Graphs). Let $\Gamma = \{B_1, \ldots, B_n\}$ be a neighborhood system in $\mathbb{R}^d$. The *intersection graph* of Γ, $G(\Gamma)$, is the undirected graph with vertices $V = [n]$ and edges:

$$E = \{(B_i, B_j) : B_i \cap B_j \neq \emptyset.\}$$

Directly from Theorem 5.22, we have:

Theorem 5.36 (Vertex Separators for Intersection Graphs). For any neighborhood system $\Gamma = \{B_1, ..., B_n\}$ in $\mathbb{R}^d$, the intersection graph $G(\Gamma)$ has a $\frac{d+1}{d+2}$-vertex separator of size $O(\text{Ply}(\Gamma)^{\frac{1}{d}} n^{1-\frac{1}{d}})$.

Algorithmically, such a separator can be computed by a scalable algorithm. The parameter Ply(Γ) characterizes the geometric locality

and separability of data's neighborhoods. For example, the k-NNG of a set of point P in $\mathbb{R}^d$ is defined by its k^{th}-nearest neighborhood system $\Gamma_k(P)$, and $\text{Ply}(\Gamma_k(P)) = \Theta(k)$ (Proposition 5.20). Thus:

Corollary 5.37. For any point set $P = \{\mathbf{p}_1, ..., \mathbf{p}_n\} \subset \mathbb{R}^d$, and integer k, the k-NNG of P has a $\frac{d+1}{d+2}$-vertex separator of size $O(k^{\frac{1}{d}} n^{1-\frac{1}{d}})$.

Scalable Approximation of Covering and Independent Sets

As a tool for network divide-and-conquer, sublinear-sized vertex separators are extremely useful for designing scalable approximation algorithms. Below, we use vertex covering as an example. The result directly extends to independent sets.

To illustrate, consider an example of environmental engineering: Suppose there is a set of n sensors — for monitoring earthquakes, nuclear hazards, and traffics — located at $P = \{\mathbf{p}_1, ..., \mathbf{p}_n\} \in \mathbb{R}^3$. Suppose that due to certain economic constraints (e.g., the sequestration imposed by Congress), members of the organization must cut back the number of sensors that they maintain and monitor. Thus, they need to solve the optimization problem, which determines the smallest subsets of these sensors that can achieve an acceptable monitoring quality. They also need to solve this problem fast. Consider one such quality condition: A subset $Q \subset P$ is an *effective covering set* if for every $\mathbf{p}_i \in P$, Q contains at least one of its 8 nearest points in Q.

> *Geometric separators can be used to scalably construct a very high quality covering set.*

The method is very simple: First, we use a divide-and-conquer method to compute $\Gamma_8(P)$, the 8^{th}-nearest neighborhood system of P. We then recursively use the scalable geometric separator algorithm to divide the intersection graph of $\Gamma_8(P)$ into components of size $\lceil \log \log n \rceil$. Then, we use exhaustive search on each component to find its optimal effective covering set, and return the union of these covering sets together with all points in the recursive separators (whose

8-nearest neighbors do not intersect this union). This algorithm can be implemented in time:

$$O(n \log n + (n/\log\log n) \cdot 2^{\log\log n}) = O(n \log n).$$

Lemma 5.38 (Quality of the Covering)**.** Let $\mathrm{OPT}(P)$ denote the size of the optimal covering set for P. Then, the size of the covering set returned by the algorithm above is at most $|\mathrm{OPT}(P)|(1 + o(1))$.

Proof. (Sketch) First note that $\mathrm{OPT}(P) \geq \frac{1}{12} \cdot \frac{1}{8}n$, where 12 is the kissing number in three dimensions. The size of the recursive separators is upper bounded by the recurrence:

$$S(n) \leq \begin{cases} O(1) & \text{if } n \leq \log\log n \\ 2S(\frac{n}{2}) + O(n^{\frac{2}{3}}) & \text{otherwise.} \end{cases}$$

The solution of this recurrence is $S(n) = O(n/(\log\log n)^{\frac{1}{3}}) = o(n)$. After removing the recursive separators, for each component C, our algorithm selects no more than $|C \cap \mathrm{OPT}(P)|$ sensors. Because these components are non-overlapping, the cover set returned by the algorithm has at most $|\mathrm{OPT}(P)|(1 + o(1))$ sensors. □

Edge Separators

Central to the second graph partitioning approach is the concept of *edge separators*: A subset of edges $E' \subset E$ is an *α-edge separator* of G if the removal of E leaves the resulting graph $(V, E - E')$ with no connected component of size more than $\alpha|V|$.

For any graph with small degrees, we can convert a vertex separator into an edge separator. For example, Lipton-Tarjan's separator theorem implies that any planar graph with maximum degree bounded by Δ has a $\frac{2}{3}$-edge separator of size $O(\Delta\sqrt{n})$, because the collection of all edges incident to the $\frac{2}{3}$-vertex separator defines a $\frac{2}{3}$-edge separator. The maximum degree of k-NNGs in $\mathbb{R}^d$ is at most $O(k)$. Thus:

Theorem 5.39 (Edge Separators for k-NNGs)**.** For any point set $P = \{\mathbf{p}_1, ..., \mathbf{p}_n\} \subset \mathbb{R}^d$, and integer k, the k-nearest neighbor graph of P has a $\frac{d+1}{d+2}$-edge separator of size $O(k^{1+\frac{1}{d}} n^{1-\frac{1}{d}})$.

For planar graphs, Gazit and Miller [142] went a step further to prove the following theorem:

Theorem 5.40 (Planar Edge Separators)**.** Every planar graph G has a $\frac{2}{3}$-edge separator of size $1.58\sqrt{\sum_i^n d_i^2}$, where d_i is the degree of the i^{th} vertex.

Motivated by Gazit-Miller's result, Spielman and Teng [312] proved the following edge separator theorem.

Theorem 5.41 (Edge Separators for Geometric Graphs)**.** The intersection graph of any d-dimensional neighborhood system $\Gamma = \{B_1, \ldots, B_n\}$ has a $\frac{d+1}{d+2}$-edge separator of size:

$$O\left(\mathrm{Ply}(\Gamma)^{\frac{1}{d}} \cdot \left(\sum_{i=1}^{n} d_i^{\frac{d}{d-1}}\right)^{1-\frac{1}{d}}\right).$$

Proof. Using Eqn. (5.3) for intersection graphs, we can upper bound the expected size of the edge-separator defined by random great circles (in the proof of Theorem 5.22) by:

$$\sum_{i=1}^{n} d_i r_i.$$

Thus, the application of Lagrange's method immediately leads to above theorem. □

When specialized to two dimensions, this approach proves that every planar graph G has a $\frac{3}{4}$-edge separator of size $\sqrt{\sum_i^n d_i^2}$.

5.7 Multiway Partition of Network and Geometric Data

In algorithm design and data clustering, one usually partitions a network or data set into multiple components. For example, our scalable algorithm for vertex covering (Section 5.6) uses multiway partitioning. In this section, we will analyze the clusterability of low-dimensional data, and prove Theorem 5.2.

We start with some notation. For networks, multiway partitions can be defined by removing either edges or nodes from the network. For an integer $t > 0$, a *t-way partition* of a graph $G = (V, E)$ is a partition of V into t subsets $(V_1, ...V_t)$. A *t-way vertex partition* is a partition of V into $t + 1$ subsets $(S, V_1, ...V_t)$ such that $\forall i, j \in [t]$, V_i and V_j have no edge between them. The set S is the separator.

For $c \geq 1$, we say the t-way partition is *c-balanced* if $\forall i \in [t]$, $|V_i| \leq c \cdot \lceil n/t \rceil$. The *cut-size* of the t-way partition, denoted by $\text{cut}(V_1, ..., V_t)$, is measured by the total number of edges whose endpoints are in the different parts of the partition. The *cut-size* of the t-way vertex partition $(S, V_1, ...V_t)$ is $|S|$.

Edge and vertex separators can be recursively applied to produce balanced multiway partitions [306]. To be precise, let us consider the following family of graphs: For constants $\alpha < 1$ and $\delta \in [1/2, 1)$, we say a graph G is *closed under (α, δ)-separators* if it has the following property: Every subgraph H of G has a δ-edge separator (or δ-vertex separator) of size $O(|V_H|^\alpha)$.

For this family of graphs, one can directly estimate the quality of the resulting multiway partition.

Proposition 5.42 (Recursive Multiway Partitioning)**.** Suppose G is closed under (α, δ)-separators. Then, for any integer $t > 0$ and constant $c \geq 1$, the recursive applications of separators can produce a t-way partition of balance c and cut-size $O(t^{1-\alpha} n^\alpha)$.

For example, because the k-NNG of any n points in $\mathbb{R}^d$ is closed under $(1-\frac{1}{d}, \frac{d+1}{d+2})$-separators, the k-NNG has a balanced t-way partition of cut-size $O(t^{\frac{1}{d}} k^{1+\frac{1}{d}} n^{1-\frac{1}{d}})$.

In fact, by a more careful argument [206], one can ensure that the boundary of every subset of the balanced t-way partition in Proposition 5.42 is small, particularly bounded by $O((\frac{n}{t})^{\alpha})$. This result of Kiwi *et al.* [206] implies Theorem 5.2.

Let's introduce one more notation. For a t-way partition $(V_1, ..., V_t)$ of $G = (V, E)$, let $\partial(V_i)$ denote the number of the *boundary edges* of V_i in G, that is, the set of edges with exactly one endpoint in V_i. Let:

$$\text{max-boundary}(V_1, ..., V_t) = \max_{i \in [t]} \partial(V_i).$$

In general, when given $c \geq 1$ and integer $t > 0$, the Min-Max-Boundary Multiway Partitioning problem [206] is to find a c-balanced t-way partition with minimum max-boundary.

Theorem 5.43 (Min-Max-Boundary Multiway Partitioning)**.** Suppose G is closed under (α, δ)-separators. Then, for any integer $t > 0$, careful recursive applications of edge-separators can produce a t-way partition $(V_1, ..., V_t)$ of balance 2, such that:

$$\text{max-boundary}(V_1, ..., V_t) = O\left(\left(\frac{n}{t}\right)^{\alpha}\right).$$

Proof. (Sketch) For a subset $S \in V$, let $\partial_V(S)$ be the set of *boundary nodes* in S that are endpoints of edges connecting S and $\bar{S}$. We now define the meaning of "careful recursive applications of edge-separators" in the statement of the theorem. We first use a small edge-separator to divide the graph into V_0 and V_1. To recursively divide V_i, for $i \in \{0, 1\}$, we use a small edge-separator that "simultaneously" divides both V_i and $\partial_V(V_i)$ into balanced subsets. We can prove such an edge-separator exists by another recursive application of regular edge-separators. We then recursively apply this "careful" divide-and-conquer. □

Algorithmically, the "careful" recursive application is scalable, provided the problem of finding a δ-edge separator of size $O(|V_H|^{\alpha})$ is scalable for any subgraph H of G. Theorem 5.2 is a direct consequence of Theorem 5.43:

Proof. (Sketch of Theorem 5.2) Applying Theorem 5.43 with the geometric separator theorem to the k-NNG of P, we obtain a 2-balanced partition $P_1, ..., P_t$ such that for all $i \in [t]$, the number of boundary edges involving points of P_i is at most

$$O(k^{1+\frac{1}{d}}|P_i|^{1-\frac{1}{d}}).$$

The total number of k-nearest-neighbor relations involving P_i is $O(k|P_i|)$. Thus, the fraction of the k-nearest-neighbor relations of P_i involving points outside P_i is at most $O\left(\left(\frac{k}{|P_i|}\right)^{\frac{1}{d}}\right) = o(1)$, as stated in the theorem. □

The multiway partitioning algorithm of this proof is scalable. Theorem 5.2 highlights the fundamental differences between "big" geometric data in high but constant dimensions (e.g., $d = 100$) and those in logarithmic dimensions, as required by the theory of Johnson-Lindenstrauss dimension reduction [180].

5.8 Spectral Graph Partitioning: The Geometry of a Graph

One may view the geometric separator theorem as a combinatorial characterization of geometric structures [326]. The converse characterizations highlighted by Koebe's beautiful theorem of planar graphs [213] features the geometric essence of graphs.

The *geometry* of a graph emerges naturally in practical applications such as graph drawing, VLSI layout, and network visualization [45]. It is also intrinsic in combinatorial optimization. For example, mathematical programming usually induces a metric structure over the input graph when solving optimization problems, such as matching, flows, and partitions in a network [224, 151].

Koebe's characterization uses the embedding framework that assigns each node a Cartesian coordinate:

Definition 5.44 (Cartesian Embedding of Graphs)**.** A *Cartesian embedding* of a graph $G = (V, E)$ in d dimensions maps each vertex $i \in V$ to a vector $\mathbf{v}_i \in \mathbb{R}^d$.

In this section, with graph partitioning as a background, we review *spectral graph embedding*, which is a powerful algebraic framework for studying the geometry of a graph. Suppose G is an unweighted and undirected graph. Recall that the Laplacian matrix of G is given by:

$$\mathbf{L}_G = \mathbf{D}_G - \mathbf{A}_G$$

where $\mathbf{A}_G$ is the adjacency matrix of G, and $\mathbf{D}_G$ is the diagonal matrix of vertex degrees in G. Because $\mathbf{L}_G$ is symmetric, all of its n eigenvalues are real. In this section, we use $\lambda_1(G) \leq ... \leq \lambda_n(G)$ to denote the n eigenvectors of $\mathbf{L}_G$. We use $\mathbf{u}_1(G), ..., \mathbf{u}_n(G)$ denote the eigenvectors of $\mathbf{L}(G)$ associated with eigenvalues $\lambda_1(G), \cdots, \lambda_n(G)$, respectively. We drop the explicit reference to G if it is clear from the context. If G has n vertices, then $\forall \mathbf{x} \in \mathbb{R}^n$:

$$\mathbf{x}^T \mathbf{L} \mathbf{x} = \sum_{(i,j) \in E} (x_i - x_j)^2.$$

Thus, $\lambda_1 = 0$, and $\mathbf{1}$ is an eigenvector associated with λ_1. Fiedler [134] proved that $\lambda_2 > 0$ iff G is connected. We will refer to $\lambda_2(G)$ as the *Fiedler value* of G and its associated eigenvector as a *Fiedler vector*. The eigenvectors of G provide a geometric interpretation of G:

Definition 5.45 (Spectral Graph Embedding)**.** For an undirected graph $G = (V, E)$ with n vertices, its *spectral embedding* is a Cartesian embedding of G given by the map $\sigma : V \to \mathbb{R}^{n-1}$:

$$\forall v \in [n], \quad \sigma(v) := [\mathbf{u}_2[v], ..., \mathbf{u}_n[v]] \tag{5.6}$$

In general, for a k-index tuple $\mathbf{j} = (j_1, ..., j_k) \in [2 : n]^k$, the $\mathbf{j}$-*spectral embedding* of G is given by the map $\sigma_{\mathbf{j}} : V \to \mathbb{R}^k$:

$$\forall v \in [n], \quad \sigma_{\mathbf{j}}(v) = [\mathbf{u}_{j_1}[v], ..., \mathbf{u}_{j_k}[v]] \tag{5.7}$$

Spielman and Teng [315] used the following simple fact to connect graph spectra with Cartesian embeddings.

Lemma 5.46 (Fiedler Value and Cartesian Embedding)**.** For any undirected graph $G = (V, E)$ and for any positive integer d:

$$\lambda_2(G) = \min \frac{\sum_{(i,j)\in E} ||\mathbf{v}_i - \mathbf{v}_j||_2^2}{\sum_{i=1}^{n} ||\mathbf{v}_i||^2} \tag{5.8}$$

where the minimum is taken over all Cartesian embeddings $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\} \subset \mathbb{R}^d$ such that $\sum_{i=1}^{n} \mathbf{v}_i = \mathbf{0}$.

While we can measure the quality of graph embedding in various ways, the right-hand side of Equation (5.8) provides a "spectral-quality" of the embedding.

Geometric techniques can be applied to the spectral embedding to partition graphs. Indeed, *spectral graph partitioning* was introduced by Hall, who applied quadratic programming to design a placement algorithm [162], and by Donath and Hoffman, who used the eigenvectors of the adjacency matrix to partition its graph [115, 116].

In real-world applications — including VLSI design, parallel computing, and image processing — spectral methods have become one of the most successful heuristics for partitioning graphs and matrices [278].

Below, we discuss a result of Spielman and Teng [315], which shows that Fiedler vectors can be used to produce separators of provably good quality for planar graphs, finite-element meshes, and k-NNGs. These graphs are widely used in VLSI layout and scientific computing.

Theorem 5.47 (Spectral Partitioning)**.** For any n-vertex planar graph G with maximum degree Δ:

$$\lambda_2(G) \leq \frac{8\Delta}{n} \tag{5.9}$$

Thus, by Cheeger's inequality, when we apply Sweep to the Fiedler vector, we obtain a subset of conductance $O(\frac{1}{\sqrt{n}})$.

In addition, repeated Cheeger's Sweeps produce a $\frac{1}{2}$-edge separator of size $O(\sqrt{n})$ for every bounded-degree planar graph.

The proof of [315] uses the conformal maps introduced in Section 5.3 to bound the Fiedler values of these graphs. It then applies Cheeger's inequality to determine partitions. In fact, a slightly weaker statement of Theorem 5.47, that $\lambda_2(G) \leq O(\frac{\Delta}{n})$, follows directly from Koebe's embedding (Theorem 5.34), Theorem 5.25 and Proposition 5.46.

To obtain the constant 8 in the Big-O notation, the following elegant geometric theorem is used in [315]. To state the theorem, recall from Section 5.3 that: (1) for $\alpha > 0$, D_α denote the dilation map of $\mathbb{R}^{d+1}$ by a factor of α. (2) For any vector $\mathbf{u} \in \mathbb{R}^{d+1}/\mathbf{0}$, $\mathbf{h_u}$ denote the d-dimensional hyperplane passing through $\mathbf{0}$ with normal vector $\mathbf{u}/||\mathbf{u}||_2$. (3) $\Pi_\mathbf{u}$ denote the stereographic projection that maps $\mathbf{h_u}$ to S^d and let $\Pi_\mathbf{u}^{-1}$ be its inverse.

Theorem 5.48 (Conformally Mapping of Center of Mass)**.** Suppose $\Gamma = \{B_1, ..., B_n\}$ is a neighborhood system in $\mathbb{R}^d$. Then, there exist $\mathbf{u} \in \mathbb{R}^{d+1}$ and $\alpha > 0$ such that the conformal map $\Phi = \Pi_\mathbf{c} \circ D_\alpha \circ \Pi_\mathbf{c}^{-1} \circ \Pi$ has the following property:

$$\sum_i \mathbf{center}(\Phi(B_i)) = \mathbf{0} \tag{5.10}$$

where $\mathbf{center}(\Phi(B_i))$ denotes the center of the spherical cap $\Phi(B_i)$ on unit d-sphere S_d.

This geometric theorem can be proved using Brouwer's Fixed-Point Theorem (See [315] for details).

Proof. (of Theorem 5.47): As in Section 5.6, we first apply Koebe's embedding (Theorem 5.34) to planar graph G to obtain a disk-packing Γ. We then apply Theorem 5.48 to Γ. We thus obtain a Cartesian embedding of the planar graph onto the unit sphere S^2: Suppose D_i is the disk of vertex i in the Koebe embedding, and let $\mathbf{v}_i = \mathbf{center}(\Phi(D_i))$. By Theorem 5.48, the center of S^d, $\mathbf{0}$, is the center of mass of the spherical centers of $\Phi(\Gamma)$. Thus, we have $\sum_{i=1}^n \mathbf{v}_i = \mathbf{0}$. Furthermore, for all $i \in V$, $||\mathbf{v}_i||_2 = 1$. Thus:

$$\sum_i ||\mathbf{v}_i||_2^2 = n \tag{5.11}$$

Let r_i be the radius of the base disk of spherical cap $\Phi(D_i)$. Because $\Phi(D_1), ..., \Phi(D_n)$ do not overlap, $\sum_i \pi \cdot r_i^2 \leq 4 \cdot \pi$, which implies that:

$$\sum_{(i,j)\in E} ||\mathbf{v}_i - \mathbf{v}_j||^2 \leq 2\Delta \sum_{i=1}^{n} r_i^2 \leq 8\Delta \tag{5.12}$$

The statement of the theorem then directly follows from Proposition 5.46 using Eqn. (5.11) and Eqn. (5.12). It then follows from Cheeger's inequality (Theorem 4.24) that for a planar $G = (V, E)$ with constant maximum degree, one can obtain a subset $S \subset V$ with $|S| \leq n/2$ and $\frac{\text{cut}(S,\bar{S})}{|S|} \leq O(\frac{1}{\sqrt{n}})$ by sweeping the Fiedler vector. □

As both Theorem 5.48 and Proposition 5.46 hold for all dimensions, the proof above directly extends to high dimensions. Therefore:

Corollary 5.49 (Fiedler Values of k-NNGs)**.** Suppose G is the intersection graph of a neighborhood system $\Gamma = \{B_1, ..., B_n\}$ in $\mathbb{R}^d$, and Δ is the maximum degree of G. Then:

$$\lambda_2(G) \leq O\left(\Delta \cdot \left(\frac{\text{Ply}(\Gamma)}{n}\right)^{\frac{2}{d}}\right) \tag{5.13}$$

In particular, for any point set $P = \{\mathbf{p}_1, ..., \mathbf{p}_n\} \subset \mathbb{R}^d$, for any k, the Fiedler value of the k^{th}-NNG of P is upper bounded by $O\left(k \cdot \left(\frac{k}{n}\right)^{\frac{2}{d}}\right)$.

In [196], Kelner extended Theorem 5.47 to graphs with bounded genus. He proved that for any graph G with genus g and maximum degree Δ, $\lambda_2(G) = O(\frac{(g+1)\cdot\Delta}{n})$. Like [315], Kelner's proof is geometric. The first combinatorial approach for analyzing the Fiedler value was given by Biswal, Lee, and Rao [52]. They further extended Theorem 5.47 to graphs characterized by the size of "forbidden minors." In particular, they proved that if a graph G with maximum degree Δ and has no K_h as minor,[3] then $\lambda_2(G) = O(\Delta \cdot \frac{poly(h)}{n})$. The combinatorial

[3]Given two graphs H and G, we say H is a *minor* of G if H can be obtained from G by a sequence of edge contractions and vertex deletions.

techniques of [52] are indeed powerful. They relate the spectral properties of graph Laplacians to multi-commodity flows. Further extending these techniques, Kelner *et al.* [195] proved the following theorem characterizing the spectra of planar graphs.

Theorem 5.50 (Spectra Characterization of Planar Graphs)**.** For any n-vertex planar graph G with constant maximum degree, for any integer $k > 0$: $\lambda_k(G) \leq O\left(\frac{k}{n}\right)$.

Kelner *et al.* also extended this result to constant-degree graphs with a bounded genus and bounded forbidden minor. This graph spectral theorem shows that, in contrast to expander graphs whose Fiedler value jumps to $\Theta(1)$, these planar-like graphs are fundamentally "local" — their spectra grow linearly.

6

Spectral Similarity: Sparsification - Making Networks Simpler

In data mining and Web search, it is challenging to measure the similarity between two objects. This is particularly true as the subjects of the study have evolved to such a great extent from documents to images, from images to videos, and from videos to complex social networks. On the one hand, for algorithmic efficiency such as in locality-sensitive hashing [74, 150, 174, 81, 25, 107], it is desirable to use simple similarity functions. On the other hand, the domain semantics are often too complex or are not well defined. This makes the validation of similarity functions application specific or even experiment specific.

When defining the similarity between two objects, a popular approach is to first map each object of the domain to a point in a metric space, and then use the distance between the "geometric" embeddings of two objects to define their similarity.

If the "semantic" similarities between the objects are sufficiently captured, this geometric-embedding framework has many benefits. One can use the vast body of geometric algorithms and data structures [150, 174, 25, 180] to support similarity-based search and classification. The geometric embeddings can also provide a wonderful way to visualize relations among complex objects.

In this Chapter, we focus on the following fundamental question in network analysis:

How should we measure the similarity between two networks?

We consider this question when two networks $G = (V, E, \mathbf{W})$ and $\widetilde{G} = (V, \widetilde{E}, \widetilde{\mathbf{W}})$ have the same and labeled node set.[1] Without any domain knowledge, the most intuitive notion of similarity is the following: For $\sigma \geq 1$, G and $\widetilde{G}$ are *σ-linkwise-similar*, if for all $u, v \in V$:

$$\frac{1}{\sigma} \cdot \widetilde{w}_{u,v} \leq w_{u,v} \leq \sigma \cdot \widetilde{w}_{u,v}.$$

The linkwise similarity, however, could be too restrictive a notion of similarity: A finite σ necessarily implies $E = \widetilde{E}$.

When $\widetilde{E} \neq E$, how should we measure the similarity between $G = (V, E, \mathbf{W})$ and $\widetilde{G} = (V, \widetilde{E}, \widetilde{\mathbf{W}})$?

6.1 Spectral Similarity of Graphs

A major conceptual development in the work of [319, 317] is a new notion of similarity, known as *spectral similarity*. It is based on the spectra of *Laplacian matrices*. Rooted in spectral graph theory, this similarity notion leads to a striking result for graph sparsification.

Definition 6.1 (Spectral Similarity of Networks)**.** Suppose $G = (V, E, \mathbf{W})$ and $\widetilde{G} = (V, \widetilde{E}, \widetilde{\mathbf{W}})$ are two weighted undirected networks over the same set V of n nodes. Let $\mathbf{L_W} = \mathbf{D_W} - \mathbf{W}$ and $\mathbf{L}_{\widetilde{\mathbf{W}}} = \mathbf{D}_{\widetilde{\mathbf{W}}} - \widetilde{\mathbf{W}}$ be the *Laplacian matrices*, respectively, of these two networks. Then, for $\sigma \geq 1$, we say G and $\widetilde{G}$ are *σ-spectrally similar* if:

$$\forall \mathbf{x} \in \mathbb{R}^n, \quad \frac{1}{\sigma} \cdot \mathbf{x}^T \mathbf{L}_{\widetilde{\mathbf{W}}} \mathbf{x} \leq \mathbf{x}^T \mathbf{L_W} \mathbf{x} \leq \sigma \cdot \mathbf{x}^T \mathbf{L}_{\widetilde{\mathbf{W}}} \mathbf{x} \tag{6.1}$$

---

[1] If nodes are not labeled, then one has to solve the graph isomorphism problem [36] to address this question. In the most general form of this question, networks may have different and unlabeled node sets.

Let $\sigma(G, \widetilde{G}) = \inf\{\sigma : G \text{ and } \widetilde{G} \text{ are } \sigma\textit{-spectrally similar}\}$. Conventionally, one normalizes similarity measures to $[0, 1]$, with 1 representing "exactly same", and 0 representing "dissimilar". In this spirit, we define: $\text{SpectralSimilarity}(G, \widetilde{G}) := 1/\sigma(G, \widetilde{G})$.

Proposition 6.2 (Normalized)**.** $\text{SpectralSimilarity}(G, \widetilde{G}) \in [0, 1]$, $\forall$ $G, \widetilde{G}$. Moreover, $\text{SpectralSimilarity}(G, \widetilde{G}) = 1$ iff $G = \widetilde{G}$.

Recall that for all $\mathbf{x} \in \mathbb{R}^n$:

$$\mathbf{x}^T \mathbf{L}_\mathbf{W} \mathbf{x} = \sum_{(i,j) \in E} w_{i,j} \, (x_i - x_j)^2 \tag{6.2}$$

Therefore:

Proposition 6.3 (Linkwise Similarity)**.** If G and $\widetilde{G}$ are σ-linkwise similar, then G and $\widetilde{G}$ are σ-spectrally similar.

As an example for illustrating spectral similarity, and how it differs from linkwise similarity, let's consider the following two popular graphs. For simplicity, we assume n is a power of two.

- Cliques: Let K_n be the complete graph on n vertices, where each edge has weight 1.
- Hypercubes: Let $H_{\log n}$ be the $(\log n)$-dimensional hypercube with edge weights $n/(2\sqrt{\log n})$.

Although cliques are the ideal graphs for parallel computer architecture, hypercubes are much widely used [225].

Proposition 6.4. K_n and $H_{\log n}$ are $\sqrt{\log n}$-spectrally similar.

Proof. Let $\mathbf{L}_K$ and $\mathbf{L}_H$ denote the Laplacian matrices of these two networks. It is well known that all non-zero eigenvalues of $\mathbf{L}_K$ are equal

n. The non-zero eigenvalues of the Laplacian of the unit-weight $(\log n)$-hypercube are $(2, 4, ..., 2\log n)$, and the multiplicity of eigenvalue $2k$ is $\binom{\log n}{k}$. Each $\mathbf{x} \in \mathbb{R}^n$ can be expressed as $\mathbf{x} = \alpha \cdot \mathbf{1} + \mathbf{z}$ for a scalar α and $\mathbf{z} \perp \mathbf{1}$. Then, $\mathbf{x}^T\mathbf{L}_K\mathbf{x} = \mathbf{z}^T\mathbf{L}_K\mathbf{z} = n \cdot (\mathbf{z}^T\mathbf{z})$ and $\mathbf{x}^T\mathbf{L}_H\mathbf{x} = \mathbf{z}^T\mathbf{L}_H\mathbf{z}$. The spectra of unit-weight hypercubes implies that the latter is in the interval:

$$\mathbf{z}^T\mathbf{L}_H\mathbf{z} \in \frac{n}{2\sqrt{\log n}} \cdot [2, 2\log n] \cdot (\mathbf{z}^T\mathbf{z}) = n \cdot \left[\frac{1}{\sqrt{\log n}}, \sqrt{\log n}\right] \cdot (\mathbf{z}^T\mathbf{z}).$$

Therefore, K_n and $H_{\log n}$ are $\sqrt{\log n}$-spectrally similar. □

Thus, SpectralSimilarity$(K_n, H_{\log n}) = \frac{1}{\sqrt{\log n}}$. Remarkably, hypercubes are much sparser than complete graphs, making it more desirable for real-world parallel computer architecture.

6.2 Some Basic Properties of Spectrally Similar Networks

For any $S \subseteq V$, let $\mathbf{1}_S : V \to \{0, 1\}$ be the indicator function of S:

$$\mathbf{1}_S(v) = \begin{cases} 1 & \text{if } v \in S \\ 0 & \text{if } v \notin S \end{cases}$$

Then, $\mathbf{1}_S^T\mathbf{L}_\mathbf{W}\mathbf{1}_S = \text{cut}_\mathbf{W}(S, \bar{S})$. Thus, the spectral similarity strengthens the *cut-similarity* of Benczur and Karger [49].

Proposition 6.5 (Spectral Similarity Implies Cut Similarity)**.** If $G = (V, E, \mathbf{W})$ and $\widetilde{G} = (V, \widetilde{E}, \widetilde{\mathbf{W}})$ are σ-spectrally similar, then:

$$\forall S \subset V, \quad \frac{1}{\sigma} \cdot \text{cut}_{\widetilde{\mathbf{W}}}(S, \bar{S}) \leq \text{cut}_\mathbf{W}(S, \bar{S}) \leq \sigma \cdot \text{cut}_{\widetilde{\mathbf{W}}}(S, \bar{S}).$$

Thus, for any pair $s, t \in V$, the value of the s-t maximum flow in G is within a multiplicative factor of σ of that in $\widetilde{G}$ and vice versa.

As noted in [317, 43], the converse of Proposition 6.5 is not true. For example, the path graph P_n and the cycle graph C_n with n vertices are 2-cut-similar, but spectrally, they are very different — they are only $\Omega(n)$-spectrally similar. Algebraically, the difference arises because the

cut-similarity only needs to satisfy the inequalities (6.1) for integer vectors in $\{-1, 1\}^n$. Thus, spectral similarity is a strictly stronger notion of similarity than cut or flow similarity. For various applications that we shall discuss later, spectral similarity plays a significant role, as it approximately preserves the spectra of the graphs and their associated eigenvectors:

Proposition 6.6 (Spectra of the Laplacian)**.** Suppose G and $\widetilde{G}$ are σ-spectrally similar. If $\lambda_1 \leq ... \leq \lambda_n$ and $\widetilde{\lambda}_1 \leq ... \leq \widetilde{\lambda}_n$ are the eigenvalues, respectively, of Laplacians $\mathbf{L_W}$ and $\mathbf{L}_{\widetilde{\mathbf{W}}}$, then $\forall i \in [n]$:

$$\frac{1}{\sigma} \cdot \widetilde{\lambda}_i \leq \lambda_i \leq \sigma \cdot \widetilde{\lambda}_i.$$

Proof. Follows directly from the Courant-Fisher Theorem. □

Spectral similarity can be more generally defined for symmetric positive semi-definite matrices. We say $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times n}$ are *σ-spectrally similar* for $\sigma \geq 1$ if for all $\mathbf{x} \in \mathbb{R}^n$:

$$\frac{1}{\sigma} \cdot \mathbf{x}^T\mathbf{B}\mathbf{x} \leq \mathbf{x}^T\mathbf{A}\mathbf{x} \leq \sigma \cdot \mathbf{x}^T\mathbf{B}\mathbf{x}.$$

Proposition 6.7 (Inverse, Null-Space, and Pseudoinverse)**.** Suppose two symmetric positive semi-definite matrices $\mathbf{A}$ and $\mathbf{B}$ are σ-spectrally similarity for a finite σ, then either:

- $\mathbf{A}$ and $\mathbf{B}$ are both non-singular, in which case, their inverse $\mathbf{A}^{-1}$ and $\mathbf{B}^{-1}$ are also σ-spectral similar, or
- $\mathbf{A}$ and $\mathbf{B}$ have the same null space, and their Moore-Penrose pseudo-inverse $\mathbf{A}^\dagger$ and $\mathbf{B}^\dagger$ are also σ-spectrally similar.

This approximation property draws a connection with another notion of graph similarity — *distance similarity.*

Definition 6.8 (Distance Similarity)**.** Two metric spaces $\mathcal{M} = (X, \mathrm{dist})$ and $\widetilde{\mathcal{M}} = \left(X, \widetilde{\mathrm{dist}}\right)$ are σ-*distance similar* for $\sigma \geq 1$ if for all $u, v \in X$:

$$\frac{1}{\sigma} \cdot \widetilde{\mathrm{dist}}(u, v) \leq \mathrm{dist}(u, v) \leq \sigma \cdot \widetilde{\mathrm{dist}}(u, v).$$

Distance similarity is the subject of study in graph spanners [274, 273, 89] and graph embedding, exemplified by the Johnson-Lindenstrauss theory [180]. In the context of Laplacian matrices, recall that each network $G = (V, E, \mathbf{W})$ defines a natural metric space, $\mathcal{M}_{\mathbf{W}}^{\mathrm{ER}} = (V, \mathrm{ER}_{\mathbf{W}})$, where $\mathrm{ER}_{\mathbf{W}}(u, v)$ denotes the effective resistance between $u, v \in V$.

Algebraically, the effective resistance between two vertices can be expressed as:

$$\mathrm{ER}_{\mathbf{W}}(u, v) = (\mathbf{1}_u - \mathbf{1}_v)^T \, \mathbf{L}_{\mathbf{W}}{}^{\dagger} \, (\mathbf{1}_u - \mathbf{1}_v) \tag{6.3}$$

It is well-known that $\mathrm{ER}_{\mathbf{W}}(u, v)$ is proportional to the random-walk commute time between u and v in G [119, 55]. By Proposition 6.7:

Proposition 6.9 (Effective Distance and Random Walks)**.** Suppose $G = (V, E, \mathbf{W})$ and $\widetilde{G} = (V, \widetilde{E}, \widetilde{\mathbf{W}})$ are σ-spectrally similar. Then, for all $u, v \in V$:

$$\frac{1}{\sigma} \cdot \mathrm{ER}_{\widetilde{\mathbf{W}}}(u, v) \leq \mathrm{ER}_{\mathbf{W}}(u, v) \leq \sigma \cdot \mathrm{ER}_{\widetilde{\mathbf{W}}}(u, v).$$

In other words, their induced metric spaces $\mathcal{M}_{\mathbf{W}}^{ER} = (V, \mathrm{ER}_{\mathbf{W}})$ and $\mathcal{M}_{\widetilde{\mathbf{W}}}^{ER} = (V, \mathrm{ER}_{\widetilde{\mathbf{W}}})$ are σ-distance similar.

6.3 Spectral Graph Sparsification

Graph sparsification is the task of approximating a "dense" graph by a "sparse" graph that can be effectively used in place of the dense one. In various field of computing, engineering trade-offs usually require making things sparse in order to convert theoretical designs into practical

implementation. In communication systems, sparser *crossbar switches* effectively emulate dense point-wise connections between inputs and outputs. In parallel computers, sparser hypercubes or "fat trees" [225] are used as the architecture to support communication patterns as complex as complete graphs. In scientific computing, sparser meshes are used to approximate the interactions among particles or "space-time" variables of the governing mathematical equation. In the real world, we drive on relatively sparse highways and streets to go between cities and neighborhoods. Not surprisingly, graph sparsification is very useful in algorithm design and combinatorial optimization.

But in each application, one need to use sparsifiers are *effective*. Like the definition of good samplers and clusters, the definition of good sparsifiers varies from application to application.

The notion of a good sparsifier often involves several parameters. For example, in urban planning and network design, while the distance-similarity is of primary importance, congestion is also an important factor. In mesh generation for finite-element methods and computer graphics, while interpolation error is the key concern, the condition number of the resulting matrix is also crucial to numerical stability. Thus, the mathematical study of graph sparsification addresses the following three questions that are central to its potential applications:

- **Conceptual Question**: *How should one measure the similarity between two graphs to ensure that one can effectively use the sparsifiers in place of the ideal dense graphs?*
- **Mathematical Question**: *For our choice of graph similarity, does every graph have a "similar" sparse graph?*
- **Algorithmic Question**: *Is there a scalable algorithm for constructing good sparsifiers?*

Recall that spectral similarity simultaneously preserves cut-similarity, which measures network congestion, and distance-similarity of the effective-resistance metric, which models random walks. Thus, it is not a surprise that hypercubes — the "universal networks" for

parallel computing [225] — can be properly weighted into sufficiently good spectral sparsifiers for the complete graphs.

In fact, the Ramanujan expander graphs [241, 244] provide a family of better spectral sparsifiers for complete graphs than hypercubes. A Ramanujan graph of degree d is a d-regular graph all of whose non-zero Laplacian eigenvalues lie between $d - 2\sqrt{d-1}$ and $d + 2\sqrt{d+1}$. Thus, all non-zero eigenvalues of a Ramanujan graph with edge weight n/d are in the interval:

$$\left[n - \frac{2n\sqrt{d-1}}{d}, n + \frac{2n\sqrt{d-1}}{d}\right].$$

This implies that the clique K_n has a sparsifier of $O(n/\epsilon^2)$ edges that is $(1+\epsilon)$-spectrally similar to K_n, by setting $\epsilon = 2\sqrt{d-1}/d$.

Essentially extending this fact about cliques, Spielman and Teng [317] proved the following spectral sparsification theorem:

Theorem 6.10 (Spectral Sparsification)**.** For any $0 < \epsilon < 1$, every weighted undirected graph $G = (V, E, \mathbf{W})$ has a sparsifier $\widetilde{G} = (V, \widetilde{E}, \widetilde{\mathbf{W}})$ with $\widetilde{E} \subset E$ and $\widetilde{O}(n/\epsilon^2)$ edges that is $(1+\epsilon)$-spectrally similar to G. Moreover, there is a scalable randomized algorithm for constructing such a spectral sparsifier.

This theorem generalizes the following remarkable result of Benczur and Karger [49] regarding cut-similarity motivated by improving optimization algorithms for the min-cut/max-flow problem.

Theorem 6.11 (Benczur-Karger)**.** For any constant $0 < \epsilon < 1$, for all weighted graphs $G = (V, E, \mathbf{W})$, one can scalably compute a graph $\widetilde{G} = (V, \widetilde{E}, \widetilde{\mathbf{W}})$ such that (1) G and $\widetilde{G}$ are $(1+\epsilon)$-cut similar, (2) $\widetilde{E} \subset E$, and (3) $|\widetilde{E}| = O(n \log n/\epsilon^2)$.

While scalable, Spielman-Teng's original sparsification algorithm is only of theoretical interest, as the factor in its $\widetilde{O}$ contains a prohibitively high-order logarithmic term. Theorem 6.10 was subsequently improved

by Spielman and Srivastava [311] who proved that spectral sparsifiers with $|\widetilde{E}| = O(n \log n/\epsilon^2)$ can be constructed. Batson, Spielman, and Srivastava [44] then give a beautiful construction that shows every graph has a spectral sparsifier essentially as sparse as the Ramanujan expander graphs.

Theorem 6.12 (Batson-Spielman-Srivastava)**.** For any approximation parameter $\epsilon \in (1/\sqrt{n}, 1/3)$, for all weighted graphs $G = (V, E, \mathbf{W})$, there exists a graph $\widetilde{G} = (V, \widetilde{E}, \widetilde{\mathbf{W}})$ such that (1) G and $\widetilde{G}$ are $(1+\epsilon)$-spectrally similar, (2) $\widetilde{E} \subset E$, and (3) $|\widetilde{E}| = O(n/\epsilon^2)$.

The sparsifier of Theorem 6.12 can be constructed in polynomial time [44]. The recent progress in Laplacian solver — see next section — provides an efficient scalable implementation of Spielman-Srivastava's sparsification algorithm [311]. It remains open if there is a scalable algorithm for constructing an $(1+\epsilon)$-spectral sparsifier of size $O(n/\epsilon^2)$. The fastest algorithm runs in almost linear time [221], with complexity $\widetilde{O}(m) + n^{1+o(1)}$.

6.4 Graph Inequalities and Low-Stretch Spanning Trees

In this section, we will review the basic graph structures and inequalities useful for understanding spectral similarity. We will then take this opportunity to introduce a remarkable graph-theoretical concept, the *low-stretch spanning tree*, of Alon, Karp, Peleg, and West [11]. In the next section, we will discuss techniques and algorithms for spectral sparsification, focusing on the Spielman-Srivastava proof [311] of Theorem 6.10.

Edge Decomposition of The Laplacian Matrix

The quadratic formula of Eqn. (6.2) immediately suggests an edge decomposition of the Laplacian matrix: For each edge $e = (u, v) \in E$, let $\mathbf{L}_e$ be the Laplacian matrix of the unweighted, single-edge graph $(V, \{(u, v)\})$, i.e., $\mathbf{L}_e$ is the symmetric $n \times n$ matrix with four non-zero

entries at $\mathbf{L}_e[u,u] = \mathbf{L}_e[v,v] = 1$ and $\mathbf{L}_e[u,v] = \mathbf{L}_e[v,u] = -1$. Then:

Lemma 6.13 (Summation of Edge Laplacians)**.**

$$\mathbf{L_W} = \sum_{e \in E} w_e \cdot \mathbf{L}_e \tag{6.4}$$

Proof. Consider the *edge-vertex incidence matrix* $\mathbf{B}$ of the unweighted part (V, E) of G. $\mathbf{B}$ is an $|E| \times |V|$ matrix with rows indexed by edges and columns indexed by nodes. For each $e = (u, v) \in E$, the e^{th} row of $\mathbf{B}$ is either $\mathbf{1}_u^T - \mathbf{1}_v^T$ or $\mathbf{1}_v^T - \mathbf{1}_u^T$. Note that:

$$\mathbf{L}_e = (\mathbf{1}_u - \mathbf{1}_v) \cdot (\mathbf{1}_u - \mathbf{1}_v)^T = \mathbf{B}[e,:]^T \cdot \mathbf{B}[e,:].$$

Let $\mathbf{C_W}$ denote the $|E| \times |E|$ diagonal matrix indexed by edges in E where the e^{th} diagonal is w_e. Then algebraically:

$$\mathbf{L_W} = \mathbf{B}^T \cdot \mathbf{C_W} \cdot \mathbf{B} = \sum_{e \in E} w_e \cdot \mathbf{B}^T[e,:] \cdot \mathbf{B}[e,:] = \sum_{e \in E} w_e \cdot \mathbf{L}_e \tag{6.5}$$

as stated in the lemma. □

Note that Eqn. (6.2) can be derived directly from Eqn. (6.4) and the basic fact that $\mathbf{x}^T \mathbf{L}_{(u,v)} \mathbf{x} = (x_u - x_v)^2$, for all $\mathbf{x} \in \mathbb{R}^n$.

A Support Theory for Spectral Similarity

The main task of spectral sparsification is to select a small set $\widetilde{E} \subset E$ of edges to support the edges in $E - \widetilde{E}$. As one may expect, this selection will be guided by a notion of *edge centrality* or *leverage score* — some edges are more important to the network than others. Thus, a good sparsifier must retain them or find a way to support them. We now formalize the notion of "to support an edge." For symmetric matrices $\mathbf{A}$ and $\mathbf{B}$, we use $\mathbf{A} \preccurlyeq \mathbf{B}$ to denote that $(\mathbf{B} - \mathbf{A})$ is positive semidefinite.

Proposition 6.14 (Spectral Similarity)**.** For two symmetric positive semi-definite matrices $\mathbf{A}$ and $\mathbf{B}$, and a parameter $\sigma \geq 1$, $\mathbf{A}$ and $\mathbf{B}$ are σ-spectrally similar if and only if $\mathbf{A} \preccurlyeq \sigma \cdot \mathbf{B}$ and $\mathbf{B} \preccurlyeq \sigma \cdot \mathbf{A}$.

Proof. If $\mathbf{A}$ and $\mathbf{B}$ are σ-spectrally similar, then $\forall \mathbf{x} \in \mathbb{R}^n$, $\frac{1}{\sigma}\mathbf{x}^T\mathbf{B}\mathbf{x} \leq \mathbf{x}^T\mathbf{A}\mathbf{x} \leq \sigma \cdot \mathbf{x}^T\mathbf{B}\mathbf{x}$. The left inequality is $\mathbf{x}^T(\sigma \cdot \mathbf{A} - \mathbf{B})\mathbf{x} \geq 0$, which is $\mathbf{B} \preccurlyeq \sigma \cdot \mathbf{A}$. The right inequality is equivalent to $\mathbf{A} \preccurlyeq \sigma \cdot \mathbf{B}$. □

Definition 6.15 (Support)**.** For $\sigma \geq 1$, we say that $G = (V, E, \mathbf{W})$ *σ-supports* $\widetilde{G} = (V, \widetilde{E}, \widetilde{\mathbf{W}})$ if:

$$\mathbf{L}_{\widetilde{\mathbf{W}}} \preccurlyeq \sigma \cdot \mathbf{L}_{\mathbf{W}} \tag{6.6}$$

We way G perfectly supports $\widetilde{G}$ if $\mathbf{L}_{\widetilde{\mathbf{W}}} \preccurlyeq \mathbf{L}_{\mathbf{W}}$.

Sparsification analysis has two directions. For the easier direction, suppose $\widetilde{G} = (V, \widetilde{E}, \widetilde{\mathbf{W}})$ is a sparsifier of $G = (V, E, \mathbf{W})$ with (1) $\widetilde{E} \subset E$, and (2) $\widetilde{w}_e = w_e$, $\forall e \in \widetilde{E}$, and $\widetilde{w}_e = 0$, $\forall e \notin \widetilde{E}$. Then by Eqn. (6.4), $\mathbf{L}_{\widetilde{\mathbf{W}}} \preccurlyeq \mathbf{L}_{\mathbf{W}}$, and thus G perfectly supports $\widetilde{G}$.

Path Inequality

For the other direction, we need to establish an upper bound c such that $\mathbf{L}_{\mathbf{W}} \preccurlyeq c \cdot \mathbf{L}_{\widetilde{\mathbf{W}}}$. We now discuss a graph inequality that is basic to the theory of spectral sparsification. This inequality is useful for providing both the intuition on sparsification design and the tool for sparsification analysis. We will use the following definition.

Definition 6.16 (Resistance of Edges and Paths)**.** In a weighted graph $G = (V, E, \mathbf{W})$, let *resistance* of $e \in E$ be $\text{resistance}(e) = 1/w_e$. For a simple path P of G, let the *resistance* of P be:

$$\text{resistance}(P) = \sum_{e \in P} \text{resistance}(e) = \sum_{e \in P} \frac{1}{w_e}.$$

The following proposition states to what degree a path of the sparsifier supports an edge in the original network.

Proposition 6.17 (Path Inequality)**.** Let $G = (V, E, \mathbf{W})$ and P be a simple path in G connecting u and v. Then:

$$\mathbf{L}_{(u,v)} \preccurlyeq \text{resistance}(P) \cdot \left(\sum_{e \in P} w_e \cdot \mathbf{L}_e \right) \tag{6.7}$$

Proof. We first prove for the case when P contains two edges. Suppose

$$P = ((uz)(zv))$$

with weights $a = w_{u,z}$ and $b = w_{z,v}$, respectively. Then, $\text{resistance}(P) = 1/a + 1/b$, and for any vector $\mathbf{x} \in \mathbb{R}^n$:

$$\begin{aligned} & \mathbf{x}^T \left(\text{resistance}(P) \cdot \left(a\mathbf{L}_{(u,z)} + b\mathbf{L}_{(z,v)} \right) - \mathbf{L}_{(u,v)} \right) \mathbf{x} \\ = \; & \left(\frac{1}{a} + \frac{1}{b} \right) \left(a(x_u - x_z)^2 + b(x_z - x_v)^2 \right) - (x_u - x_v)^2 \\ = \; & \frac{\left(a(x_u - x_z) - b(x_z - x_v) \right)^2}{ab} \geq 0. \end{aligned}$$

The proposition then follows from a simple proof-by-induction on the number of edges in P. □

In sparsification analysis, we uses multiple paths in the sparsifier $\widetilde{G}$ to support an edge of G. Thus, the path inequality provides the foundation for this analysis framework. By Lemma 6.13, the Laplacian of G is the weighted sum of the Laplacian of its edges. Together, these paths then provide an upper bound on how well $\mathbf{L}_{\widetilde{\mathbf{W}}}$ supports $\mathbf{L}_{\mathbf{W}}$.

Low Stretch Spanning Trees

Using Path Inequality (Proposition 6.17), we first analyze how good spectral sparsifiers that spanning trees can be. We recall a classical result. In their studies of *k-server problem*, Alon *et al.* [11] introduced a beautiful way to measure a spanning tree in a "distance" graph:

Definition 6.18 (The Stretch of a Spanning Tree)**.** Suppose T is a spanning tree in a distance network $G = (V, E, \text{length})$. For each $e \in E$, let $P_T(e)$ denote the unique path in T connecting e's two endpoints. Then, the stretch of $e \in E$ with respect to T is:

$$\text{st}_T(e) = \frac{\sum_{e' \in P_T(e)} \text{length}(e')}{\text{length}(e)}.$$

The *average stretch* of G with respect to T is:

$$\text{st}_T(G) = \frac{1}{|E|} \cdot \sum_{e \in E} \text{st}_T(e).$$

For example, suppose $G = (V, E, \text{length})$ represents a network of roads, where each edge $e \in E$ is a road of $\text{length}(e)$ between its two endpoints. Suppose a "budget cut" reduces this "road network" to a backbone tree network T. $P_T(e)$ is the detour from road e in the sparser road network T. Then, $\text{st}_T(G)$ measures the average delay ratio due to the budget cut.

By setting the length of an edge to be inversely proportional to its affinity weight, we immediately obtain the following proposition:

Proposition 6.19 (Stretch and Resistance)**.** For each $e \in E$ in $G = (V, E, \mathbf{W})$, let $\text{length}_{\mathbf{W}}(e) := \frac{1}{w_e}$. Then, for each spanning tree T:

$$\text{st}_T(G) = \frac{1}{|E|} \cdot \sum_{e \in E} w_e \cdot \text{resistance}(P_T(e)).$$

Proof.

$$\text{st}_T(e) = \frac{\sum_{e' \in P_T(e)} \text{length}(e')}{\text{length}(e)} = w_e \cdot \sum_{e' \in P_T} \frac{1}{w_{e'}}.$$

□

As first observed by Boman and Hendrickson, Path Inequality implies:

Lemma 6.20 (Spectral Similarity of Spanning Trees)**.** Suppose T is a spanning tree in network $G = (V, E, \mathbf{W})$. Let $\mathbf{L}_T$ be the Laplacian matrix of T. Then:

$$\mathbf{L}_T \preccurlyeq \mathbf{L}_\mathbf{W} \preccurlyeq |E| \cdot \mathrm{st}_T(G) \cdot \mathbf{L}_T.$$

Proof.

$$\begin{aligned}
\mathbf{L}_\mathbf{W} &= \sum_{e \in E} w_e \cdot \mathbf{L}_e \preccurlyeq \sum_{e \in E} w_e \cdot \mathrm{resistance}(P_T(e)) \left(\sum_{e' \in P_T(e)} w_{e'} \cdot \mathbf{L}_{e'} \right) \\
&\preccurlyeq \sum_{e \in E} \mathrm{st}_T(e) \cdot \left(\sum_{e' \in P_T(e)} w_{e'} \cdot \mathbf{L}_{e'} \right) = \sum_{e' \in T} w_{e'} \cdot \mathbf{L}_{e'} \cdot \sum_{e : P_T(e) \ni e'} \mathrm{st}_T(e) \\
&\preccurlyeq |E| \cdot \mathrm{st}_T(G) \cdot \sum_{e' \in T} w_{e'} \cdot \mathbf{L}_{e'} = |E| \cdot \mathrm{st}_T(G) \cdot \mathbf{L}_T.
\end{aligned}$$

□

This algebraic connection in fact provides a sharp bound on the quality of spanning trees as spectral sparsifiers, thanks to a series of fundamental advances in low-stretch spanning trees [11, 125, 5]:

Theorem 6.21 (Scalable Low Stretch Spanning Trees)**.** Each distance network $G = (V, E, \mathrm{length})$ has a spanning tree T with:

$$\mathrm{st}_T(G) = O(\log n \log \log^2 n)$$

that can be scalably constructed.

We bound the quality of $\widetilde{G} = (V, \widetilde{E}, \widetilde{\mathbf{W}})$ as a spectral sparsifier for $G = (V, E, \mathbf{W})$ by fractionally "routing" each edge $e \in E - \widetilde{E}$ over a set of paths in $\widetilde{G}$: For each $e \in E$, a routing scheme Π routes $e = (u, v)$ over a set of paths $\Pi(e)$ connecting u and v in $\widetilde{G}$, where $P \in \Pi(e)$ carries τ_P fractional weights of e, such that $\sum_{P \in \Pi(e)} \tau_P = 1$.

By Eqn. (6.7) and (6.4), the routing Π provides an estimate on the quality of $\widetilde{G}$ for supporting edges in G:

$$\begin{aligned}\mathbf{L}_G &= \sum_{e\in E} w_e \cdot \mathbf{L}_e \preccurlyeq \sum_{e\in E} w_e \cdot \sum_{P\in\Pi(e)} \tau_P \cdot \text{resistance}(P) \cdot \left(\sum_{\widetilde{e}\in P} w_{\widetilde{e}} \cdot \mathbf{L}_{\widetilde{e}}\right) \\ &= \sum_{\widetilde{e}\in\widetilde{E}} \left(\sum_{(e\in E)\&(P\in\Pi(e))\&(\widetilde{e}\in P)} \tau_P \cdot w_e \cdot \text{resistance}(P)\right) \cdot w_{\widetilde{e}} \cdot \mathbf{L}_{\widetilde{e}}\end{aligned}$$

The quantity:

$$\text{RC}_\Pi(\widetilde{e}) = \left(\sum_{(e\in E)\&(P\in\Pi(e))\&(\widetilde{e}\in P)} \tau_P \cdot w_e \cdot \text{resistance}(P)\right)$$

represents Π's *resistance congestion* over edge $\widetilde{e}$, when routing edges of E over paths in $\widetilde{G}$. Then:

$$\mathbf{L}_{\widetilde{G}} \preccurlyeq \mathbf{L}_G \preccurlyeq \left(\min_{\Pi} \max_{\widetilde{e}\in\widetilde{E}} \text{RC}_\Pi(\widetilde{e})\right) \cdot \mathbf{L}_{\widetilde{G}} \tag{6.8}$$

6.5 Edge Centrality, Sampling, and Spectral Approximation

Schematically, sparsification can be viewed either as a process of removing edges or a process of selecting edges. Our Path Inequality offers a greedy scheme that repeatedly removes edges from networks, which can be efficiently supported by other edges. However the first two scalable spectral sparsification algorithms [315, 311] use sampling-based methods. They randomly select network edges from carefully designed distributions.

At a high level, instead of deterministically selecting edges, a sampling algorithm first selects a probability $p_e \in [0, 1]$ for each $e \in E$. Then, it selects e with probability p_e and includes it in the sparsifier $\widetilde{G}$ by setting its weight to w_e/p_e. This sampling process guarantees that:

$$\text{E}\left[\widetilde{\mathbf{W}}\right] = \mathbf{W} \tag{6.9}$$

Alternatively, given probability vectors $(p_e : e \in E)$, and a target $\widetilde{m}$, we say $\widetilde{G} = (V, E, \widetilde{\mathbf{W}})$ is an *$\widetilde{m}$-edge sample*, according to $(p_e : e \in E)$, if $\widetilde{G}$ is obtained by the following random sampling process:

Algorithm: GraphSampling$(G, \widetilde{m}, (p_e : e \in E))$

1: Set $\widetilde{E} = \emptyset$ and $\widetilde{\mathbf{W}} = \mathbf{0}$.
2: **for** $t = 1 : \widetilde{m}$ **do**
3: Choose a random edge of G with probability p_e.
4: Add e to $\widetilde{E}$ and increase $\widetilde{w}_e$ by w_e/p_e.
5: Return $\widetilde{G} = (V, \widetilde{E}, \widetilde{\mathbf{W}})$.

For GraphSampling, the basic mathematical question becomes:

> *What are the* right *edge probabilities* $\{p_e : e \in E\}$ *for spectral sparsification? How many edges do we need sample?*

Intuitively, answers to this question introduce some notion of edge centrality (or leverage score): Edges that are more essential to the network — those that are difficult for other edges to support by Path Inequality — should be given larger probabilities. Using the local clustering algorithms discussed in Chapter 4, Spielman and Teng [318] gave a scalable algorithm for constructing a "good enough" edge distribution in their proof of Theorem 6.10. Spielman and Srivastava [311] then provided a remarkable answer to this question based on effective resistances, which led to the current best scalable algorithm for spectral graph sparsification.

Theorem 6.22 (Effective-Resistance Based Sampling)**.** For any $G = (V, E, \mathbf{W})$ and $\epsilon, \delta \in (0, 1]$, let:

$$\widetilde{G} = \mathsf{GraphSampling}(G, \widetilde{m}, (p_e, e \in E))$$

where $\widetilde{m} = 8n \log n \log \delta^{-1}/\epsilon^2$ and $(p_e : e \in E)$ is the distribution in which p_e is proportional to $w_e \cdot \mathrm{ER}_{\mathbf{W}}(e)$. Then, with probability at least $1 - \delta$, $\widetilde{G}$ is a $(1 + \epsilon)$-spectral approximation of G.

Proof. (Sketch) The theorem can be proved using the following concentration theorem by Rudelson [292]:

Theorem 6.23 (Rudelson). Suppose $\mathbf{p}$ is a probability distribution over $\Omega \subset \mathbb{R}^n$ such that $\sup_{\mathbf{x}\in\Omega} ||\mathbf{x}||_2 \leq M$ and $\mathrm{E}_{\mathbf{x}\sim\mathbf{p}(\Omega)}\left[||\mathbf{x}\mathbf{x}^T||_2\right] \leq 1$. Let $\Pi = \mathrm{E}_{\mathbf{x}\sim\mathbf{p}(\Omega)}\left[\mathbf{x}\mathbf{x}^T\right]$, and $\mathbf{y}_1, ..., \mathbf{y}_{\widetilde{m}}$ be independent $\widetilde{m}$ samples drawn from $\mathbf{p}$, we have:

$$\mathrm{E}\left[\left|\left|\frac{1}{\widetilde{m}}\left(\sum_{k=1}^{\widetilde{m}} \mathbf{y}_k\mathbf{y}_k^T\right) - \Pi\right|\right|_2\right] \leq 8M\sqrt{\log\widetilde{m}/\widetilde{m}}.$$

For $e \in E$, let $\{p_e : e \in E\}$ be a probability vector, where p_e is proportional to $w_e \cdot \mathrm{ER}_{\mathbf{W}}(e)$. Recall that G's incidence matrix is an $|E|\times|V|$ matrix, in which $\mathbf{1}_u^T - \mathbf{1}_v^T$ is the row associated with $(u, v) \in E$. Note that this incidence matrix is a square-root factor of $\mathbf{L}_{\mathbf{W}}$. For each edge $(u, v) \in E$, let:

$$\mathbf{y}_{(u,v)} = \frac{1}{\sqrt{p_{u,v}}} \cdot w_{u,v}^{1/2}(\mathbf{L}_{\mathbf{W}}{}^{\dagger})^{1/2}(\mathbf{1}_u - \mathbf{1}_v).$$

In other words, we map $\mathbf{1}_u - \mathbf{1}_v$ to $\mathbf{y}_{(u,v)}$, whose length is at most $\sqrt{n-1}$. Consider the i.i.d distribution over $\Omega = \{\mathbf{y}_e : e \in E\}$, that selects $\mathbf{y}_e$ with probability p_e. The inverse of this map from $\mathbf{1}_u - \mathbf{1}_v$ to $\mathbf{y}_{(u,v)}$ then transforms the process of sampling of Ω by Rudelson's Theorem to the process of sampling of edges of G, according to $(p_e : e \in E)$. Thus, $\widetilde{m} = 8n\log n\log\delta^{-1}/\epsilon^2$ samples probably produce a sparsifier $\widetilde{G}$ that $(1+\epsilon)$-spectrally approximates G. See the recent survey [43] on spectral sparsification for details. □

Scalable Algorithms for Spectral Sparsification

Spielman and Teng [317] designed the first scalable spectral sparsification algorithm. Their algorithm repeatedly uses local clustering to decompose the input network into "well-connected" components. The clustering algorithm guarantees that the smallest non-zero eigenvalue of the normalized Laplacian matrices of these components is at least $\Omega(\frac{1}{\log^2 n})$. The eigenvalue bound ensures that these components contain no cluster with small conductance. Theorem 6.22 then guarantees that

we can obtain good sparsifiers of these components by sampling, based simply on the uniform edge distributions.

Local clustering also guarantees that at most $|E|/2$ edges are removed to separate these components. Thus, we can recursively sparsify these edges using local clustering and sampling. When local clustering stops producing well-connected components, the union of the remaining edges and the sparsifiers of these well-connected components forms the final sparsifier. This sparsification algorithm is scalable. However, its running time has a polylogarithmic term with a large exponent.

The Spielman-Srivastava sampling process naturally reduces spectral sparsification to effective-resistances approximation. Recall from Section 7.2, for all $(u, v) \in E$:

$$\mathrm{ER}_{\mathbf{W}}(u, v) = (\mathbf{1}_u - \mathbf{1}_v)^T \mathbf{L}_{\mathbf{W}}{}^{\dagger} (\mathbf{1}_u - \mathbf{1}_v) .$$

Thus, the effective resistance between a pair $u, v \in V$ in the network can be computed by solving the following numerical algebraic problem:

Definition 6.24 (Laplacian Linear Systems)**.** Given an $n \times n$ Laplacian matrix $\mathbf{L} = \mathbf{D}_{\mathbf{W}} - \mathbf{W}$ and a vector $\mathbf{b} \in \mathbb{R}^n \perp \mathbf{1}$, find an $\mathbf{x} \in \mathbb{R}^n$ such that:

$$\mathbf{L} \cdot \mathbf{x} = \mathbf{b} \tag{6.10}$$

We will focus on applications of Laplacian linear systems in the next chapter. As a wonderful illustration of scalable Laplacian solvers, Spielman and Srivastava [311] developed a scalable algorithm for building an efficient data structure for answering queries of the form:

What is the effective resistance between two vertices $u, v \in V$ in G?

We will discuss their data structure in Theorem 7.8. This data structure can be used for conducting (approximate) sampling from the probability distribution $(p_e : e \in E)$, where p_e is proportional to $w_e \cdot \mathrm{ER}_{\mathbf{W}}(e)$. Thus, we can use $\widetilde{G} = \mathsf{GraphSampling}(G, \widetilde{m}, (p_e, e \in E))$ to provide a scalable implementation of Theorem 6.22.

6.6 Scalable Dense-Matrix Computation via Sparsification

We conclude this chapter with an application of spectral sparsification:

> *We can solve Laplacian linear systems in $\widetilde{O}(n^2)$ time using scalable spectral sparsification as subroutine.*

Note that this solver is scalable for dense Laplacian matrices, i.e., $\text{nnz}(\mathbf{L}) = \Omega(n^2)$. Scalable solvers for general Laplacian systems will be discussed in the next chapter. It remains an outstanding open question in algorithm design and numerical analysis to determine if one can, in $\widetilde{O}(n^2)$ time, find an (approximate) solution of $n \times n$ linear system:

$$\mathbf{A}\mathbf{x} = \mathbf{b}.$$

The question remains open, even when $\mathbf{A}$ is symmetric and positive-definite. Some matrix operations are known to have scalable implementation. An example is matrix-vector product: $\mathbf{A} \cdot \mathbf{x}$. This operation is scalable in the strong sense that when $\mathbf{A}$ is given in its sparse-matrix form, $\mathbf{A} \cdot \mathbf{x}$ — for any $\mathbf{x} \in \mathbb{R}^n$ — can be computed in time $O(\text{nnz}(\mathbf{A}))$. While Gaussian elimination [155] can be used to solve any linear system in polynomial time, the fastest algorithm for solving general symmetric, positive semi-definite linear systems is the Conjugate Gradient (CG) method:

Algorithm: $\mathsf{CG}(\mathbf{A}, \mathbf{b}, \epsilon)$

1: Set $\mathbf{x} = \mathbf{0}$ and set $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$, and $\mathbf{p} = \mathbf{r}$.
2: **while** $||\mathbf{r}|| \geq \epsilon$ **do**
3: Set $\alpha = ||\mathbf{r}||^2/(\mathbf{p}^T\mathbf{A}\mathbf{p})$, $\mathbf{x} = \mathbf{x} + \alpha\mathbf{p}$ and $\mathbf{r}_{new} = \mathbf{r} - \alpha\mathbf{A}\mathbf{p}$.
4: Set $\beta = (||\mathbf{r}_{new}||/||\mathbf{r}||)^2$, $\mathbf{p} = \mathbf{r}_{new} + \beta\mathbf{p}$ and $\mathbf{r} = \mathbf{r}_{new}$
5: Return $\mathbf{x}$.

We include the pseudocode above to highlight that CG's main computation steps are matrix-vector products, (i.e., $\mathbf{A} \cdot \mathbf{p}$ in Step 4). We will have more discussion in the next chapter regarding CG's numerical properties when it used as an approximate solver. In this section, we will use the following basic fact [334]: *When implemented with infinite*

precision, CG with n iterations can be used as an exact solver. Thus, in time $O(n \cdot \text{nnz}(A))$, CG computes a solution to $\mathbf{Ax} = \mathbf{b}$, for any $n \times n$ symmetric positive semi-definite matrix $\mathbf{A}$ and vector $\mathbf{b}$ (in the span of $\mathbf{A}$). Clearly, due to this additional factor of n, CG as implemented above is not scalable, neither for dense nor for sparse matrices.

We now use spectral sparsification in CG to obtain a scalable solver for dense Laplacian systems. For $\mathbf{z} \in \mathbb{R}^n$, let the $\mathbf{A}$-*norm* of $\mathbf{z}$ be:

$$||\mathbf{z}||_{\mathbf{A}} := \sqrt{\mathbf{z}^T \mathbf{A} \mathbf{z}}.$$

Let's first recall a classical result from numerical analysis.

Theorem 6.25 (Convergence of CG)**.** For any symmetric positive definite $n \times n$ matrix $\mathbf{A}$ and vector $\mathbf{b} \in \mathbb{R}^n$, let $\mathbf{x}^*$ be the exact solution to $\mathbf{Ax} = \mathbf{b}$. Then, for any ϵ, the number of iterations for CG to obtain an approximation solution $\mathbf{x}_{CG}$ such that $||\mathbf{x}^{CG} - \mathbf{x}^*||_{\mathbf{A}} \leq \epsilon ||\mathbf{x}^*||_{\mathbf{A}}$ is:

$$O(\sqrt{\kappa(\mathbf{A})} \log(1/\epsilon))$$

where $\kappa(\mathbf{A})$ denotes the condition number of $\mathbf{A}$.

This convergence bound is the result of the following general error bound for CG: For $t \geq 1$, let $\mathbf{x}^{CG(t)}$ be the approximate solution obtained after the t^{th} iteration (of the **while loop**) of the CG. Then:

$$||\mathbf{x}^{CG(t)} - \mathbf{x}^*||_{\mathbf{A}} \leq 2 \left(\frac{\sqrt{\kappa(\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{A})} + 1} \right)^t ||\mathbf{x}_0 - \mathbf{x}^*||_{\mathbf{A}} \tag{6.11}$$

For a matrix $\mathbf{A}$ whose condition number $\kappa(\mathbf{A})$ is large, one usually first constructs a *preconditioner* $\mathbf{M}$ of $\mathbf{A}$. Instead of using CG to solve $\mathbf{Ax} = \mathbf{b}$ directly, mathematically, we solve a symmetric version of the preconditioned system $\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}$:

$$\mathbf{M}^{-1/2}\mathbf{A}\mathbf{M}^{-1/2}\mathbf{y} = \mathbf{M}^{-1/2}\mathbf{b} \tag{6.12}$$

Then, we set $\mathbf{x} = \mathbf{M}^{-1/2}\mathbf{y}$.

Algorithmically, when we run preconditioned conjugate gradient (PCG) using $\mathbf{M}$ as the preconditioner, we need neither compute

$\mathbf{M}^{-1/2}\mathbf{b}$ nor perform the last $\mathbf{x} = \mathbf{M}^{-1/2}\mathbf{y}$ operation [155]. $\mathbf{M}^{-1/2}$ is only used symbolically. The main computation steps of PCG are matrix-vector products involving $\mathbf{A} \cdot \mathbf{M}^{-1}$, and a final multiplication of $\mathbf{M}^{-1}$ to remove the symbolic $\mathbf{M}^{-1/2}$ [155]. In other words, PCG reduces a linear system $\mathbf{Ax} = \mathbf{b}$ to multiple linear systems, of the form $\mathbf{M} \cdot \mathbf{z} = \mathbf{c}$.

Now consider a Laplacian linear system $\mathbf{L} \cdot \mathbf{x} = \mathbf{b}$. Let $\widetilde{\mathbf{L}}$ be a σ-spectral sparsifier of $\mathbf{L}$. Then for all $\mathbf{y} \in \mathbb{R}^n \perp \mathbf{1}$, with $\mathbf{z} = \widetilde{\mathbf{L}}^{-1/2}\mathbf{y}$:

$$\frac{\mathbf{y}\left(\widetilde{\mathbf{L}}^{-1/2}\mathbf{L}\widetilde{\mathbf{L}}^{-1/2}\right)\mathbf{y}}{\mathbf{y}^T\mathbf{y}} = \frac{\mathbf{zLz}}{\mathbf{z}\widetilde{\mathbf{L}}\mathbf{z}} \in [1/\sigma^2, \sigma^2].$$

Thus, for any constant σ (for example, $\sigma = 2$), $\widetilde{\mathbf{L}}^{-1/2}\mathbf{L}\widetilde{\mathbf{L}}^{-1/2}$ is well conditioned — its condition number is $O(1)$! So, PCG needs $O(\log(1/\epsilon))$ iterations to solve $\mathbf{L} \cdot \mathbf{x} = \mathbf{b}$ using $\widetilde{\mathbf{L}}$ as the preconditioner.

In each iteration, any matrix-vector product involving $\mathbf{L}$ can be computed in $O(\text{nnz}(\mathbf{L})) = O(n^2)$ time. We can use CG to solve linear systems of form $\widetilde{\mathbf{L}} \cdot \mathbf{z} = \mathbf{c}$. Thus, crucial to scalability of our algorithm, any matrix-vector product involving $\widetilde{\mathbf{L}}^{\dagger}$ can be computed in $O(n \cdot \text{nnz}(\widetilde{\mathbf{L}})) = \widetilde{O}(n^2)$ time, because $\text{nnz}(\widetilde{\mathbf{L}}) = \widetilde{O}(n)$. We have:

Theorem 6.26 (Scalable Dense Laplacian Solvers)**.** For any $\epsilon > 0$ and Laplacian matrix $\mathbf{L} = \mathbf{D_W} - \mathbf{W}$, one can solve:

$$\mathbf{L} \cdot \mathbf{x} = \mathbf{b}$$

to ϵ-precision in $\widetilde{O}(n^2 \log \frac{1}{\epsilon})$ time.

6.7 PageRank Completion of Networks

We will now conclude this chapter with a graph-theoretical reinterpretation of Theorem 3.7. This reinterpretation, first presented in [331], provides a partial answer to the following question:

> *Given a sparse network $G = (V, E, \mathbf{W})$, can we construct a complete-information network model that is consistent with G?*

Conceptually, we need to formulate the meaning of "a complete-information network model consistent with G." This question is relevant to *network completion* [165, 203, 245], a basic and fundamentally challenging problems in network analysis. One of the goals of network completion is to infer the missing links from sparse, observed network data.

Theorem 6.27 (PageRank Completion)**.** Suppose $G = (V, E, \mathbf{W})$ is a undirected weighted network. If G is connected, then for any restart constant $0 < \alpha < 1$, we can construct a network $\overline{G}_\alpha = (V, \bar{E}_\alpha, \overline{\mathbf{W}}_\alpha)$ with the following properties:

1. **(Complete Information)**: $(V, \bar{E}_\alpha)$ defines a complete graph with $|V|$ self-loops.
2. **(Degree Preserving)**: $\mathbf{D}_{\mathbf{W}} = \mathbf{D}_{\overline{\mathbf{W}}}$. i.e., $\mathbf{W} \cdot \mathbf{1} = \overline{\mathbf{W}} \cdot \mathbf{1}$.
3. **(PageRank Conforming)**: The row-sum of $\overline{G}$'s random-walk matrix $\mathbf{M}_{\overline{\mathbf{W}}} = \overline{\mathbf{W}}\mathbf{D}_{\overline{\mathbf{W}}}^{-1}$ is $\mathbf{PageRank}_{\mathbf{W},\alpha}$ (the PageRank of G with restart constant α).
4. **(Simultaneously Diagonalizable)**: The normalized Laplacian matrices of G and $\overline{G}$ — respectively, $\mathcal{L}_{\mathbf{W}}$ and $\mathcal{L}_{\overline{\mathbf{W}}}$ — are simultaneously diagonalizable.
5. **(Spectral Approximation)**: G and $\frac{1}{1-\alpha} \cdot \overline{G}_\alpha$ are $\frac{1}{\alpha}$-spectrally similar. In addition, $\mathcal{L}_{\mathbf{W}}$ and $\frac{1}{1-\alpha} \cdot \mathcal{L}_{\overline{\mathbf{W}}}$ are $\frac{1}{\alpha}$-spectrally similar.

Proof. Consider:

$$\overline{\mathbf{W}}_\alpha = \mathbf{D}_{\mathbf{W}} \cdot \mathbf{PPR}_{\mathbf{W},\alpha} \tag{6.13}$$

Let $\overline{G}_\alpha = (V, \bar{E}_\alpha, \overline{\mathbf{W}}_\alpha)$, where $\bar{E}_\alpha$ is the the nonzero pattern of $\overline{\mathbf{W}}_\alpha$. By Proposition 4.18, for any pair of nodes $u, v \in V$, $\mathbf{PPR}_{\mathbf{W},\alpha}[u, v]$ is equal to the probability that a run of random walk starting at u passes by v immediately before it restarts. Thus, if G is connected, then $\mathbf{PPR}_{\mathbf{W},\alpha}[u, v] > 0$, which implies that $\mathrm{nnz}(\overline{\mathbf{W}}_\alpha) = n^2$, and $(V, \bar{E}_\alpha)$ is

the complete graph with $|V|$ self-loops. By Theorem 3.7, $\overline{\mathbf{W}}_\alpha$ is symmetric because $\mathbf{W}$ is symmetric. Conditions (2) and (3) follows directly from Theorem 3.7.

We now prove Conditions (4) and (5).[2] By Eqn. (3.8), we can express $\overline{\mathbf{W}}$ as:

$$\overline{\mathbf{W}}_\alpha = \mathbf{D_W} \cdot \mathbf{PPR}_{\mathbf{W},\alpha} = \left(\alpha \sum_{k=0}^{\infty} (1-\alpha)^k \cdot \mathbf{D_W} \cdot \left(\mathbf{D_W^{-1}} \mathbf{W} \right)^k \right).$$

We compare the Laplacian matrices associated with $\mathbf{W}$ and $\overline{\mathbf{W}}$:

$$\begin{aligned} \mathbf{L_W} &= \mathbf{D_W} - \mathbf{W} = \mathbf{D_W^{1/2}} \left(\mathbf{I} - \mathbf{D_W^{-1/2}} \mathbf{W} \mathbf{D_W^{-1/2}} \right) \mathbf{D_W^{1/2}} \\ &= \mathbf{D_W^{1/2}} \mathcal{L}_\mathbf{W} \mathbf{D_W^{1/2}}. \end{aligned}$$

Similarly:

$$\mathbf{L}_{\overline{\mathbf{W}}} = \mathbf{D_W} - \overline{\mathbf{W}} = \mathbf{D_W^{1/2}} \mathcal{L}_{\overline{\mathbf{W}}} \mathbf{D_W^{1/2}}$$

where

$$\mathcal{L}_{\overline{\mathbf{W}}} = \mathbf{I} - \alpha \sum_{k=0}^{\infty} (1-\alpha)^k \cdot (\mathbf{D_W^{-1/2}} \mathbf{W} \mathbf{D_W^{-1/2}})^k.$$

Let $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$ be the n eigenvalues of $\mathbf{D_W^{-1/2}} \mathbf{W} \mathbf{D_W^{-1/2}}$. Let $\mathbf{v}_1, ..., \mathbf{v}_n$ denote the unit-length eigenvectors of $\mathbf{D_W^{-1/2}} \mathbf{W} \mathbf{D_W^{-1/2}}$ associated with eigenvalues $\lambda_1, \cdots, \lambda_n$, respectively. We have $|\lambda_i| \leq 1$. Let $\mathbf{\Lambda}$ be the diagonal matrix associated with $(\lambda_1, ..., \lambda_n)$. By the spectral theorem — i.e., the eigenvalue decomposition for symmetric matrices — we have:

$$\mathbf{U}^T \mathbf{D_W^{-1/2}} \mathbf{W} \mathbf{D_W^{-1/2}} \mathbf{U} = \mathbf{\Lambda} \tag{6.14}$$

$$\mathbf{U}\mathbf{U}^T = \mathbf{U}^T \mathbf{U} = \mathbf{I} \tag{6.15}$$

Therefore:

$$\begin{aligned} \mathbf{L_W} &= \mathbf{D_W^{1/2}} \mathbf{U}\mathbf{U}^T \left(\mathbf{I} - \mathbf{D_W^{-1/2}} \mathbf{W} \mathbf{D_W^{-1/2}} \right) \mathbf{U}\mathbf{U}^T \mathbf{D_W^{1/2}} \\ &= \mathbf{D_W^{1/2}} \mathbf{U} \left(\mathbf{I} - \mathbf{\Lambda} \right) \mathbf{U}^T \mathbf{D_W^{1/2}}. \end{aligned}$$

[2] Thanks to Dehua Cheng for assisting this proof.

Similarly:

$$\begin{aligned}
\mathbf{L}_{\overline{\mathbf{W}}_\alpha} &= \mathbf{D}_{\mathbf{W}} - \overline{\mathbf{W}}_\alpha = \mathbf{D}_{\mathbf{W}}^{1/2} \mathcal{L}_{\overline{\mathbf{W}}} \mathbf{D}_{\mathbf{W}}^{1/2} \\
&= \mathbf{D}_{\mathbf{W}}^{1/2} \left(\mathbf{I} - \alpha \sum_{k=0}^{\infty} (1-\alpha)^k \cdot (\mathbf{D}_{\mathbf{W}}^{-1/2} \mathbf{W} \mathbf{D}_{\mathbf{W}}^{-1/2})^k \right) \mathbf{D}_{\mathbf{W}}^{1/2} \\
&= \mathbf{D}_{\mathbf{W}}^{1/2} \mathbf{U} \left(\mathbf{I} - \alpha \sum_{k=0}^{\infty} (1-\alpha)^k \cdot \mathbf{U}^T (\mathbf{D}_{\mathbf{W}}^{-1/2} \mathbf{W} \mathbf{D}_{\mathbf{W}}^{-1/2})^k \mathbf{U} \right) \mathbf{U}^T \mathbf{D}_{\mathbf{W}}^{1/2} \\
&= \mathbf{D}_{\mathbf{W}}^{1/2} \mathbf{U} \left(\mathbf{I} - \alpha \sum_{k=0}^{\infty} (1-\alpha)^k \cdot \mathbf{\Lambda}^k \right) \mathbf{U}^T \mathbf{D}_{\mathbf{W}}^{1/2} \\
&= \mathbf{D}_{\mathbf{W}}^{1/2} \mathbf{U} \left(\mathbf{I} - \frac{\alpha}{\mathbf{I} - (1-\alpha)\mathbf{\Lambda}} \right) \mathbf{U}^T \mathbf{D}_{\mathbf{W}}^{1/2}.
\end{aligned}$$

The derivation above has proved Condition (4). To prove Condition (5), consider an arbitrary $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$. With $\mathbf{y} = \mathbf{U}^T \mathbf{D}_{\mathbf{W}}^{1/2} \mathbf{x}$, we have:

$$\frac{\mathbf{x}^T \frac{1}{1-\alpha} \mathbf{L}_{\overline{\mathbf{W}}} \mathbf{x}}{\mathbf{x}^T \mathbf{L}_{\mathbf{W}} \mathbf{x}} = \frac{\mathbf{y} \frac{1}{1-\alpha} \mathcal{L}_{\overline{\mathbf{W}}} \mathbf{y}}{\mathbf{y}^T \mathbf{L}_{\mathbf{W}} \mathbf{y}} = \frac{1}{1-\alpha} \cdot \frac{\mathbf{y}^T \left(\mathbf{I} - \frac{\alpha}{\mathbf{I} - (1-\alpha)\mathbf{\Lambda}} \right) \mathbf{y}}{\mathbf{y}^T \left(\mathbf{I} - \mathbf{\Lambda} \right) \mathbf{y}}$$

This ratio is in the interval of:

$$\left[\min_{\lambda : |\lambda| \leq 1} \frac{1}{1-(1-\alpha)\lambda}, \max_{\lambda : |\lambda| \leq 1} \frac{1}{1-(1-\alpha)\lambda} \right] = \left[\frac{1}{2-\alpha}, \frac{1}{\alpha} \right].$$

□

Remarks *We rescale* $\mathbf{L}_{\overline{\mathbf{W}}}$ *and* $\mathcal{L}_{\overline{\mathbf{W}}}$ *by* $\frac{1}{1-\alpha}$ *because* $\overline{G}$ *has self-loops of magnitude* $\alpha \mathbf{D}_{\mathbf{W}}$*. In other words,* $\overline{G}$ *only uses* $(1-\alpha)$ *fraction of its weights for the connections between different nodes in* V*.*

In contrast to spectral sparsification (Theorems 6.10 and 6.12), we can view the construction of $\overline{G}$ *as the "spectral densification" or "spectral completion" of network* G*.*

Together, Theorem 6.27 and Proposition 6.5 and Proposition 6.9 then imply that the PageRank graph $\frac{1}{1-\alpha} \cdot \overline{G}_\alpha$ *(with self-loops removed) approximately preserves cuts, flows, and effective resistances of network* G *up to a factor of* $\frac{1}{\alpha}$*.*

7

Electrical Flows: Laplacian Paradigm for Network Analysis

Linear programs and convex programs are exemplary algorithmic primitives for polynomial-time computation. Not only can they be solved in polynomial time, but convex programming techniques have transformed the field of combinatorial optimization. They have enabled polynomial-time solution or approximation to numerous fundamental problems that are important to real-world applications.

For over half a century, researchers in the field of algorithm design have developed efficient algorithms for many fundamental problems in vast application domains. Breakthroughs in one area often lead to breakthroughs in others. Thus, the field continuously encourages holistic approaches to identify general algorithmic methodologies and techniques, as well as mathematical characterizations of problems that can be solved efficiently by these techniques. Together, the combined efforts have steadily expanded the *library* of solvable problems and advanced the body of algorithmic techniques that will ultimately lead to the development of new algorithms. The design of scalable algorithms is no exception. In this chapter, we focus on a basic scalable algorithmic primitive at the intersection of network analysis and numerical linear algebra. This primitive has a natural connection to electrical flows. It

also arises in Gaussian random fields, graph spanning trees, random walks, convex optimization, and finite-element methods.

7.1 SDD Primitive and Its Scalability

A matrix $\mathbf{M}$ is *symmetric diagonally dominant* (SDD) if $\mathbf{M} = \mathbf{M}^T$ and:

$$m_{i,i} > \sum_{j \neq i} |m_{i,j}|, \quad \text{for all } i.$$

$\mathbf{M}$ is *weakly* SDD, if $\forall i$, $m_{i,i} \geq \sum_{j \neq i} |m_{i,j}|$. $\mathbf{M}$ is an SDDM matrix if it is SDD and all its off-diagonal entries are non-positive. Every Laplacian matrix $\mathbf{D_W} - \mathbf{W}$ is an SDDM matrix with weak diagonal dominance.

Definition 7.1 (SDD Primitive)**.** Given a linear system: $\mathbf{M} \cdot \mathbf{x} = \mathbf{b}$, with SDD $\mathbf{M} \in \mathbb{R}^{n \times n}$, $\mathbf{b} \in \mathbb{R}^n$, and $\epsilon > 0$, find an $\widetilde{\mathbf{x}} \in \mathbb{R}^n$ such that:

$$||\widetilde{\mathbf{x}} - \mathbf{M}^{-1}\mathbf{b}||_{\mathbf{M}} \leq \epsilon ||\mathbf{M}^{-1}\mathbf{b}||_{\mathbf{M}} \tag{7.1}$$

Like linear programs to optimization, linear systems are fundamental to numerical analysis and scientific computing. In practice, one can have a small degree of imprecision, as modeled by ϵ in Definition 7.1. The parameter ϵ could either be set according to machine precision, or to domain-specific precision. Thus, it is desirable that a solver's dependence on ϵ be logarithmic.

Linear systems can be solved in polynomial time. However, the scalability of linear solvers (and matrix multiplication) continues to be among the most fascinating subjects in algorithm design. The best theoretical bound for solving linear systems and dense matrix multiplication is $O(n^{2.372873})$ [351, 102]. When solvers' dependences on ϵ are required to be logarithmic, $O(n^{2.372873})$ remains the fastest bound known for obtaining ϵ-approximate solutions. For SDD matrices, the following holds:

Theorem 7.2 (Spielman-Teng)**.** SDD primitive can be solved to ϵ-precision in (expected) $\widetilde{O}_n(1) \cdot \text{nnz}(\mathbf{M}) \cdot \log \frac{1}{\epsilon}$ time.

This result makes no assumption on the structure of the non-zero entries, which is crucial to its algorithmic applicability. The complexity of the solver adapts to the density of the matrix, and is nearly linear in the number of the non-zeros. Therefore, scalable SDD solvers can be used in any task from network analysis to optimization, where SDD matrices usually arise.

In a comprehensive 141-page article [344] titled, "Lx = b," Nisheeth Vishnoi gave a survey of the algorithmic and mathematical backgrounds for establishing Theorem 7.2. Readers interested in the proof of this theorem are referred to Vishnoi's article, which also contains the much improved SDD solvers due to Koutis, Miller, Peng [216, 215] and Kelner, Orecchia, Sidford, and Zhu [200].

In the rest of this chapter, we will mainly focus on some basic scalable algorithms that are enabled and inspired by SDD primitive.

7.2 Electrical Flow and Laplacian Linear Systems

We first focus on Laplacian linear systems (Definition 6.24), which is a special case of SDD primitive. We start with their connection with electrical flow and random-walks.

A Trip Down Memory Lane

Electrical flow is a subject that many of us learned in high school physics class: A voltage difference of V at a resistor induces an electrical flow (current). According to Ohm's law, if the resistor has resistance R, then the magnitude of the current is $I = V/R$. The *energy* of this current is I^2R. One can also make a circuit by connecting resistors together. The two basic connection patterns are:

- **Series circuit**: a chain of resistors — like edges of a path in a graph — so the same current flows through all the resistors.

- **Parallel circuit**: multiple resistors sharing corresponding endpoints — like parallel edges in a graph — so that the same voltage is applied to each resistor.

Suppose the resistors have resistances $R_1, ..., R_k$, respectively. In a series circuit, a voltage difference V across the path induces a current flow of magnitude $I = V/(R_1 + ... + R_k)$. Thus, the *effective resistance* of the series circuit is:

$$R_1 + ... + R_k.$$

In this setting, the voltage difference across the i^{th} resistor is $\frac{R_i}{R_1+...+R_k}$. In other words, with a common flow I, the voltage difference V is distributed across the resistors proportionally to their resistances. In a parallel circuit, with voltage difference V, $I_i = V/R_i$ unit of current flows across resistor i. So, the total current is:

$$I_1 + ... + I_k = V \cdot \left(\frac{1}{R_1} + ... + \frac{1}{R_k} \right).$$

Thus, the *effective resistance* of the parallel circuit is:

$$\left(\frac{1}{R_1} + ... + \frac{1}{R_k} \right)^{-1}.$$

Complex *series-parallel* circuits can be built by an iterative process: Starting with one resistor between s and t, one can then repeatedly replace a resistor either by a series or by a parallel circuit. One can inductively determine the effective resistance between s and t. For a voltage difference V between s and t, one can also inductively calculate the currents flow across all resistors in the circuit. In fact, it takes time linear in the number of resistors in the circuit to compute effective resistance and electrical flows. So, we learned a scalable algorithm in high school. By the end of high school, many of us have mastered the above construction and effective-resistance/electrical flow computation.

Electric Flow in Weighted Networks

> *Every weighted, undirected network $G = (V, E, \mathbf{W})$ can be viewed as a* network of resistors*. For every $e \in E$, one may view edge e of affinity w_e as a resistor of resistance:*
>
> $$r_e = 1/w_e.$$

Now, no longer in high school, we can use any variables to denote current and voltage differences. While our network is undirected, electrical current has directions. Along any edge, it flows from the endpoint with the higher voltage to the endpoint with the lower voltage. To track the direction, we arbitrarily orient each edge in $e \in E$, *a priori*. Thus, edges incident to a node $v \in V$ can now be classified into two sets:

- (1) $E^{in}(v)$: consisting of all edges oriented towards v
- (2) $E^{out}(v)$: consisting of edges oriented away from v

In other words, if an edge has positive flow and is in $E^{in}(v)$, then the flow is towards v. But if the flow on the edge is negative and is in $E^{in}(v)$, then the flow is away from v.

We now define the space of network flows that include electrical flows, as well as maximum flows and minimum-cost flows.

Definition 7.3 (s-t Flows)**.** An *s-t flow* is a function $\mathbf{f} : E \to \mathbb{R}$ that satisfies the *flow-conservation constraints* (also known as Kirchhoff's current law):

$$\sum_{e \in E^{out}(v)} f_e - \sum_{e \in E^{in}(v)} f_e = 0 \quad \text{for all } v \in V \setminus \{s, t\} \tag{7.2}$$

Two basic quantities associated with an s-t flow are:

Definition 7.4 (Value and Energy of s-t Flows)**.** For a given s-t flow $\mathbf{f}$ in a network $G = (V, E, \mathbf{W})$:

$$\text{Energy}_{\mathbf{W}}(\mathbf{f}) := \sum_e r_e f_e^2 = \sum_e \frac{1}{w_e} f_e^2 \tag{7.3}$$

$$\text{Value}(\mathbf{f}) := \sum_{e \in E^{out}(s)} f_e - \sum_{e \in E^{in}(s)} f_e \tag{7.4}$$

Like many structures in physics, electrical flow also follows the *minimum total potential energy principle*.

Definition 7.5 (Electrical Flows and Effective Resistance)**.** *Electrical flow*, denoted by $\mathbf{f}_{\mathbf{W}}^{s\to t}$, from s to t in a network $G = (V, E, \mathbf{W})$ is an s-t flow ***of value*** 1 that minimizes $\text{Energy}_{\mathbf{W}}(\mathbf{f})$ among all unit s-t flows $\mathbf{f}$ in G. The *effective resistance*, $\text{ER}_{\mathbf{W}}(s,t)$, between s and t in G is:

$$\text{ER}_{\mathbf{W}}(s,t) = \text{Energy}_{\mathbf{W}}(\mathbf{f}_{\mathbf{W}}^{s\to t}) \tag{7.5}$$

In the discussion below, let $\mathbf{B}$ be the *incidence matrix* of the unweighted graph (V, E), according to the *a priori* orientation:

$$\mathbf{B}[:, e] = \mathbf{1}_v - \mathbf{1}_u, \quad \forall e = (u, v) \in E^{in}(v) \cap E^{out}(u).$$

Then, Kirchhoff's current law requires:

$$\mathbf{B} \cdot \mathbf{f} = \mathbf{1}_s - \mathbf{1}_t \tag{7.6}$$

Let $\mathbf{C}_{\mathbf{W}}$ denote the $|E| \times |E|$ diagonal matrix, indexed by edges in E, where $\mathbf{C}_{\mathbf{W}}[e, e] = w_e$. The s-t electrical flow $\mathbf{f}_{\mathbf{W}}^{s\to t}$ is then the solution to the following mathematical program:

$$\begin{aligned} \textbf{Minimize} \quad & \text{Energy}_{\mathbf{W}}(\mathbf{f}) = \textstyle\sum_e \frac{1}{w_e} f_e^2 = ||\mathbf{C}_{\mathbf{W}}^{-1/2}\mathbf{f}||_2^2 \\ \textbf{Subject to} \quad & \mathbf{B} \cdot \mathbf{f} = \mathbf{1}_s - \mathbf{1}_t \end{aligned}$$

According to Ohm's law, a current of f_e across e comes with a voltage difference f_e/w_e between the endpoints of e. It is well known (see [55]) that when $\mathbf{f}$ is an electrical flow, the voltage differences across all edges of G fit magically into a potential field $\boldsymbol{\phi} \in \mathbb{R}^{|V|}$, such that:

$$\mathbf{f}_{u,v} = w_{u,v} \cdot (\phi(v) - \phi(u)).$$

In other words, when $\mathbf{f}$ is an electrical flow, the following holds:

$$\mathbf{f} = \mathbf{C}_{\mathbf{W}} \cdot \mathbf{B}^T \boldsymbol{\phi} \tag{7.7}$$

To see the connection of electrical flow to Laplacian systems, recall $\mathbf{L}_{\mathbf{W}} = \mathbf{B} \cdot \mathbf{C}_{\mathbf{W}} \cdot \mathbf{B}^T$. Then, Eqn. (7.6) and Eqn. (7.7) implies:

$$\mathbf{L}_{\mathbf{W}} \cdot \boldsymbol{\phi} = \mathbf{B} \cdot \mathbf{C}_{\mathbf{W}} \cdot \mathbf{B}^T \boldsymbol{\phi} = \mathbf{B} \cdot \mathbf{f} = \mathbf{1}_s - \mathbf{1}_t \tag{7.8}$$

Let $\boldsymbol{\phi}_{\mathbf{W}}^{s\to t}$ be the solution to (7.8). Then $\boldsymbol{\phi}_{\mathbf{W}}^{s\to t}$ defines the potential field that generates the electrical flow $\mathbf{f}_{\mathbf{W}}^{s\to t}$ in G: $\mathbf{f}_{\mathbf{W}}^{s\to t} = \mathbf{C}_{\mathbf{W}} \cdot \mathbf{B}^T \boldsymbol{\phi}_{\mathbf{W}}^{s\to t}$. Thus:

$$\mathbf{f}_{\mathbf{W}}^{s\to t} = \mathbf{C}_{\mathbf{W}}\mathbf{B}^T \cdot \mathbf{L}_{\mathbf{W}}{}^{\dagger}(\mathbf{1}_s - \mathbf{1}_t).$$

Proposition 7.6 (Potential Energy)**.** For any weighted network $G = (V, E, \mathbf{W})$, and $s, t \in V$:

$$||\boldsymbol{\phi}_{\mathbf{W}}^{s\to t}||_{\mathbf{L}_{\mathbf{W}}}^2 = \text{Energy}_{\mathbf{W}}(\mathbf{f}_{\mathbf{W}}^{s\to t}).$$

Proof. It follows directly from the definition of $\mathbf{L}_{\mathbf{W}}$ norm and $\text{Energy}_{\mathbf{W}}(\mathbf{f})$, and the derivation above. □

Applying SDD primitive, i.e., Theorem 7.2, to Eqn. (7.8), we can scalably compute an approximate electrical flow as well as the effective resistance between s and t in network G.

Theorem 7.7 (Scalable Electrical Flows)**.** For any weighted network $G = (V, E, \mathbf{W})$, $s, t \in V$, and $\epsilon > 0$, one can compute, in $\widetilde{O}(n+\text{nnz}(\mathbf{W}) \log \frac{1}{\epsilon})$ time, a vector of potential fields $\tilde{\boldsymbol{\phi}}_{\mathbf{W}}^{s\to t}$ and an s-t unit-flow $\widetilde{\mathbf{f}}_{\mathbf{W}}^{s\to t}$, such that:

(1) $||\tilde{\boldsymbol{\phi}}_{\mathbf{W}}^{s\to t} - \boldsymbol{\phi}_{\mathbf{W}}^{s\to t}||_{\mathbf{L}_{\mathbf{W}}} \leq \epsilon ||\boldsymbol{\phi}_{\mathbf{W}}^{s\to t}||_{\mathbf{L}_{\mathbf{W}}} = \epsilon \cdot \text{Energy}_{\mathbf{W}}(\mathbf{f}_{\mathbf{W}}^{s\to t})$

(2) $\text{Energy}_{\mathbf{W}}(\widetilde{\mathbf{f}}_{\mathbf{W}}^{s\to t}) \geq (1+\epsilon) \cdot \text{Energy}_{\mathbf{W}}(\mathbf{f}_{\mathbf{W}}^{s\to t})$

(3) $\text{ER}_{\mathbf{W}}(s,t) \leq \text{Energy}_{\mathbf{W}}(\widetilde{\mathbf{f}}_{\mathbf{W}}^{s\to t}) \leq (1+\epsilon) \cdot \text{ER}_{\mathbf{W}}(s,t)$

Spielman and Srivastava [311] went one step further. They developed a scalable data structure for answering the following question, which is important for spectral sparsification:

> *What is the effective resistance between a pair of vertices $u, v \in V$ in G?*

Their data structure uses the fact that $\mathcal{M}_{\mathbf{W}}^{\text{ER}} = (V, \text{ER}_{\mathbf{W}})$ is a metric space. They applied Johnson-Lindenstrauss projection to reduce the

computation from $O(n)$-dimensions to $O(\log n/\epsilon^2)$ dimensions. Then, using scalable Laplacian solvers, they constructed an $O(\log n/\epsilon^2) \times n$ matrix $\widetilde{\mathbf{Z}}$, such that for all $u, v \in V$:

$$(1-\epsilon) \cdot \mathrm{ER}_{\mathbf{W}}(u,v) \leq ||\widetilde{\mathbf{Z}}(\mathbf{1}_u - \mathbf{1}_v)||^2 \leq (1+\epsilon) \cdot \mathrm{ER}_{\mathbf{W}}(u,v).$$

Note that $||\widetilde{\mathbf{Z}}(\mathbf{1}_u - \mathbf{1}_v)||^2$ can be evaluated in $O(\log n/\epsilon^2)$ time.

Theorem 7.8 (Scalable Structures for Effective Resistances)**.** For any $G = (V, E, \mathbf{W})$, and $\epsilon > 1$, one can construct a data structure in $\widetilde{O}(\frac{n+\mathrm{nnz}(\mathbf{W})}{\epsilon^2})$ time. Using this structure, when given $u, v \in V$, one can in $O(\log n/\epsilon^2)$ time compute $\widetilde{\mathrm{ER}}_{\mathbf{W}}(u,v)$, such that:

$$(1-\epsilon) \cdot \mathrm{ER}_{\mathbf{W}}(u,v) \leq \widetilde{\mathrm{ER}}_{\mathbf{W}}(u,v) \leq (1+\epsilon) \cdot \mathrm{ER}_{\mathbf{W}}(u,v).$$

7.3 Spectral Approximation and Spectral Partitioning

Recall that the Fielder value of a weighted and connected network $G = (V, E, \mathbf{W})$ is the second smallest eigenvalue $\lambda_2(\mathbf{L}_{\mathbf{W}})$ of its Laplacian $\mathbf{L}_{\mathbf{W}}$. Using SDD primitive, we can scalably approximate the Fiedler value of any network. We also produce a vector whose Rayleigh quotient is approximately equal to the Fiedler value.

Definition 7.9 (Approximate Fiedler Value and Vector)**.** For a weighted network $G = (V, E, \mathbf{W})$, $\mathbf{v}$ is an ϵ-*approximate Fiedler vector* of $\mathbf{L}_{\mathbf{W}}$ if $\mathbf{v} \perp \mathbf{1}$ and:

$$\lambda_2(\mathbf{L}_{\mathbf{W}}) \leq \frac{\mathbf{v}^T \mathbf{L}_{\mathbf{W}} \mathbf{v}}{\mathbf{v}^T \mathbf{v}} \leq (1+\epsilon) \cdot \lambda_2(\mathbf{L}_{\mathbf{W}}).$$

We use SDD primitive inside the classical inverse power method.

Theorem 7.10 (Scalable Spectral Approximation)**.** For any $\epsilon > 0$ and connected $G = (V, E, \mathbf{W})$, we can compute an ϵ-approximate Fiedler vector in $\widetilde{O}_n(1) \cdot \mathrm{nnz}(\mathbf{W}) \cdot \frac{\log(1/\epsilon)}{\epsilon}$ time, with high probability.

Proof. (Sketch) Assume the eigenvalues of $\mathbf{L_W}$ — from the smallest to the largest — are $\lambda_1 = 0$, λ_2, ...,λ_n. Let $\mathbf{v}_i$ be the eigenvector of λ_i. Note that $\mathbf{v}_1$ is proportional to $\mathbf{1}$. We start with a unit random vector $\mathbf{r} \perp \mathbf{v}_1$, which can be expressed as $\mathbf{r} = \sum_{i=2}^{n} c_i \mathbf{v}_i$. Note that $\mathbf{L}_\mathbf{W}^\dagger \mathbf{r} = \sum_{i=1}^{n} c_i \lambda_i^{-1} \mathbf{v}_i$. In general, for positive integer $t \geq 1$:

$$(\mathbf{L}_\mathbf{W}^\dagger)^t \mathbf{r} = \sum_{i=2}^{n} c_i \lambda_i^{-t} \mathbf{v}_i.$$

With high probability, c_2 is not too small. By choosing $t = \Theta(\log(n/\epsilon)/\epsilon)$, we can compute an ϵ-approximate Fiedler vector using this inverse power method. The error can be bounded by standard numerical analysis. Using SDD primitive, we can scalably compute $\mathbf{L}_\mathbf{W}^\dagger \mathbf{x}$, for any $\mathbf{x} \perp \mathbf{1}$, to a desired precision. □

The scalable Fiedler-vector approximation can be used in combination with the spectral characterization of planar graphs and graphs with bounded genus or forbidden minors. The following algorithmic result — which applies Sweep to approximate Fiedler vectors — is an immediate consequence of Theorems 5.47 and 4.8.

Corollary 7.11 (Scalable Fiedler Sweep)**.** Suppose $G = (V, E)$ is a connected, constant-degree planar graph (or a graph with bounded genus or forbidden minor). Then, in time linear in $n = |V|$, we can identify a set $S \subset V$ (with $|S| \leq |V|/2$) of cut ratio $\frac{\text{cut}(S,\bar{S})}{|S|} \leq O(\frac{1}{\sqrt{n}})$ from any $O(1)$-approximate Fiedler vector.

In general, to apply SDD primitive to Theorem 4.8, we need to approximate the eigenvector of the second smallest eigenvalue associated with the normalized Laplacian $\mathbf{D}_\mathbf{W}^{-1/2} \mathbf{L_W} \mathbf{D}_\mathbf{W}^{-1/2}$. The smallest eigenvalue of this matrix is also 0, whose eigenvector is $\mathbf{D}^{1/2}\mathbf{1}$. Note that $(\mathbf{D}_\mathbf{W}^{-1/2} \mathbf{L_W} \mathbf{D}_\mathbf{W}^{-1/2})^\dagger = \mathbf{D}_\mathbf{W}^{1/2} \mathbf{L_W}^\dagger \mathbf{D}_\mathbf{W}^{1/2}$. So, scalable Laplacian solvers can be used to scalably multiply $(\mathbf{D}_\mathbf{W}^{-1/2} \mathbf{L_W} \mathbf{D}_\mathbf{W}^{-1/2})^\dagger$ with any vectors in $\mathbb{R}^n$. Thus, the inverse power method has a scalable implementation for normalized Laplacians.

Corollary 7.12 (Scalable Cheeger Cut)**.** If $G = (V, E, \mathbf{W})$ is a weighted network and λ_2 is the second smallest eigenvalue of its normalized Laplacian $\mathbf{D}_{\mathbf{W}}^{-1/2}\mathbf{L}_{\mathbf{W}}\mathbf{D}_{\mathbf{W}}^{-1/2}$, then, we can scalably compute a subset S with $\text{conductance}_{\mathbf{W}}(S) = O(\sqrt{\lambda_2})$.

7.4 Learning from Labeled Network Data

In 1963, Tutte [335] proved perhaps one of the most beautiful theorems for planar graphs:

Theorem 7.13 (Tutte Embedding)**.** Every tri-connected planar graph has a straight-line drawing on the plane in which the embedding of every face is convex.

Tutte's construction was extremely elegant. It first chooses an arbitrary face F in the planar graph $G = (V, E)$ and chooses an $|F|$-*gon*, a convex polygon P with $|F|$ corners in the plane. It then"pins down" the vertices of F to the points of P, in the manner that respects the order of the points of F.

Let's assume P — e.g., a regular $|F|$-gon — is placed so that the average of its points are $\mathbf{0}$. Let $\boldsymbol{\rho}(z)$ denote the embedded point for $z \in F$. Tutte's construction then uses the "spring embedding" that simultaneously embeds each vertex $v \in V - F$ to a point $\boldsymbol{\rho}(v)$ inside P, such that, $\boldsymbol{\rho}(v)$ is *the barycenter* of the set of points, $\{\boldsymbol{\rho}(u) : u \in N_G(v)\}$, associated with v's neighbors. In other words:

$$\boldsymbol{\rho}(v) = \frac{1}{|N_G(v)|} \cdot \left(\sum_{u \in N_G(v)} \boldsymbol{\rho}(u) \right), \quad \forall v \in V - F \tag{7.9}$$

Each v defines two equations, respectively, for x- and y-coordinate. Let $d_v = |N_G(v)|$ be the degree of v and $\boldsymbol{\rho}(v) = (x_v, y_v)$. Let $\mathbf{x} = (...x_v...)$ and $\mathbf{y} = (...y_v..)$, and $\boldsymbol{\chi}_{P,x}$ be the following vector: If v is the k^{th} vertex of face F, then $\boldsymbol{\chi}_{P,x}[v]$ is equal to the x-coordinate associated with the

k^{th} corner of P; otherwise, $\boldsymbol{\chi}_{P,x}[v] = 0, \forall v \in V - F$. We can similarly define $\boldsymbol{\chi}_{P,y}$. By Eqn. (7.9), Tutte's embedding is the solution of the following two Laplacian linear systems:

$$\mathbf{Lx} = \boldsymbol{\chi}_{P,x} \tag{7.10}$$

$$\mathbf{Ly} = \boldsymbol{\chi}_{P,y} \tag{7.11}$$

Therefore, not only the solution to these two linear systems (when given F and P) is unique, it now can be scalably computed (to near machine precision) by SDD primitive.

Theorem 7.14 (Scalable Tutte Embedding)**.** There exists a scalable algorithm for computing Tutte's Embedding for tri-connected planar graphs.

Tutte's barycentric spring-embedding can be viewed a network inference problem. It infers the values associated with vertices in $V - F$ from the values associated with vertices in F. From this perspective, Tutts's theorem can be viewed as the first result that connects graph learning to Laplacian linear systems. Indeed, Tutte's embedding has been extended for solving more general tasks of learning on networks, as illustrated by Zhou, Huang, and Schölkopf [358].

The task of [358] is to learn from labeled and unlabeled data on a graph. Let $G = (V, E)$ be a strongly connected (aperiodic) directed network and $S \subset V$ is a subset of V. Suppose we have a labeling function $\mathbf{y}$ that assigns a label from a classification set $Y = \{1, -1\}$ to each vertex in S, and assigns 0 to vertices in $V - S$. Let $\mathcal{H}(\mathcal{V})$ be the set of "fitness" functions of the form $V \to \mathbb{R}$ for labeling vertices in the network. The mathematical goal of this learning problem is to find a function $\mathbf{f} \in \mathcal{H}(\mathcal{V})$ that optimizes the following objective:

$$\textbf{minimize } ||\mathbf{f} - \mathbf{y}||^2 + \mu \cdot L(\mathbf{f}) \tag{7.12}$$

where μ is a regularization parameter, and

$$L(\mathbf{f}) = \frac{1}{2} \sum_{(u,v) \in E} \pi_u \cdot \mathbf{M}[u, v] \left(\frac{f_u}{\sqrt{\pi_u}} - \frac{f_v}{\sqrt{\pi_v}} \right)^2 \tag{7.13}$$

is the *smoothness* measure based upon a stochastic perspective of random walks: $\boldsymbol{\pi}$ is the stationary distribution of the standard random walk on the network with transition matrix $\mathbf{M} = \mathbf{A}^T \cdot \left(\mathbf{D}^{out}\right)^{-1}$, where $\mathbf{A}$ is the adjacency matrix of G.

It is shown in [358] that the optimal solution $\mathbf{f}^*$ to the mathematical programming defined by (7.12) is the solution to the following linear system:

$$\left(\boldsymbol{\Pi} - \frac{\mu}{1+\mu}\frac{\boldsymbol{\Pi}\mathbf{M} + \mathbf{M}^T\boldsymbol{\Pi}}{2}\right)\left(\boldsymbol{\Pi}^{-1/2}\mathbf{f}^*\right) = \left(1 - \frac{\mu}{1+\mu}\right)\boldsymbol{\Pi}^{1/2}\mathbf{y}$$

where $\boldsymbol{\Pi}$ the diagonal matrix with $\boldsymbol{\Pi}[v,v] = \pi_v$. Zhou, Huang, and Schölkopf applied scalable SDD primitive to obtain the following result, by observing that the following matrix is SDD (Lemma 8.20):

$$\left(\boldsymbol{\Pi} - \frac{\mu}{1+\mu}\frac{\boldsymbol{\Pi}\mathbf{M} + \mathbf{M}^T\boldsymbol{\Pi}}{2}\right) \tag{7.14}$$

Theorem 7.15 (Scalable Learning from Network Data)**.** The graph learning problem formulated by (7.12) has a scalable solution.

7.5 Sampling From Gaussian Markov Random Fields

Graphical models neatly connect statistics with graph theory. Recall that a *Markov random field* is a set of random variables whose dependencies are represented by an undirected graph. For many machine learning tasks, we need to draw samples from an underlying graphical model. Thus, the central algorithmic question is:

> *How efficiently can we draw a sample from the distribution of a Markov random field?*

When the joint probability distribution of the random variables is strictly positive, the most popular sampling method is the *Gibbs process*. This Markov Chain Monte Carlo (MCMC) method repeatedly

resamples each variable conditioning upon the values of their graph neighbors. The mathematical correctness of the Gibbs process follows from the celebrated Hammersley-Clifford Theorem [163, 181, 214]: In limit, the mean and covariance of samples produced by the Gibbs process are the exact ones defined by the random field.

An important subclass of the Markov random field is the *Gaussian Markov random field*, where a multivariate normal distribution is specified by its *precision matrix* $\mathbf{\Lambda}$ and potential vector $\boldsymbol{h}$. The precision matrix is the inverse of the covariance matrix of the multivariate normal distribution. In other words:

$$\Pr\left[\mathbf{x}|\mathbf{\Lambda}, \boldsymbol{h}\right] \propto \exp(-\frac{1}{2}\mathbf{x}^T\mathbf{\Lambda}\mathbf{x} + \boldsymbol{h}^T\mathbf{x}).$$

Indeed, sampling from multivariate probability distributions is one of the most fundamental problems in machine learning and statistical inference. Although the Gibbs process is widely popular and often effective for sampling from Markov random fields, it has some algorithmic limitations. First, while Gibbs sampling is exact in limit, many distributions require it to run for a significant number of iterations to achieve a reliable estimate [181, 237]. Theoretically, the Gibbs process is not a scalable sampling algorithm. For distributions that require the Gibbs process to run many iterations to converge, this sampling process is also highly sequential [86]. In practice, one cannot run the Gibbs process to the limit. Thus, finite termination introduces errors to samples' mean and covariance, making the practical Gibbs process an approximate sampling algorithm.

An alternative approach to obtain random samples from a *Gaussian Markov random field* $(\mathbf{\Lambda}, \boldsymbol{h})$ of n variables is to use numerical methods. These algorithms use the following three-step numerical framework:

1. Compute the mean $\boldsymbol{\mu}$ of the distribution by solving the linear system $\mathbf{\Lambda}\boldsymbol{\mu} = \boldsymbol{h}$.

2. Compute a "square-root" factor of the covariance matrix by finding an inverse square-root factor of the precision matrix $\mathbf{\Lambda}$:

$$\mathbf{C}\mathbf{C}^T = \mathbf{\Lambda}^{-1}.$$

3. Suppose the inverse square-root factor $\mathbf{C}$ of Step 2 is an $n \times n'$ matrix. Then, we can obtain a sample of the Gaussian Markov random field $(\mathbf{\Lambda}, \boldsymbol{h})$ by matrix-vector product:
   (a) Draw a random vector $\boldsymbol{z} = (z_1, z_2, \ldots, z_{n'})^T$, where z_i is an i.i.d. sample from the standard normal distribution.
   (b) Return $\boldsymbol{x} = \mathbf{C}\boldsymbol{z} + \boldsymbol{\mu}$.

Mathematically, $\mathbf{C}$ is a map from the *standard* multivariate normal distribution to the distribution of the Gaussian Markov random field given by $(\mathbf{\Lambda}, \boldsymbol{h})$. If $\mathbf{C}$ has a *sparse representation*, then Step 3 has a scalable implementation.

To illustrate the connection to SDD primitive, we consider a special case of Gaussian Markov random fields, in which the precision matrices are symmetric H-matrices. Symmetric H-matrices have *generalized diagonal dominance*: A symmetric matrix $\mathbf{\Lambda} \in \mathbb{R}^{n \times n}$ is an *H-matrix*, if there exists a positive diagonal scaling matrix $\mathbf{D}$ such that:

$$\mathbf{D\Lambda D} \text{ is SDD} \tag{7.15}$$

H-precision matrices were first considered in [263, 179] when studying a parallel implementation of the Gibbs process, known as *Hogwild Gaussian Gibbs sampling*. This implementation distributes the graphical model with an H-precision matrix over a number of machines. Each machine then performs its "local" Gibbs process. Periodically, these machines exchange values of the "boundary" variables. Both papers [263, 179] address the question of how often these machines need to exchange boundary variables in order to ensure the distributed Gibbs process to converge properly.

The following result of Daitch and Spielman [105] reduces many numerical problems on symmetric H-matrices to SDD matrices.

Lemma 7.16 (Scaling of Symmetric H Matrices)**.** Given a symmetric H-matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$, one can compute a positive diagonal matrix $\mathbf{D}$ such that $\mathbf{DMD}$ is SDD, with $\widetilde{O}_n(1)$ calls to SDD primitive.

Note that for any precision matrix $\mathbf{\Lambda}$ and diagonal scaling matrix $\mathbf{D}$, the solution to $\mathbf{\Lambda}\boldsymbol{\mu} = \boldsymbol{h}$ is:

$$\boldsymbol{\mu} = \mathbf{\Lambda}^{-1}\boldsymbol{h} = \mathbf{D}(\mathbf{D}\mathbf{\Lambda}\mathbf{D})^{-1}\mathbf{D}\boldsymbol{h}.$$

If $\mathbf{D}\mathbf{\Lambda}\mathbf{D}$ is SDD, then $\mathbf{\Lambda}\boldsymbol{\mu} = \boldsymbol{h}$ can be solved in $\widetilde{O}(\text{nnz}(\mathbf{\Lambda})$ time by SDD primitive. Thus, when working with Gaussian Markov random fields with H-precision matrices, Lemma 7.16 scalably reduces the sampling problem to finding a square-factor of $\mathbf{\Lambda}^{-1}$. Let $\bar{\mathbf{\Lambda}} = \mathbf{D}\mathbf{\Lambda}\mathbf{D}$. Suppose $\bar{\mathbf{C}}\bar{\mathbf{C}}^T = \bar{\mathbf{\Lambda}}^{-1}$. Then: $(\mathbf{D}\bar{\mathbf{C}})(\mathbf{D}\bar{\mathbf{C}})^T = \mathbf{D}\bar{\mathbf{\Lambda}}^{-1}\mathbf{D} = \mathbf{\Lambda}^{-1}$. Thus, $\mathbf{D}\bar{\mathbf{C}}$ is an inverse square-root factor of $\mathbf{\Lambda}$. So, it is sufficient to find an inverse-square-root factor of the SDD matrix $\bar{\mathbf{\Lambda}}$.

The following useful property of SDD matrices [56] extends the well-known fact regarding graph Laplacians:

Lemma 7.17 (Scalable Width-2 Factor)**.** Let $\mathbf{M} \in \mathbb{R}^{n\times n}$ be an SDD matrix with $2m$ non-zero off-diagonal entries. Then, there exists an $n \times (m+n)$ matrix $\mathbf{B}$ with at most 2 non-zeros per column such that:

$$\mathbf{M} = \mathbf{B}\mathbf{B}^T.$$

Moreover, $\mathbf{B}$ can be constructed in $O(m+n)$ time.

Proof. Let $E = \{(u,v) : m_{u,v} \neq 0\}$. The proof is essentially the same as Lemma 6.13. For each $e = (u,v) \in E$, set the e^{th} column of $\mathbf{B}$ as:

$$\mathbf{b}_e := \begin{cases} \sqrt{m_{u,v}}(\mathbf{1}_u + \mathbf{1}_v) & \text{if } m_{u,v} > 0 \\ \sqrt{|m_{u,v}|}(\mathbf{1}_u - \mathbf{1}_v) & \text{if } m_{u,v} < 0 \end{cases}$$

Then, $\mathbf{\Delta} = \mathbf{M} - \mathbf{B}\mathbf{B}^T = \mathbf{M} - \sum_{e\in E}\mathbf{b}_e\mathbf{b}_e^T$ is a diagonal matrix, where the u^{th} diagonal entry is $\delta_u = m_{u,u} - \sum_{v\neq u}|m_{u,v}|$. Because $\mathbf{M}$ is SDD, $\delta_u > 0$. Thus, $\mathbf{\Delta} = \sum_u(\sqrt{\delta_u}\mathbf{1}_u)(\sqrt{\delta_i}\mathbf{1}_u)^T$. But setting the $m + u^{th}$ column of $\mathbf{B}$ to be $\sqrt{\delta_u}\mathbf{1}_u$, we have $\mathbf{M} = \mathbf{B}\mathbf{B}^T$, and $\mathbf{B}$ satisfies the condition of the lemma. □

Therefore, as shown in [86], one can use SDD primitive to numerically generate approximate Gaussian samples.

Theorem 7.18 (Scalable GMRF Sampling)**.** Suppose $(\mathbf{\Lambda}, \boldsymbol{h})$ is a Gaussian Markov random field of n variables with a symmetric H-precision matrix $\mathbf{\Lambda}$. Then, for any $\epsilon > 0$, one can generate a random sample $\boldsymbol{s} \in \mathbb{R}^n$, in $\widetilde{O}(n + \text{nnz}(\mathbf{\Lambda})$ time, from a Gaussian Markov random field $(\widetilde{\mathbf{\Lambda}}, \boldsymbol{h})$, such that $\widetilde{\mathbf{\Lambda}}$ is $(1+\epsilon)$-spectrally similar to $\mathbf{\Lambda}$.

Proof. Let $m = \text{nnz}(\mathbf{\Lambda}$. The sampling algorithm first applies Lemma 7.16 to compute a positive diagonal matrix $\mathbf{D}$, such that $\bar{\mathbf{\Lambda}} = (\mathbf{D\Lambda D})$ is SDD. It then applies Lemma 7.17 to construct $\mathbf{B} \in \mathbb{R}^{n,m+n}$, such that $\bar{\mathbf{\Lambda}} = \mathbf{B}\mathbf{B}^T$. Note that, $\text{nnz}(\mathbf{B}) \leq 2(m+n)$ and $\text{nnz}(\bar{\mathbf{\Lambda}}) = O(m+n)$.

According to Theorem 7.2, any linear system with matrix $\bar{\mathbf{\Lambda}}$ can be scalably solved to ϵ-precision. In fact, the scalable solver can be viewed as a linear operator $\mathbf{Z}$, such that $\mathbf{Z}$ is $(1+\epsilon)$-spectrally similar $\bar{\mathbf{\Lambda}}^{-1}$ [319]. Let $\tilde{\mathbf{C}} = \mathbf{DZB}$. Then:

$$\widetilde{\mathbf{\Lambda}}^{-1} = \tilde{\mathbf{C}}\tilde{\mathbf{C}}^T = \mathbf{DZBB}^T\mathbf{ZD}$$

is $(1 + \Theta(\epsilon))$-spectrally similar to $\mathbf{\Lambda}^{-1}$. Thus, if $\mathbf{y}$ is drawn randomly from the standard $(m+n)$ multivariate normal distribution, then $\mathbf{x} = \mathbf{DZBy}$ is a random sample from the Gaussian Markov random field defined by $(\widetilde{\mathbf{\Lambda}}, \boldsymbol{h})$. The sampling algorithm is then scalable, because for any $\mathbf{y} \in \mathbb{R}^{m+n}$, $\mathbf{DZBy}$ can be computed in $\widetilde{O}(m+n)$ time. □

7.6 Scalable Newton's Method via Spectral Sparsification

Although the sampling algorithm above is scalable, it has one drawback: In order to generate an n-dimensional sample from a Gaussian Markov random field, this algorithm needs to first generate an $(m+n)$-dimensional sample from the standard multivariate normal distribution. Recall that m is the number of edges in the input graphical model. When $m \gg n$, the algorithm's use of randomness is highly inefficient. In fact, many sampling algorithms for Gaussian Markov random fields are inefficient in their use of randomness. For example, whenever the Gibbs process requires super-linear steps to converge, the number

of random scalars that must be drawn is also super-linear. The goal to design sampling algorithms that are both scalable and efficient in the use of randomness leads to the following beautiful algorithmic question:

> *Is there a scalable sampling algorithm that uses only n random scalars to generate a random sample from an n-variable Gaussian Markov random field (with an H-precision matrix)?*

We can apply scalable spectral sparsification — the key algorithmic component of SDD primitive — directly to input graphical models. For example, we can use Theorem 6.22 to reduce the number of edges in graphical models to $8n \log n/\epsilon^2$. The almost scalable algorithm of Lee and Sun [221] can further decrease the number of edges to $O(n/\epsilon^2)$.

Using spectral sparsification to speed up Newton's method, Cheng *et al.* [86] sharply reduced the number of random scalars for sampling. Their scalable sampling algorithm is optimal for Gaussian Markov random fields with SDDM precision matrices. In which case, the algorithm requires n random scalars to generate a random sample from an n-variable random field. For Gaussian Markov random fields with H-precision matrices, their result is also essentially optimal:

Theorem 7.19 (Scalable Sparse-Newton Sampling)**.** Suppose $(\mathbf{\Lambda}, \boldsymbol{h})$ is an n-variable Gaussian Markov random field with an H-precision matrix $\mathbf{\Lambda}$. Then, for any $\epsilon > 0$, one can in $\widetilde{O}(n + \text{nnz}(\mathbf{\Lambda})$ time, from a $2n$-dimensional standard Gaussian sample, generate a random sample $\boldsymbol{s} \in \mathbb{R}^n$ from a Gaussian Markov random field $(\widetilde{\mathbf{\Lambda}}, \boldsymbol{h})$, such that $\widetilde{\mathbf{\Lambda}}$ is $(1+\epsilon)$-spectrally similar to $\mathbf{\Lambda}$.

In this section, to illustrate the usefulness of spectral sparsification, we will highlight the algorithm that proves this theorem.

Basic Idea: Sparse Newton's Method

We will focus on the case when the precision matrix $\mathbf{\Lambda}$ is an SDDM matrix. The construction can be extended to H-precision matrices: We

first use Daitch-Spielman's scaling (Lemma 7.16) to reduce H-precision matrices to SDD matrices (Theorem 7.18). We then use an extension [86] to the well-known Gremban's reduction in spectral graph theory to further reduce the computation to finding an inverse-square-root factor of an SDDM matrix, which is twice as large. This reduction is responsible for the $2n$ normal variables rather than n normal variables in the sampling result stated in Theorem 7.19.

The sampling algorithm of Theorem 7.18 uses the inverse square-root factor of the precision matrix that has dimension $n \times (m + n)$. Although this factor can be constructed by the scalable SDD primitive, it can be too "rectangular" to be efficient. The longer dimension determines the number of random scalars needed to drawn a Gaussian sample. In order to reduce the number of random scalars to n for SDDM precision matrices, we need a scalable algorithm for constructing a sparse-representation of an $n \times n$ inverse square-root factor:

Definition 7.20 (Canonical Inverse Square-Root Factor)**.** For an $n \times n$ SDDM matrix $\mathbf{\Lambda}$, the *canonical inverse square-root factor* of $\mathbf{\Lambda}$ is the $n \times n$ matrix $\mathbf{B}$ such that $\mathbf{B}\mathbf{B}^T = \mathbf{\Lambda}^{-1}$.

Cheng *et al.*'s algorithm [86] uses Newton's method to compute a canonical inverse square-root factor of an SDDM matrix. Newton's method is one of the most powerful methods for numerical approximation. However, it has one major drawbacks for matrix equations:

> *Newton's method uses dense matrix multiplications, even when the original matrix is sparse. This is a major obstacle to its application in scalable network analysis, where graphs are usually sparse. Although Newton's method may converge rapidly, which provides a numerical framework for designing not only sequential but also parallel algorithms, its intermediate computation could be prohibitively expensive for handling big data.*

The matrix factoring algorithm of [86] uses the following key observation: For SDDM precision matrices, the intermediate matrices introduced by Newton's method are also SDDM matrices. Therefore, if we

can efficiently replace these dense matrices with their spectral sparsifiers, then we can make Newton's iterations scalable.

Basic Setup: Matrix Polynomials

To formulate Newton iterations for computing the canonical inverse-square root of an SDDM matrix, let us first draw some intuitions from (normalized) Laplacian matrices:

$$\mathbf{L_W} = \mathbf{D_W} - \mathbf{W} = \mathbf{D_W}^{1/2}(\mathbf{I} - \mathbf{D_W}^{-1/2}\mathbf{W}\mathbf{D_W}^{-1/2})\mathbf{D_W}^{1/2}.$$

To construct the inverse square-root of $\mathbf{L_W}$, it is sufficient to compute the inverse square-root of normalized Laplacian $(\mathbf{I} - \mathbf{D_W}^{-1/2}\mathbf{W}\mathbf{D_W}^{-1/2})$. Note that $\mathbf{D_W}^{-1/2}\mathbf{W}\mathbf{D_W}^{-1/2}$ is similar to the random-walk transition matrix $\mathbf{W}\mathbf{D_W}^{-1}$.

In general, every SDDM matrix $\mathbf{\Lambda}$ can be expressed in the following form: $\mathbf{\Lambda} = c^{-1}(\mathbf{I} - \mathbf{X})$.[1] Therefore, we only need to consider SDDM matrices of form $\mathbf{I} - \mathbf{X}$. We view $\mathbf{X}$ as a symmetrized random walk transition matrix. To set up our discussion, let us first recall that for computing $(1-x)^{-1/2}$ involving a scalar x, Newton's method solves the following equation: $f(y) = y^{-2} - (1-x)$.

As $f'(y) = -2y^{-3}$, the iterative steps in Newton's method are:

$$y_{k+1} = y_k - f(y_k)/f'(y_k) = y_k\left[1 + (1 - (1-x)y_k^2)/2\right].$$

It is more illustrative to examine the residue of each step. Let $r_k = 1 - (1-x)y_k^2$ be the residue of y_k. Then, these iterative steps become:

$$y_{k+1} = y_k\left(1 + \frac{r_k}{2}\right) = y_0\prod_{i=0}^{k}\left(1 + \frac{r_i}{2}\right).$$

Suppose we start with $y_0 = 1$, which implies $r_0 = x$, and $y_1 = (1 + x/2)$ and $r_1 = 1 - (1-x)y_1^2 = 1 - (1-x)(1+x/2)^2$. Thus, after

[1] To construct this representation, suppose $\mathbf{\Lambda} = \mathbf{D} - \mathbf{A}$ with diagonal $\mathbf{D} = \operatorname{diag}(d_1, d_2, \ldots, d_n)$, and τ be the condition number of $\mathbf{\Lambda}$. Let $\kappa = \max(2, \tau)$ and $c = (1 - 1/\kappa)/(\max_i d_i)$, so that all eigenvalues of $c(\mathbf{D} - \mathbf{A})$ lie between $\frac{1}{2\kappa}$ and $2 - \frac{1}{2\kappa}$. Thus, we can rewrite $c(\mathbf{D} - \mathbf{A})$ as an SDDM matrix $\mathbf{I} - \mathbf{X}$, where all entries of $\mathbf{X} = \mathbf{I} - c(\mathbf{D} - \mathbf{A})$ are non-negative, and $\rho(\mathbf{X}) \leq 1 - \frac{1}{2\kappa}$.

first step of Newton's method, we have:

$$(1-x)^{-1} = (1-r_1)^{-1}y_1^2 = \left(1 - \frac{3}{4}x^2 - \frac{1}{4}x^3\right)^{-1}\left(1+\frac{x}{2}\right)^2 \quad (7.16)$$

Therefore:

$$(1-x)^{-1/2} = \left(1 - \frac{3}{4}x^2 - \frac{1}{4}x^3\right)^{-1/2}\left(1+\frac{x}{2}\right) \quad (7.17)$$

In general, subsequent steps imply that these residues satisfy the following equation:

$$(1-r_k)^{-1/2} = \left(1 - \frac{3}{4}r_k^2 - \frac{1}{4}r_k^3\right)^{-1/2}\left(1+\frac{r_k}{2}\right) \quad (7.18)$$

Eqn. (7.17) is a remarkable formula for the inverse square-root of $(1-x)$. It states that the computation of $(1-x)^{-1/2}$ can be reduced to the computation of $(1-\frac{3}{4}x^2-\frac{1}{4}x^3)^{-1/2}$. If the latter is C, then the former is $C(1+\frac{x}{2})$. Furthermore, if $x<1$, then $\frac{3}{4}x^2-\frac{1}{4}x^3 < x$. Thus, the iteration converges! The fact that $\frac{3}{4}x^2-\frac{1}{4}x^3$ has no linear term suggests a quadratic convergence.

Eqn. (7.16) naturally extends to SDDM matrix $(\mathbf{I}-\mathbf{X})$:

$$(\mathbf{I}-\mathbf{X})^{-1} = \left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right)\left(\mathbf{I} - \frac{3}{4}\mathbf{X}^2 - \frac{1}{4}\mathbf{X}^3\right)^{-1}\left(\mathbf{I}+\frac{1}{2}\mathbf{X}\right) \quad (7.19)$$

This equation reduces canonical inverse square-root factor of $(\mathbf{I}-\mathbf{X})$ to canonical inverse square-root factor of $(\mathbf{I}-\frac{3}{4}\mathbf{X}^2-\frac{1}{4}\mathbf{X}^3)$. In other words, from $\mathbf{C}_1 = (\mathbf{I}-\frac{3}{4}\mathbf{X}^2-\frac{1}{4}\mathbf{X}^3)^{-1/2}$, we obtain $\mathbf{C}_1(\mathbf{I}+\mathbf{X}/2) = (\mathbf{I}-\mathbf{X})^{-1/2}$. Moreover, Newton's method rapidly converges because the spectral radius $\rho(\frac{3}{4}\mathbf{X}^2+\frac{1}{4}\mathbf{X}^3)$ is quadratically smaller than $\rho(\mathbf{X})$.

> *This is all excellent and well known. However, Newton's method is not scalable because*
>
> $$\mathbf{I} - \frac{3}{4}\mathbf{X}^2 - \frac{1}{4}\mathbf{X}^3$$
>
> *could be a dense matrix!*

Sparsification of Random-Walk Matrix Polynomials

Cheng *et al.* [86] observed the following (via a proof by induction):

Proposition 7.21 (Convex Combination of SDDM Data)**.** If $\mathbf{I} - \mathbf{X}$ is SDDM, then $(\mathbf{I} - \frac{3}{4}\mathbf{X}^2 - \frac{1}{4}\mathbf{X}^3)$ is also SDDM. In fact, for any integer $d > 1$, $\mathbf{I} - \sum_r^t \alpha_r \mathbf{X}^r$ is also SDDM, for any $\sum_r^t \alpha_r = 1$.

For any $\epsilon > 0$, $(\mathbf{I} - \frac{3}{4}\mathbf{X}^2 - \frac{1}{4}\mathbf{X}^3)$ has a $(1+\epsilon)$-spectral sparsifier of $\widetilde{O}(n)$ nonzero entries, according to Theorems 6.10 and 6.22.

> *Therefore, if we can efficiently replace these SDDM matrix polynomials by spectrally similar sparsifiers, then we can make Newton's method scalable.*

However, the algorithms of Theorems 6.10 and 6.22 cannot be directly used to sparsify $(\mathbf{I}-\frac{3}{4}\mathbf{X}^2-\frac{1}{4}\mathbf{X}^3)$, because these algorithms require that the input matrix be explicitly given. In Newton's method, the matrix we need to sparsify is implicitly given as a matrix polynomial. To understand the graph-theoretical connection of Newton's matrix polynomials, let's consider the special case when $\mathbf{X} = \mathbf{D}_{\mathbf{W}}^{-1/2}\mathbf{W}\mathbf{D}_{\mathbf{W}}^{-1/2}$. Then, we have:

$$\begin{aligned}\mathbf{I} - \mathbf{X} &= (\mathbf{I} - \mathbf{D}_{\mathbf{W}}^{-1/2}\mathbf{W}\mathbf{D}_{\mathbf{W}}^{-1/2}) \\ &= \mathbf{D}_{\mathbf{W}}^{-1/2}(\mathbf{D}_{\mathbf{W}} - \mathbf{W})\mathbf{D}_{\mathbf{W}}^{-1/2} \\ &= \mathbf{D}_{\mathbf{W}}^{-1/2}\mathbf{L}_{\mathbf{W}}\mathbf{D}_{\mathbf{W}}^{-1/2}\end{aligned}$$

where $\mathbf{L}_{\mathbf{W}}$ is the Laplacian matrix of $\mathbf{W}$. More generally:

$$\mathbf{I} - \frac{3}{4}\mathbf{X}^2 - \frac{1}{4}\mathbf{X}^3 = \mathbf{D}_{\mathbf{W}}^{-1/2}\left(\mathbf{D}_{\mathbf{W}} - \frac{3}{4}\mathbf{D}_{\mathbf{W}}(\mathbf{D}_{\mathbf{W}}^{-1}\mathbf{W})^2 - \frac{1}{4}\mathbf{D}_{\mathbf{W}}(\mathbf{D}_{\mathbf{W}}^{-1}\mathbf{W})^3\right)\mathbf{D}_{\mathbf{W}}^{-1/2}$$

Note that, $\mathbf{D}_{\mathbf{W}} - \frac{3}{4}\mathbf{D}_{\mathbf{W}}(\mathbf{D}_{\mathbf{W}}^{-1}\mathbf{W})^2 - \frac{1}{4}\mathbf{D}_{\mathbf{W}}(\mathbf{D}_{\mathbf{W}}^{-1}\mathbf{W})^3$, the term in the middle, is the Laplacian matrix of the networks composed of two-step and three-step random-walk interactions.

Thus, the algorithmic need to speed up Newton's method leads to the following basic spectral sparsification problem:

Definition 7.22 (Random-Walk Polynomials: Sparsification)**.** Suppose $G = (V, E, \mathbf{W})$ is a weighted network, and $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_t)$ is a vector satisfying $\sum_{r=1}^{t} \alpha_r = 1$. Let:

$$\mathbf{L}_{\mathbf{W},\boldsymbol{\alpha}} := \mathbf{D}_{\mathbf{W}} - \sum_{r=1}^{t} \alpha_r \mathbf{D}_{\mathbf{W}} \cdot \left(\mathbf{D}_{\mathbf{W}}^{-1}\mathbf{W}\right)^r \tag{7.20}$$

For $\epsilon > 0$, find a weighted network $\widetilde{G}_{\boldsymbol{\alpha}} = (V, \widetilde{E}_{\boldsymbol{\alpha}}, \widetilde{\mathbf{W}}_{\boldsymbol{\alpha}})$ such that (1) $\text{nnz}(\widetilde{\mathbf{W}}_{\boldsymbol{\alpha}}) = \widetilde{O}(n/\epsilon^2)$ and (2) $\mathbf{L}_{\widetilde{\mathbf{W}}_{\boldsymbol{\alpha}}}$ is $(1+\epsilon)$-spectrally similar to $\mathbf{L}_{\mathbf{W},\boldsymbol{\alpha}}$.

The matrix power $\mathbf{D}_{\mathbf{W}}\left(\mathbf{D}_{\mathbf{W}}^{-1}\mathbf{W}\right)^r$ of Eqn. (7.20) encodes the probability distribution of G's paths with r edges. Such a path $\mathbf{p}$ can be specified as a sequence of vertices $\mathbf{p} = (p_0, ..., p_r) \in V^{r+1}$ such that $(p_{i-1}, p_i) \in E$, $\forall i \in [r]$. Let $\text{ends}(\mathbf{p}) := (p_0, p_r)$. Mathematically, $\mathbf{D}_{\mathbf{W}}\left(\mathbf{D}_{\mathbf{W}}^{-1}\mathbf{W}\right)^r$ assigns the following weight to path $\mathbf{p}$:

$$w_{\mathbf{p}} = \frac{\prod_{i=1}^{r} w_{p_{i-1},p_i}}{\prod_{i=1}^{r-1} d_{p_i}}.$$

For any $u, v \in V$, let $P_{u,v}^r := \left\{\mathbf{p} \in V^{r+1} : \text{ ends}(\mathbf{p}) = (u, v)\right\}$. This set defines the weight of the edge between (u, v) in the following weighted network: $G^{(r)} = (V, E^{(r)}, \mathbf{W}^{(r)})$, where:

$$\mathbf{W}^{(r)}[u, v] := \sum_{\mathbf{p} \in P_{u,v}^r} w_{\mathbf{p}}.$$

We can inductively prove:

Proposition 7.23. $\mathbf{L}_{\mathbf{W},\boldsymbol{\alpha}}$ is the Laplacian matrix of a weighted network whose affinity matrix $\mathbf{W}_{\boldsymbol{\alpha}}$ is:

$$\mathbf{W}_{\boldsymbol{\alpha}}[u, v] = \sum_{r=1}^{t} \alpha_r \mathbf{W}^{(r)}.$$

In other words, $\mathbf{W}_{\boldsymbol{\alpha}}$ is the convex combination — according to $\boldsymbol{\alpha}$ — of networks defined by random walks:

$$\mathbf{L}_{\mathbf{W},\boldsymbol{\alpha}} = \sum_{r=1}^{t} \alpha_r \mathbf{L}_{\mathbf{W}^{(r)}} = \sum_{r=1}^{t} \alpha_r \left(\mathbf{D}_{\mathbf{W}^{(r)}} - \mathbf{W}^{(r)}\right) \tag{7.21}$$

Even for sparse G, $\mathbf{W}_{\boldsymbol{\alpha}}$ encodes a huge number of paths.

Is there a scalable algorithm to sparsify $\mathbf{L}_{\mathbf{W},\boldsymbol{\alpha}}$*?*

Cheng *et al.* [86] gave an affirmative answer to this question with a simple sampling algorithm.

Theorem 7.24 (Scalable Sparsification of Random-Walks)**.** Suppose $G = (V, E, \mathbf{W})$ is a weighted network, and $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_t)$ is a non-negative vector satisfying $\sum_{r=1}^{t} \alpha_r = 1$. One can construct, in time

$$O(t^2 \cdot m \cdot \log^2 n \cdot \frac{1}{\epsilon^2})$$

a $(1+\epsilon)$-spectral sparsifier with $O(n \log n \cdot \frac{1}{\epsilon^2})$ non-zeros for $\mathbf{L}_{\mathbf{W},\boldsymbol{\alpha}}$, where $m = \text{nnz}(\mathbf{W})$.

Proof. (Sketch) Let us focus on sparsifying $G^{(r)} = (V, E^{(r)}, \mathbf{W}^{(r)})$, for a given r. The algorithm of [86] takes two basic steps to sparsify:

$$\mathbf{L}_{\mathbf{W}}^{(r)} = \mathbf{D}_{\mathbf{W}^{(r)}} - \mathbf{W}^{(r)} = \mathbf{D}_{\mathbf{W}} - \mathbf{D}_{\mathbf{W}}(\mathbf{D}_{\mathbf{W}}^{-1}\mathbf{W})^r.$$

The first one is the critical step to achieve scalability. It obtains an initial sparsifier with $O(r \cdot m \log n \frac{1}{\epsilon^2})$ non-zeros for $\mathbf{L}_{\mathbf{W}}^{(r)}$. The second step then applies the standard spectral sparsification algorithms (Theorems 6.22, 6.10) to further reduce the number of non-zeros to $O(n \log n \frac{1}{\epsilon^2})$.

The first step uses an elegant identity regarding random walks: For any $\mathbf{p} \in V^{r+1}$, let us extend the measure of resistance to non-simple paths:

$$Z(\boldsymbol{p}) := \text{resistance}(\mathbf{p}) = \sum_{i=1}^{r} \frac{1}{w_{p_{i-1},p_i}} \tag{7.22}$$

Proposition 7.25 (Random-Walk Identity)**.**

$$\sum_{\boldsymbol{p}:\ |\mathbf{p}|=r} w(\boldsymbol{p}) \cdot Z(\boldsymbol{p}) = m \cdot r \tag{7.23}$$

Eqn. (7.23) suggests the following algorithm, $\mathsf{PathSampling}(G, r)$, to sample r-step random-walks. It selects a path $\mathbf{p} \in V^{r+1}$ from G with probability equal to:

$$\tau_{\boldsymbol{p}} := \frac{w(\boldsymbol{p})Z(\boldsymbol{p})}{m \cdot r}.$$

Algorithm: $\mathsf{PathSampling}(G, r)$

1: Choose a random edge e, uniformly from E, and a random integer k, uniformly from $[r]$.
2: Return the path $\mathbf{p}$ obtained by taking, respectively, $(k-1)$-step and $(r-k)$-step random walks from the two endpoints of e.

Using $\mathsf{PathSampling}(G, r)$, we can support the execution of:

$$\widetilde{G}^{(r)} = \mathsf{GraphSampling}(G^{(r)}, \widetilde{m}, (\tau_{\boldsymbol{p}} : \mathbf{p} \in V^{r+1}))$$

without explicitly constructing weighted edges in $G^{(r)}$.

Recall that, for each pair $u, v \in V$, the weighted edge between (u, v) in $G^{(r)}$ is formed by a collection of parallel edges, one for each $\mathbf{p}$ with $\text{ends}(\mathbf{p}) = (u, v)$. Although Theorem 6.22 extends to networks with parallel edges, $Z(\boldsymbol{p})$ usually differs from $\text{ER}_{\mathbf{W}^{(r)}}(\text{ends}(\mathbf{p}))$, which prevents its direct application. The analysis of [86] uses the following key spectral inequalities:

Proposition 7.26 (Spectral Support of Random Walks)**.** For any symmetric weighted matrix $\mathbf{W}$:

- $\frac{1}{2}\mathbf{L}_{\mathbf{W}} \preceq \mathbf{L}^{(r)}_{\mathbf{W}} \preceq r\mathbf{L}_{\mathbf{W}}$, for any odd positive integer r.
- $\mathbf{L}^{(2)}_{\mathbf{W}} \preceq \mathbf{L}^{(r)}_{\mathbf{W}} \preceq \frac{r}{2}\mathbf{L}^{(2)}_{\mathbf{W}}$, for any even positive integer r.

Using this fact, we can show:

Corollary 7.27 (Effective Resistance of Random Walks)**.** For any $G = (V, E, \mathbf{W})$ and $\mathbf{p} \in V^{r+1}$, $\text{ER}_{\mathbf{W}^{(r)}}(\text{ends}(\mathbf{p})) \leq 2 \cdot Z(\boldsymbol{p})$.

The following extension [198, 311] of Theorem 6.22 can then be applied.

Theorem 7.28 (Sparsification by Oversampling)**.** Suppose $G = (V, E, \mathbf{W})$ is a network (with possible parallel edges), $\epsilon > 0$, and $h : E \to \mathbb{R}$ satisfies: $\forall e \in E$, $\mathrm{ER}_{\mathbf{W}}(e) \leq h(e)$, Then, with high probability, $\tilde{G} = \mathsf{GraphSampling}(G, \widetilde{m}, (\tau_e, e \in E))$ satisfies the following, with $\tau(e) = w(e)h(e)$, and $\widetilde{m} = O\left(\sum_{e\in E} \tau_e \cdot \log n \cdot \frac{1}{\epsilon^2}\right)$:

$$(1-\epsilon)\mathbf{L}_{\mathbf{W}} \preccurlyeq \mathbf{L}_{\tilde{\mathbf{W}}} \preccurlyeq (1+\epsilon)\mathbf{L}_{\mathbf{W}} \tag{7.24}$$

Thus, by Theorem 7.28, it is sufficient to sample $\widetilde{O}_n(1) \cdot (m \cdot r)$ r-length paths to sparsify $\mathbf{L}_{\mathbf{W}}^{(r)}$. For a given $\boldsymbol{\alpha}$, we independently sparsify each random-walk Laplacian matrix in $\sum_{r=1}^{t} \alpha_r \mathbf{L}_{\mathbf{W}}^{(r)}$. The union of these sparsifiers has $\widetilde{O}_n(1) \cdot (t \cdot m \cdot r)$ non-zeros. We then apply Theorem 6.22 to further reduce the size of the sparsifier to $O(n \log n \frac{1}{\epsilon^2})$. □

Sparse Newton Chain for the Inverse Square-Root

Theorem 7.24 can be extended to sparsify polynomial, $\mathbf{I} - \sum_r^t \alpha_r \mathbf{X}^r$, associated with any SDDM matrix $\mathbf{I} - \mathbf{X}$ and $\sum_r^t \alpha_r = 1$. Thus, we can sparsify (the cubic) matrix polynomial of Eqn. (7.19) in the Newton iteration. This sparse Newton's method produces a sequence of sparse matrices $\mathbf{X}_0, \mathbf{X}_1, ..., \mathbf{X}_d$, where $\mathbf{X}_0 = \mathbf{X}$, for computing $(\mathbf{I} - \mathbf{X})^{-1/2}$. Formally, for a target $\epsilon > 0$, we set $d = O(\log(\kappa/\epsilon))$, and $\epsilon_i = \epsilon/d$. For $i \in [d]$, we iteratively compute a sparse matrix $\mathbf{X}_i$, such that $\mathbf{I} - \mathbf{X}_i$ is a $(1 + \epsilon_i)$-spectral sparsifier of $(\mathbf{I} - \frac{3}{4}\mathbf{X}_{i-1}^2 - \frac{1}{4}\mathbf{X}_{i-1}^3)$.

Using the standard numerical analysis of Newton's method, we can prove that $\mathbf{I}$ is $(1 + \epsilon_d)$-spectrally similar to $\mathbf{I} - \mathbf{X}_d$. Thus, the matrix product:

$$\prod_{i=0}^{d-1} \left(\mathbf{I} + \frac{\mathbf{X}_i}{2}\right) \tag{7.25}$$

is the canonical square root of a matrix that is $(1+\epsilon)$-spectrally similar to $(\mathbf{I} - \mathbf{X})^{-1}$. In other words, for any SDDM matrix $\mathbf{M} = \mathbf{I} - \mathbf{X}$ and

$\epsilon > 0$, Eqn. (7.25) is a sparse representation of an $n \times n$ matrix $\tilde{\mathbf{C}}$, such that $\tilde{\mathbf{C}}\tilde{\mathbf{C}}^T$ is $(1+\epsilon)$-spectrally similar to $(\mathbf{I}-\mathbf{X})^{-1}$. This "sparse" factorization has the following desirable property: For any $\boldsymbol{x} \in \mathbb{R}^n$, we can compute $\tilde{\mathbf{C}}\boldsymbol{x}$ in time $\widetilde{O}(m+n\log\frac{1}{\epsilon})$, where $\widetilde{O}$ contains a polynomial factor of $\log\kappa$. Theorem 7.19 then follows accordingly.

7.7 Laplacian Paradigm

The examples of this chapter illustrate an emerging paradigm, the *Laplacian Paradigm* [329], for the design of scalable network-analysis algorithms. This paradigm is built on a family of scalable algebraic and numerical algorithms, highlighted by SDD primitive. This scalable family — in addition to SDD primitive [319, 215, 200, 97, 217] — includes graph algorithms for:

- Spectral sparsification [317, 311]
- Low-stretch spanning trees [11, 125, 5]
- PageRank estimation [177, 22, 65]
- Local clustering and graph partitioning [318, 22, 92, 24]

The scalable family also includes algorithms for electrical flow computation [319], effective-resistance query structures [105, 311], and random-walk estimators [113], that use SDD primitive. This is a rapidly expanding family of scalable algorithms, building on new applicational, mathematical, and algorithmic insights.

> *To apply the Laplacian paradigm to solve a problem defined on massive networks or big matrices, we attempt to reduce the computational and optimization problem to one or more algebraic or spectral graph-theoretical problems, solvable by SDD or other scalable primitives.*

Our earlier examples of Fiedler vector approximation, spectral partitioning, Tutte's embedding, and machine learning illustrate the direct applications of the Laplacian paradigm. Even when attempts do

not immediately yield scalable solutions — like polynomial-time algorithm design with other algorithmic paradigms — the pursuit itself often leads us to mathematical structures — particularly of an algebraic and numerical nature — that are useful for scalable solution or approximation.

Compared with commonly-used graph-theoretical techniques such as breadth-first search, depth-first search, and divide-and-conquer, SDD primitive appears to offer more global mixtures of network data. It provides a scalable operator for the inverse of graph Laplacians as well as a scalable solver for electrical flows and effective resistances. The algebraic and numerical underpinnings of SDD primitive resemble those in linear and convex programming. But, the fact that an SDD matrix can be inverted as fast as sorting edge weights provides a powerful framework for scalable computation.

For example, the direct application of SDD primitive (of Section 7.5) only yields a semi-efficient scheme for Gaussian sampling. But this numerical approach together with the spectral sparsification technique (developed for SDD primitive) ultimately lead to a scalable realization of Newton's method for this basic sampling problem.

The recent breakthroughs in maximum-flow and minimum-cut approximation are also exemplary applications of the Laplacian paradigm. We now highlight these developments below.

A Success Story: Max Flows and Min Cuts

The Maximum s-t Flow and the Minimum s-t Cut are two of the most fundamental combinatorial optimization problems [297, 8]. The input for both problems includes a capacitated network, $G = (V, E, \mathbf{C})$, where for each edge $e \in E$, $\mathbf{C}$ assigns a capacity $c_e \in \mathbb{Z}^+$. The input also includes two *distinguished nodes*: a *source* node $s \in V$ and a *sink* node $t \in V$.

In the discussion below, we assume the input network is *undirected*, i.e., $c_{(u,v)} = c_{(v,u)}$ for all $u, v \in V$. In G, an s-t *flow* $\mathbf{f} : E \to \mathbb{R}$ (Definition 7.3) is *feasible* if:

$$|f_e| \leq c_e, \quad \forall e \in E \tag{7.26}$$

To solve the MAXIMUM s-t FLOW problem, we need to find a feasible s-t flow in G with maximum flow value (Definition 7.4).

A cluster $S \subset V$ defines an s-t *cut* if $s \in S$ and $t \in \bar{S}$. The *capacity* $c(S)$ of the cut is:

$$c(S) := \sum_{u \in S, v \in \bar{S}} c_{(u,v)}.$$

To solve the MINIMUM s-t CUT problem, we need to find an s-t cut of G with minimum capacity. The celebrated *Max Flow-Min Cut Theorem* [135, 124] states that:

$$\max_{\mathbf{f} \text{ is a feasible } s\text{-}t \text{ flow}} \text{Value}(\mathbf{f}) = \min_{S \text{ is an } s\text{-}t \text{ cut}} c(S) \tag{7.27}$$

for any capacitated network. Moreover, one can solve the MAXIMUM s-t FLOW and the MINIMUM s-t CUT problem in polynomial time using linear programming.

Both the Max Flow-Min Cut Theorem and its linear programming algorithm hold more generally for directed networks $G = (V, E, \mathbf{C})$, in which $c_{(u,v)}$ may be different from $c_{(v,u)}$ for some $u, v \in E$. However, the MAXIMUM s-t FLOW and MINIMUM s-t CUT appeared to be challenging problems for scalable computing. Even for the basic version of computing or $(1-\epsilon)$-approximating the maximum flow in sparse undirected, unit-capacity graphs, the asymptotically fastest known algorithm, developed in 1975 by Even and Tarjan [126], takes time $O(n^{3/2})$. More generally, in combination with the Benczur-Karger cut-sparsifiers [49], the Even-Tarjan flow algorithm implies an $O(m + n^{3/2})$ time $(1+\epsilon)$-approximation algorithm for MINIMUM s-t CUT, where O hides a polynomial factor of $\frac{1}{\epsilon}$.

In a landmark application of the Laplacian paradigm, Christiano *et al.* [91] broke the $O(n^{3/2})$ complexity barrier described above for approximately computing maximum s-t flows and minimum s-t cuts in undirected capacitated networks. Christiano *et al.*'s algorithm is algebraic and conceptually simple. It is well-known that to approximately solve MAXIMUM s-t FLOW, it is sufficient to focus on the problem of finding a flow of a given value F or determine that it is infeasible to flow this much: Once we can solve this problem, we can then apply the algorithm with the standard binary search to find a feasible approximate maximum flow.

Christiano *et al.*'s algorithm is based on the following physical intuition: One may view the capacitated network as a network of resistors (proportional to the inverse of the capacity). Then, for any given value F, by scaling up the electrical flow[2] from s to t by a factor F, we obtain an initial s-t flow of value F. In the very lucky event that the electrical flow of value F is feasible, we obtain an s-t flow of value F.

Of course, one cannot count on this to always happen. In the typical case when this electrical flow is infeasible, intuitively we can adapt the resistances of edges to make the electrical flow more desirable: We should increase the resistances of overflowing edges to penalize them and decrease the resistances of underflowing edges. Combined with the method of *multiplicative weights update* for mathematical programming and regret minimization [30], Christiano *et al.* [91] converted this intuition into an algorithm.

Formally, for an s-t-flow $\mathbf{f}$, let $\text{congestion}_{\mathbf{f}}(e) := |f_e|/c_e$. For an approximation parameter ϵ, the flow algorithm of Christiano *et al.* [91] uses an additional parameter $\rho > 0$ and sets $T = \widetilde{O}_n(1) \cdot \frac{\rho}{\epsilon^2}$. It computes a sequence of electrical flows $\mathbf{f}^{(0)}, ..., \mathbf{f}^{(T)}$ together with a sequence of *latent parameters*:

$$\mathbf{W}^{(0)}, ..., \mathbf{W}^{(T)}$$

to assist the setting of edge resistances for each iteration. Initially, $\mathbf{W}^{(0)}$ sets $w_e^{(0)} = 1$ for $e \in E$. These parameters then evolve according to the following multiplicative update rule: For each $e \in E$:

$$w_e^{(t)} := w_e^{(t-1)} \left(1 + \frac{\epsilon}{\rho} \cdot \text{congestion}_{\mathbf{f}^{(t)}}(e)\right)$$

In other words, higher congestion induces higher increase of weight, which in turn introduces higher increase of resistance: The algorithm uses the following resistance setting:

$$r_e^{(t)} \leftarrow \frac{w_e^{(t)}}{c_e^2} + \left(\frac{\epsilon \sum_e w_e^{(t)}}{3mc_e^2}\right)$$

where the second part of the sum is a regularization term. Flow $\mathbf{f}^{(t)}$ is then the s-t electrical flow of value F in the network of resistors

[2] See Definition 7.5.

with resistances $\mathbf{r}^{(t)}$, which can be scalably computed by SDD primitive. Christiano *et al.* [91] proved that by setting $\rho = \widetilde{O}(m^{1/3}/\epsilon)$ and removing edges whose congestion is more than ρ, the average flow:

$$\bar{\mathbf{f}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{f}^{(t)}$$

is a good approximation of a feasible s-t flow of value F if such feasible flow exists. The setting of ρ implies $T = \widetilde{O}(m^{1/3})$. The algorithm then applies binary search to approximate the maximum flow value. Combined with the sparsification technique of Benczur-Karger [49], Christiano *et al.* obtained an approximate maximum flow algorithm with complexity $\widetilde{O}(m + n^{4/3})$.

Although this application of the Laplacian paradigm fell short of obtaining a scalable maximum-flow/minimum-cut approximation algorithm for undirected networks, it pointed out a new numerical approach to this optimization problem. Unlike the traditional max-flow algorithms which improve the flow by "locally defined" augmenting paths, Christiano *et al.*'s approach updates the flow with a more "globally defined" electrical flow.

This new approach inspired two breakthrough algorithms two years later, one by Sherman [303], and one by Kelner, Lee, Orecchia, and Sidford [197] for approximating max flows and min cuts. Both algorithms are almost scalable, i.e., the running time of these algorithms is $\widetilde{O}(mn^{o(1)})$. Both groups replaced electrical flows with oblivious routing [281], which provides a more flexible scheme with better approximation to flow demands than electrical flows.

An *oblivious routing algorithm* determines a routing scheme between pairs of vertices in a given network before the actual pair-wise multi-commodity demands are known. However, the performance of an oblivious routing scheme is measured against the best off-line routing schemes, which know both the network and the demands.

Two powerful techniques to achieve competitive congestion are randomization and tree-based decomposition. The former balances the worst-case congestion by routing demands over randomly chosen paths [341], while the latter effectively spreads pairwise flows over predetermined routes given by hierarchical decomposition [281].

In a remarkable work using the latter approach, Räcke [281] demonstrate that an oblivious routing scheme with $\widetilde{O}_n(1)$-competitive ratio exists for undirected networks. His initial oblivious-routing algorithm is not scalable [35]. Both flow algorithms of [303] and [197] use an almost scalable[3] tradeoff between running time and competitive ratio of the oblivious routing scheme devised by Mądry [242].

These new flow algorithms have in turn been used by Räcke, Shah, and Täubig [282] to improve hierarchical tree decomposition, which recursively partitions G until each set is a singleton vertex. This recursive process defines a tree T, in which internal nodes represent the induced subsets and leaves are singletons in V. The network $G = (V, E, \mathbf{C})$ assigns natural capacities for tree edges of T. Each tree edge (p, c) is associated with two subsets V_p and V_c induced by the recursive partition, and $V_c \subset V_p$ assuming p is the parent of c. The capacity of (p, c) in T is assigned to $c(V_c)$, the total capacity of edges between V_c and $\bar{V_c}$. This assignment guarantees that every feasible pair-wise multi-commodity flow demand in G can be feasibly routed in T: We simply route the demand between each pair along their unique path in T.

To show that this hierarchical decomposition tree can be used to define an oblivious routing scheme with competitive (approximation) ratio $\widetilde{O}_n(1)$, Räcke, Shah, and Täubig [282] prove that the converse is approximately true: Any feasible multi-commodity flow demand among leaf-nodes of T can be routed in G with $\widetilde{O}_n(1)$ congestion.

Subsequently, using *ultra-sparsifiers*[4] — a key concept in the original Spielman-Teng SDD solver — Peng [275] untangled the "chicken-egg" dilemma between maximum flow computation and oblivious routing construction. Peng obtained the first scalable algorithm for approximating maximum s-t flows and minimum s-t cuts. His algorithm also leads to the first scalable cut-approximating hierarchical tree decompo-

[3] i.e., the running time is $mn^{o(1)}$ and competitive ratio is $n^{o(1)}$, where $n^{o(1)} \gg \widetilde{O}_n(1)$.

[4] A (k, h)-ultra-sparsifier [319] of a network $G = (V, E, \mathbf{W})$ is a pair (T, U) where T is a spanning tree of G and $U \subset E$ such that (1) $|U| = hm/k$ and (2) $\mathbf{L}_{T \cup U} \preceq \mathbf{L}_{\mathbf{W}} \preceq k \cdot \mathbf{L}_{T \cup U}$, where $\mathbf{L}_{T \cup U}$ is the Laplacian of the matrix of the spanning subgraph of G defined by $T \cup U$ and their weights. It is shown in [319] that for any k, a $(k, \widetilde{O}_n(1))$-ultra sparsifier can be scalably constructed.

sition for oblivious routing. We welcome these basic graph algorithms to the Laplacian family of scalable primitives.

Efficient Algorithmic Paradigms and Their Scalability

In the field of computing, several powerful algorithmic paradigms have been developed and applied to various fundamental problems. These include theory-leaning paradigms [7, 103, 209]:

- Divide and Conquer
- Greedy and Local Search
- Dynamic Programming
- Mathematical Programming and Approximation
- Sampling and Randomization

They also include practice-leaning paradigms:

- Branch-and-Bound
- Multilevel Methods
- Markov Chain Monte Carlo Methods
- Simulated Annealing
- Genetic Algorithms

On the one hand, successful applications of dynamic programming, linear/convex programming, and branch-and-bound have led to polynomial-time algorithms. However, the complexity of many of these algorithms is quadratic or higher.

On the other hand, successful applications greedy and divide-and-conquer have led to scalable [173, 101, 157, 125] or sub-quadratic-time algorithms [235]. But, these algorithms usually only solve problems or instances with special properties.

For example, among the best known such algorithms relevant to SDD primitive is the divide-and-conquer-based nested dissection

[144, 235]. This method solves linear systems defined by symmetric positive-definite matrices. For matrices whose non-zero patterns are planar or planar-like, nested dissection takes $O(n^{1.5})$ time and uses $O(n)$ space. This divide-and-conquer solver significantly improves on the earlier solvers for these linear systems. But, it relies critically on the fact that underlying graphs have balanced separators computable in linear time. These types of separators only exist in special families of graphs, such as planar-like [236, 147, 13] or geometric graphs [253, 252]. However, most real-world graphs, such as the Web and social networks, do not have balanced separators with quality desired by fast nested dissection.

Many practical-leaning paradigms — particularly the multilevel methods [69, 328] — have led to scalable algorithms in practice. Others — including MCMC methods, simulated annealing, and genetic algorithms — also have fast implementation. However, mathematical analyses of their wonderful practical performances are only known for limited problem domains [337]. Understanding the algorithmic behaviors of these methods remains a subject for future research.

Prospects of the Laplacian Paradigm

In addition to the examples presented in earlier sections, the Laplacian paradigm has found additional applications in designing scalable algorithms for image segmentation [250], web-spam detection [20], protein network analysis [348], and elliptic finite-element systems [57]. The scalable SDD primitive has been extended to other matrix functions, including:

- Matrix Exponentials [265]: Given an SDD matrix $\mathbf{M}$ and a vector $\mathbf{v}$, approximately compute:

$$\exp(-\mathbf{M}) \cdot \mathbf{v} = \sum_{k=0}^{\infty} \frac{1}{k!}(-\mathbf{M})^k \cdot \mathbf{v}.$$

- Matrix Roots [86]: Given an SDD matrix $\mathbf{M}$, a constant $-1 \leq p \leq 1$, and a vector $\mathbf{v}$, approximately compute $\mathbf{M}^p \cdot \mathbf{v}$.

It has also been used to design faster algorithms for lossy-generalized-flow computation [105] and random sampling of spanning trees [199].

These progresses in the Laplacian Paradigm signify that the SDD primitive, and the body of scalable tools supporting it, are powerful algorithmic primitives for combinatorial optimization and numerical analysis. Unlike the separator-based divide-and-conquer paradigm, these scalable algorithms make no assumption about graph structures. Their complexity depends only on the number of nodes and edges in the underlying graph. Moreover, the complexity of the SDD solver is logarithmic in the accuracy parameter. Before these scalable SDD solvers, no linear solver with complexity better than $O(m^{1.5})$ are known for general graph structures. All other fast solvers are domain specific. For example, multigrid methods [72] have been proven effective for solving linear systems derived from well-shaped discretizations that arise in elliptic partial differential equations. Divide-and-conquer methods have been used to design scalable algorithms for solving linear systems defined by hierarchically semi-separable matrices [161, 352, 152, 123, 16]. Thus, the Laplacian Paradigm has the potential to lead to more significant advances in graph algorithms.

Limitations of the Laplacian Paradigm

On the other hand, while the Laplacian paradigm could open a new page for scalable algorithm design, the current techniques are largely limited to diagonally-dominant matrices and undirected networks. Extending these scalable algorithms to directed networks and beyond linear systems remains challenging despite the recent breakthroughs in the max-flow-min-cut problem. It is an outstanding research question to determine whether scalable algorithms exist for:

- Linear systems with symmetric positive definite matrices
- Linear programming
- Sampling from Gaussian Markov random fields
- Maximum-flow computation in directed networks

8

Remarks and Discussions

In this chapter, we will conclude by discussing a few basic topics, particularly regarding the *constant challenge* in algorithm design, optimization, and data analysis:

> *Which objective functions should algorithms optimize? What solution concepts should algorithms use to capture input data?*

We will revisit the notion of centrality, clusterability, and the basic conceptual question of what constitutes a community in a social network. We will also take this opportunity to discuss network models beyond graphs, as well as algorithmic performance frameworks beyond worst-case and average-case analyses.

8.1 Beyond Graph-Based Network Models

In this section, we will follow up the brief discussion of Section 2.3. Studying network models beyond graphs helps to broaden our understanding of multifaceted networks data. Some of these models, such as preference networks, are inspired by social-choice theory [33], while others, such as incentive networks, are based on game-theoretical and

economical principles [259, 258, 302]. In the real world, these network models are used to capture different facets of network data.

- Weighted graphs are widely used for modeling social networks, the Internet, and the Web.
- Preference networks are used for expressing preferences in college admissions and medical-residency assignments [140, 289, 160], and for specifying routing preferences in the Border Gateway Protocol between autonomous systems in the Internet [284, 77].
- Incentive networks are used in cooperative game theory [302] mechanism design [222], and coalition theory in political science [67, 290].

In this section, we will examine some basic properties of preference networks and incentive networks in order to address the conceptual questions of collective ranking and community identification. The connections of these basic network problems with game theory, economics, and social-choice theory [33] point to both challenges and exciting opportunities for understanding fundamental questions in network sciences. We believe that a broader view of networks beyond the graph model will enable us to comprehensively examine different facets of network data.

8.1.1 Network Centrality

We first consider the most basic *dimensional-reduction problem* of network data.

Social-Choice View of Centrality and Collective Ranking

While PageRank is a popular measure for the significances of nodes in graph-based networks, social-choice theory offers a natural framework for defining the *collective ranking* of nodes in preference networks. Traditionally, social-choice theory [33] considers a set of *candidates* C and a set of *voters* V, in which each voter $v \in V$ casts a vote expressing its ranking $\boldsymbol{\pi}_v \in L(C)$ of the candidates. A voting scheme then applies

an *aggregation function* to summarize individual rankings into one collective ranking. Voting schemes may allow ties in both individual and collective preferences. For such elections, one can mathematically use *ordered partitions*[1] of C to express rankings with ties.

Definition 8.1 (Social-Choice Aggregation)**.** Let $\overline{L(C)}$ denote the set of all *ordered partitions* of C. A *social-choice aggregation scheme* (or *voting scheme*) is an aggregation function $F : \overline{L(C)}^* \to \overline{L(C)}$.

Given a non-empty finite set of voters V and their preference profile $\Pi_V = \{\boldsymbol{\pi}_v \,:\, v \in V\} \in \overline{L(C)}^{|V|}$, the image $F(\Pi_V)$ is then called the *aggregated preference ranking* of C.

Recall that a preference network $A = (V, \Pi)$ is defined by $|V|$ *individual preference profiles*, one for each member $u \in V$, ranking all nodes in the network. We can also extend the strict notion of preferences by assuming $\Pi \in \overline{L(C)}^{|V|}$. We can view a preference network as a *self-voting* system, that is, the candidates are themselves voters (i.e., $V = C$). Thus, social-choice theory immediately provides the following approach to formulating *collective rankings* in preference networks:

> *Construct a desirable function that aggregates the set of individual preferences* Π *into a single ranking of nodes in the network.*

On the one hand, while many aggregation-based collective ranking schemes for preference networks may be reasonable, one has to be extremely careful in defining the meaning of *desirable* collective rankings. Arrow's celebrated impossibility theorem[2] and the subsequent work in

[1] An *ordered partition* of a set C is a total order of a partition of the set C.

[2] When there are at least three candidates, there exists no rank-order voting system that satisfies three basic fairness criteria: *unanimity* — if all voters agree, then the aggregated outcome must be the same as the voters' choice; *non-dictatorship* — no single voter has the power to always determine the final outcome; and *independence of irrelevant alternatives* (IIA) — if in two elections every voter's preference for two particular candidates remains unchanged, then the ranking of these two candidates must be the same in the outcomes of the two elections.

social choice theory [33] illustrate the fundamental challenge in collective ranking. When projected onto graph-based network models, these impossibility theorems indeed raise questions regarding the potential limitations of PageRank or any other ranking methods [267, 15].

On the other hand, social-choice theory offers many concrete preference aggregation functions. Some of them are already used in practice. For example, the Baseball Writers' Association of America uses a variant of the well-known Borda count [356] (a unanimous voting method with no dictator) as the aggregation scheme for the selection of MVP and Cy Young award winners. The Borda count and its variants are part of a *weighted voting scheme*:[3] For a set C of n candidates, it uses a vector $\mathbf{w} = (w_1, ..., w_n)$. When aggregating preference profiles Π_V, it computes a numerical *score* for each candidate $c \in C$ as:

$$\phi_{\mathbf{w}}(c) = \sum_{u \in V} w_{\pi_v(c)}.$$

It then ranks the candidates by sorting them according to their scores. Aggregation functions can be less numerical and more topological [66]. Consider the following one, which will be referred to as a *harmonious aggregation*: Let $G_{\Pi_V} = (C, E_{\Pi_V})$ be the following directed graph: For $i, j \in C$, $(i, j) \in E_{\Pi_V}$ if and only if at least half of V prefer i to j. When $|V|$ is an odd number, G_{Π_V} is a *tournament graph.* When $|V|$ is an even number, E_{Π_V} contains both (i, j) and (j, i) iff i and j are both preferred by half of Π_V. G_{Π_V} is *total* since $\forall i, j \in V$, either $(i, j) \in E_{\Pi_V}$ or $(j, i) \in E_{\Pi_V}$. Thus, the graph $\hat{G}_{\Pi_V}$ obtained from G_{Π_V} by contracting each strongly connected component into a single vertex is an *acyclic* tournament graph. So, $\hat{G}_{\Pi_V}$ has exactly one Hamiltonian path that completely orders its vertices. Let $(C_1, ..., C_t)$ be the strongly connected components of G_{Π_V} along this Hamiltonian path. The harmonious aggregation is then this ordered partition $(C_1, ..., C_t)$ of C.

Cooperative Game-Theoretical View of Centrality

We will now examine network centrality through the lens of cooperative game theory [302]. Compared to graphs — as we illustrated above —

[3]The Borda count uses $\mathbf{w} = (n, n-1, ..., 1)$ as the weighting vector for ranking n candidates.

the preference network model further highlights the connection between *social-choice aggregation* and *network centrality*. In the same fashion — as we shall show below — the incentive-network model accentuates the connection between the following *game-theoretical concept* and *network centrality*.

Mathematically, a *cooperative game* over n-players $V = [n]$ is specified by a *characteristic function* $\boldsymbol{\tau} : 2^V \to \mathbb{R}$, where for any coalition $S \subseteq V$, $\boldsymbol{\tau}[S]$ defines the *cooperative utility* of S [302]. In 1953, Shapley defined the following payoff-distribution scheme to address the basic question of "how important is each player to overall cooperation?"

Definition 8.2 (Shapley Value)**.** Suppose $\boldsymbol{\tau}$ is the characteristic function of a cooperative game over $V = [n]$. Recall that $L(V)$ denotes the set of all permutations of V. Let $S_{\boldsymbol{\pi},v}$ denotes the set of players preceding v in a permutation $\boldsymbol{\pi} \in L(V)$. Then, the *Shapley value* $\boldsymbol{\phi}_{\boldsymbol{\tau}}^{Shapley}[v]$ of a player $v \in V$ in the game is:

$$\boldsymbol{\phi}_{\boldsymbol{\tau}}^{Shapley}[v] = \mathrm{E}_{\boldsymbol{\pi} \sim L(V)} \left[\boldsymbol{\tau}[S_{\boldsymbol{\pi},v} \cup \{v\}] - \boldsymbol{\tau}[S_{\boldsymbol{\pi},v}] \right].$$

The Shapley value $\boldsymbol{\phi}_{\boldsymbol{\tau}}^{Shapley}[v]$ of player $v \in V$ in the game is the *expected marginal contribution* of v over the set preceding v in a random permutation of the players. Shapley further proved that his solution is the unique payoff-distribution vector that is (1) efficient (i.e., summing up to $\boldsymbol{\tau}[V]$), (2) invariant under permutation, (3) additive, and (4) assigns zero payoffs to players with persistent zero marginal utility. The Shapley value is widely considered to be the *fairest* measure of a player's power index in a cooperative game.

Each symmetric incentive network $U = (V, \boldsymbol{u})$ can be viewed as a natural cooperative game: For all non-empty $S \subseteq V$, the utility of S is $\boldsymbol{\tau}_U[S] = u_s(S \backslash \{s\})$, for any $s \in S$. More generally, we can "symmetrize" any incentive network $U = (V, \boldsymbol{u})$ by defining a cooperative game with $\boldsymbol{\tau}_U[\emptyset] = 0$, and:

$$\boldsymbol{\tau}_U[S] = \frac{1}{|S|} \cdot \sum_s u_s(S \setminus \{s\}), \ \forall \text{ non-empty } S \subseteq V.$$

Thus, in the incentive network model, the game-theoretical Shapley value provides a natural solution concept for network centrality. For

any incentive network $U = (V, \boldsymbol{u})$, we call the Shapley value — $\boldsymbol{\phi}^{Shapley}_{\boldsymbol{\tau}_U}$ — the *Shapley centrality* of U. Then, the following question becomes highly relevant to studying graph-based network models through the lens of cooperative game theory:

What are natural cooperative network games?

Interplay between Influence Processes and Networks

Chen and Teng [84][4] considered the following fundamental question in network science:

Given a social network, what is the impact of influence models on network centrality?

Recall that a social-influence instance $\mathcal{I}$ can be specified by a directed graph $G = (V, E)$ and its interaction with an influence model $\mathcal{D}$ [201, 114, 286]:

- G defines the graph structure of the social network.
- The social-influence model $\mathcal{D}$ is usually defined by a stochastic process that characterizes how nodes in each *seed set* $S \subseteq V$ *collectively influence* other nodes using the edge structures in G [201]. Two commonly considered influence models are *independent cascade* (IC) and *linear threshold* models [201].

Mathematically, the influence process $\mathcal{D}$ and the network structure G together define a probability distribution:

$$\boldsymbol{P}_{G,\mathcal{D}} : 2^V \times 2^V \to [0, 1].$$

For each $T \supset S$, $\boldsymbol{P}_{G,\mathcal{D}}[S, T]$ specifies the probability that T is the final activated set when S cascades its influence through the network G. Thus, $\boldsymbol{P}_{G,\mathcal{D}}$ can be viewed as the *ultimate interplay* between the *static*

[4]This paper appeared during the final revision of this article. We add this short summary of its results because of the direct relevance to our discussion. For detailed presentations, we refer interested readers to the arXiv version of this paper [84].

network structure G and *dynamic* influence process $\mathcal{D}$. An important quality measure of S in this process is S's *influence spread*:

$$\sigma_{G,\mathcal{D}}(S) = \sum_{T \supset S} |T| \cdot \boldsymbol{P}_{G,\mathcal{D}}[S, T] \tag{8.1}$$

See [201, 114, 286] for examples and properties of social-influence models as well as motivations from viral marketing and social sciences.

Clearly, formulations based solely on static network structures — such as degrees [260], betweenness [138, 137, 63], and local sphere-of-influence [46, 58, 59, 127, 294, 249, 1] — do not sufficiently capture the impact of social-influence models on centrality. Numerous centrality measures in the literature have gone beyond static graph structures by incorporating various underlying mathematical processes over graphs. For example, the popular PageRank centrality assumes a random-walk Markov process is taking place on the graph, which induces interactions among nodes. Its heat-kernel extensions[5] [93, 146] also make similar assumptions. However, the random-walk process and heat-diffusion are very different from the typical social-influence processes [201]. Other centrality measures are formulated based on different processes such as percolations [277], synchronizations [117] network flows [62], and sociometric analysis [191]. Likewise, these processes do not capture social-influence models as naturally formulated in [201].

The work of [84] attempts to directly capture centrality in social-network influence processes. It first uses the following natural cooperative game associated with network influence to capture both the static network structure and the dynamic influence model.

Definition 8.3 (Cooperative Games of Social Influence)**.** Each social-influence instance $\mathcal{I} = (V, E, \mathcal{D})$ defines a cooperative game over the nodes $V = [n]$ of netowrk $G = (V, E)$, whose characteristic utility function from 2^V to $\mathbb{R}$ is the influence spread function $\sigma_{G,\mathcal{D}}$.

In [84], the Shapley value of this cooperative game is then used as the centrality measure of the social-influence instance.

[5] See Section 4.6.

Definition 8.4 (Shapley Centrality)**.** For any social-influence instance $\mathcal{I}$ defined by an influence model $\mathcal{D}$ over a social network $G = (V, E)$, the *Shapley centrality* $\mathbf{sc}_{\mathcal{I}}$ of $\mathcal{I}$ is the Shapley value $\phi^{Shapley}(\sigma_{\mathcal{I}})$ of the cooperative social-influence game given by $\sigma_{G,\mathcal{D}}$.

Two *complementary* results in [84] characterize this game-theoretical centrality formulation for network influences.

Algorithmically, the paper gives a provably-good scalable algorithm for approximating the Shapley values of cooperative games associated with a large family of social-influence instances as defined in [201]. This scalable algorithm expands upon techniques from the recent algorithmic breakthroughs in influence maximization [64, 325]. Like the scheme for PageRank approximation (Chapter 3), this algorithm builds robust estimators for social-influence Shapley values by sampling *reversed influence processes*.

Mathematically, it gives an axiomatic characterization of social-influence Shapley centrality. It presents five natural centrality axioms for incorporating the impact of social-influence processes on network centrality, and captures the essence of Shapley centrality by proving that it is the unique solution to these axioms. This part of the work was inspired by social choice theory [33], and particularly by the work of Palacios-Huerta and Volij on measures of intellectual influence [267], and Altman and Tennenholtz on PageRank [15].

The work of [84] represents a step in a long line of research in applying the following game-theoretical approach to study network phenomena and network centrality:

1. Formulating a network game based on network data.

2. Applying game theory to formulate network centrality.

Grofman and Owen [159] were among the first to use this approach. They focused on voting games and centrality based on Penrose-Banzhaf power index [276, 41]. Gómez *et al.* [156] then applied this approach using cooperative network games and their Shapley values [302]. For

example, the method of Gómez *et al.* starts with an n-person cooperative game C, and then uses it to define another "projected" cooperative game C_G based on network G. It then uses the difference of the Shapley values of these two games as the network's centrality measure. In their view, the network structure of G limits the effectiveness of cooperation among nodes. The work of Gómez *et al.* [156] was further extended in [322, 257, 249, 1, 323].

8.1.2 Network Communities

We now return to the basic question:

What constitutes a community in a social network?

Community identification in social and information networks is an *inverse* problem: A community is formed by a group of individuals. However, information/social networks are usually given by data specifying pairwise interactions among its members, or the individuals' affinities or preferences towards other members in the network.

Thus, the basic question for community identification is to understand:

How do individual preferences (affinities/connectivities) result in group preferences or community coherence?

The basic task for community identification is to develop a consistent scheme that extends individual interactions/preferences to *community characterization*. Community identification has been recognized as a major challenging problem in network sciences. The sparsity of practical social networks further adds to the ambiguity and difficulty of this basic question.

A Social-Choice View of Network Communities

We will first focus on community characterization in preference networks. Like its role for modeling the stable marriage problem arising in the real-world residency assignments [140, 289, 160], preference networks offer simple yet rich frameworks for focusing on the relationship between individual preferences and community characterization

[39, 66]. In addition, classical results regarding preferences can be used to jump-start the community formulation.

For example, the collective ranking framework of social-choice theory [33] immediately leads to some basic community rules for preference networks [66, 39]: Suppose $F : L(V)^* \to \overline{L(V)}$ is a preference aggregation function. For a subset S in a preference network $A = (V, \Pi)$, consider the aggregated preference $F(\Pi_S)$ of S. Intuitively, if S — as a group — does poorly in its own collective ranking $F(\Pi_S)$, then S is not a strong candidate to be a community. Taking this view to the limit, Balcan *et al.* [39, 66] introduced the following notion of *self-certification*.

Definition 8.5 (Self-Certification)**.** $S \subseteq V$ is said to be a *self-certified* (or self-determined) group in a preference network $A = (V, \Pi)$ with respect to a social-choice aggregation function F, if its aggregated preference F_{Π_S} places S as the most preferred $|S|$ elements in V. In other words, $u \succ_{F(\Pi_S)} v$, $\forall u \in S, v \in V - S$.

Let's start with two examples. The first one uses the weighted aggregation, $\phi_S^{\Pi}(i) = |\{s \ : \ (s \in S) \ \& \ (\pi_s(i) \in [|S|])\}|$, for $S \subseteq V$ and $i \in V$. In other words, $\phi_S^{\Pi}(i)$ denotes the number of votes that i receives when members of S votes for their $|S|$ most preferred members in V. Following Definition 8.5, we say S is a *democratic community* of A, if everyone in S receives more votes from S than everyone outside S, i.e., $\min_{u \in S} \phi_S^{\Pi}(u) - \max_{v \notin S} \phi_S^{\Pi}(v) > 0$.

The second rule uses *harmonious aggregation*: In a preference network $A = (V, \Pi)$, we say $S \subseteq V$ is a *harmonious community*, if $\forall u \in S$, $\forall v \in V - S$, the majority of $\{\pi_s \ : \ s \in S\}$ prefer u over v [66].

We can also apply these community rules to graph-based networks. For example, suppose $G = (V, E, \mathbf{W})$ is a weighted (directed) network. Let $\mathbf{PPR}_{\mathbf{W},\alpha}$ denote the personalized PageRank matrix (Definition 3.5). For each $u \in V$, let $\boldsymbol{\pi}_u$ be the ranking V obtained by sorting the personalized PageRank vector $\mathbf{p}_u = \mathbf{PPR}_{\mathbf{W},\alpha}[u, :]$ in descending order. We view $\mathbf{p}_u$ as u's preference of V and reduce the analysis of weighted network G to that of the preference network $A = (V, (\boldsymbol{\pi}_u)_{u \in V})$: We can

use communities in A, such as its harmonious communities, democratic communities, or self-certified communities based on other aggregation functions, as candidate communities for G.

Stable Communities: Structures and Local Algorithms

We can also extend Definition 8.5 to capture how *stable* these social-choice-based communities are. In real-world social networks, some communities are more durable than others when members' preferences evolve over time. For example, some teams of collaborators stay together longer than others. Balcan *et al.* [39] considered the following parameterization of self-certified communities: For $0 < \gamma < 1$, a subset $S \subseteq V$ is a γ-*stable* democratic community in a preference network $A = (V, \Pi)$ if:

$$\min_{u \in S} \phi_S^{\Pi}(u) - \max_{v \notin S} \phi_S^{\Pi}(v) > \gamma \cdot |S|.$$

Similarly, S is a γ-*stable* harmonious community in A if for all $u \in S$ and $v \notin S$, at least an $(\frac{1}{2} + \gamma)$-fraction of $\{\pi_s \ : \ s \in S\}$ prefer u over v [66]. Both conditions imply that S remains a self-certified community, even if $\gamma/2$-fraction of its members change their preferences.

For graph models, Mishra, Schreiber, Stanton, and Tarjan [254] proposed the following notion of stable communities: For $0 \leq \beta < \alpha < 1$, a subset $S \subset V$ is an (α, β)-community of a graph $G = (V, E)$ if every $u \in S$ has at least $\alpha|S|$ neighbors in S, and every $v \notin S$ has no more than $\beta|S|$ neighbors in S. Note that the notion of γ-stable democratic communities can be viewed as the extension of (α, β)-communities to preference networks: Suppose S is a γ-stable community of $A = (V, \Pi)$. Let $\alpha = \frac{1}{|S|} \cdot \min_{u \in S} \phi_S^{\Pi}(u)$ and $\beta = \frac{1}{|S|} \cdot \max_{v \notin S} \phi_S^{\Pi}(v)$. Then, (i) $\alpha - \beta > \gamma$, (ii) $\phi_S^{\Pi}(u) \geq \alpha \cdot |S|$, $\forall u \in S$, and (iii) $\phi_S^{\Pi}(v) < \beta \cdot |S|$, $\forall v \notin S$.

Balcan *et al.* [39] proved the following structural theorem.

Theorem 8.6 (Polynomiality of Stable Communities)**.** For any constant $0 < \gamma < 1$, every preference network $A = (V, \Pi)$ of n members has at most $n^{O(\log(1/\gamma)/\gamma)}$ γ-stable democratic communities. Moreover, they can be enumerated in polynomial time.

Surprisingly, Theorem 8.6 hold true for multifaceted preference networks, in which each member has a constant number of preferences [39]. These results provide a structural contrast between stable communities in (multifaceted) preference networks and the (α, β)-stable communities defined by Mishra, *et al* [254]. Some graphs have a super-polynomial number of (α, β)-communities. In addition, the problem of identifying a single (α, β)-community is computationally as hard as identifying planted clique in random graphs [39]. The contrasting structural difference between preference networks and graphs highlights the inherent data ambiguity from the perspective of community identification in graph-based networks.

The structural property also leads to a scalable local algorithm, for identifying stable communities:

Theorem 8.7 (Scalable Identification of Stable Communities)**.** Suppose $A = (V, \Pi)$ is a preference network, and $\alpha > \frac{1}{2}, \gamma < \frac{1}{2}$ are two constants. Suppose S is a γ-stable community in A, such that $\forall u \in S$, $\phi_S^{\Pi}(u) \geq \alpha \cdot |S|$. Then, S can be identified locally in time $O(t \log t)$ with probability at least $(2\alpha - 1)$, when given $t = |S|$ and a node chosen uniformly at random from S.

Proof. (Sketch) Let $\eta := 2\alpha - 1 > 0$. Given $v \in V$ and a target community size t, the goal of the local algorithm is to find a community S of size t containing v. For $v \in V$, let $R(v)$ be a uniformly random element from $\boldsymbol{\pi}_v[1:t]$. The key observation is the following: For any γ-stable community S in A satisfying $\forall u \in S$, $\phi_S^{\Pi}(u) \geq \alpha \cdot |S|$, there is a subset $T \subseteq S$ such that $|T| \geq \eta t$ and for each pair $v \in T$ and $u \in S$:

$$\Pr[R(R(v)) = u] \geq \frac{\eta}{2t}.$$

To prove this, for $v \in S$, let $\text{fans}_S(v) := \boldsymbol{\pi}_v[1:t] \cap S$ and $\text{friends}_S(v) := \{u \in S : \ v \in \boldsymbol{\pi}_u[1:t]\}$. Then, for all $v \in S$, $|\text{friends}_S(v)| \geq \alpha t$. Thus:

$$\sum_{v \in S} |\text{fans}_S(v)| = \sum_{v \in S} |\text{friends}_S(v)| \geq \alpha t^2.$$

Let $T = \{v \in S : |\text{fans}_S(v)| \geq \frac{t}{2}\}$. By Markov's inequality, $\frac{|T|}{|S|} \geq \eta$. For any $v \in T$ and any $u \in S$, we have:

$$|\text{fans}_S(v) \cap \text{friends}_S(u)| \geq |\text{fans}_S(v)| + |\text{friends}_S(u)| - t \geq \frac{1}{2}\eta \cdot t.$$

Thus, for any $u \in S$:

$$\Pr\left[R(R(v)) = u\right] \geq \Pr\left[R(v) \in \text{fans}_S(v) \cap \text{friends}_S(u)\right] \cdot \frac{1}{t} \geq \frac{\eta}{2t}.$$

T are *good seeds* for S because S can be identified from any $v \in T$ locally by a simple sampling and purification process. See [39] for details. □

Game-Theoretical Community Rules in Incentive Networks

The stable marriage problem [140, 289, 160] can be viewed as a study of two-person community structures in preference networks. The solution concept — *stable matching*[6] — are more game-theoretically motivated than self-certification communities: A stable matching does not contain two members that are unmatched to each other, but both prefer each other to their matched partners.

This notion of stability is also more globally defined than self-certification. For example, the latter satisfies a local property called the *Independence of Outside Opinions*, stating that the preferences of outsiders cannot influence whether or not a subset is a community. This property is clearly not satisfied by the former. The notion of stability has been extended from matching to coalition formation [166, 67, 48, 333, 290, 283]. Like clustering, a *coalition structure* of V is a *partition* of V. For example, for $k \geq 1$, a coalition $S \subset V$ of k members is said to be *stable* if no member in S would prefer[7] to be in a different coalition of size k.

Recall that an incentive network is a pair $U = (V, \boldsymbol{u})$, where for each $s \in V$, $u_s : 2^{V \setminus \{s\}} \to \mathbb{R}$ defines s's participation utility with subsets of $V \setminus \{s\}$. Mathematically, an incentive network is defined by

[6] A matching is a partition of V into $\frac{|V|}{2}$ groups of pairs.

[7] When $k > 2$, the notion that "a member prefers one k-member coalition over another k-member coalition" is not uniquely defined for a preference network $A = (V, \Pi)$, thus requires careful modeling.

$|V| \cdot 2^{|V|-1}$ utility values. In practice, these utility functions are succinctly defined. For example, the popular modularity-based clustering method in fact works with the following family of incentive networks: For each weighted network $G = (V, E, \mathbf{W})$, let $\boldsymbol{M} = \mathbf{W} - \boldsymbol{d}\boldsymbol{d}^T/\mathrm{vol}_W(V)$ be the modularity matrix of G. Then, $U^{\text{modularity}} = (V, \mathbf{u}^{\text{modularity}})$ is an incentive network derived from $\boldsymbol{M}$ based on the following simple scheme: For $s \in V$ and $T \in V \setminus \{s\}$, $u_s^{\text{modularity}}(T) = \sum_{t \in T} \boldsymbol{M}[s,t]$. The social utility of any $S \subseteq V$ is then equal to its moduality:

$$\begin{aligned} \text{Social-Utility}(S) &= \sum_{s \in S} u_s^{\text{modularity}}(S \setminus \{s\}) \\ &= \sum_{s \in S} \sum_{t \in S} \boldsymbol{M}[s,t] = \text{modularity}(S). \end{aligned}$$

This incentive-network formulation scheme can be applied to other matrices, for example:

- personalized PageRank matrix $\mathbf{PPR}_{\mathbf{W},\alpha}$
- random walk matrix $\mathbf{M}_{\mathbf{W}} = \mathbf{W}\mathbf{D}_{\mathbf{W}}^{-1}$
- effective conductance matrix $C_{\mathbf{W}}[i,j] := \mathrm{ER}_{\mathbf{W}}[i,j]^{-1}$

derived from a weighted network $G = (V, E, \mathbf{W})$.

For $U^{\text{modularity}} = (V, \mathbf{u}^{\text{modularity}})$, the social utility of a coalition structure $C = (V_1, ..., V_K)$ is:

$$\text{Social-Utility}(C) = \sum_{i \in K} \text{modularity}(V_i).$$

Thus, clustering with optimal modularity is equivalent to computing a coalition structure with maximum total social utility in incentive network $U^{\text{modularity}}$.

The rich information structure of incentive networks enables us to capture other more complex facets of network data. For example, as we discussed above, the cooperative social-influence game of [84] can be viewed as a symmetric incentive network, $U_{\mathcal{I}} = (V, \sigma_{G,\mathcal{D}})$, derived from a social-influence instance $\mathcal{I} = (G, \mathcal{D})$. Thus, the incentive network — with influence spreads as group utilities — can be used to model the interplay between the influence processes and social networks [84].

Mathematically, the incentive network fully specifies the *utilities* in node-group interactions. Thus, this model permits conceptual studies of clusterability measures and community-identification rules for analyzing networks with explicitly specified individual preference/utility profiles.[8] It allows us to focus on fundamental questions such as:

> *What constitutes a community in an incentive network? How should we measure clusterability in an incentive network?*

Comprehensive mathematical and algorithmic theory remains to be developed for general incentive networks and their communities and clustering structures. For symmetric incentive networks, concepts from cooperative game theory [302], such as the Shapley values and *core*, can be instrumental. For example, harnessing the social-influence properties, a new Shapley-value based clusterability measure is formulated in [83] for the symmetric incentive networks derived from influence instances. This game-theoretical clusterability shares some intrinsic commonality with PageRank contributions [21]. However, instead of capturing PageRank random-walks, this measure reflects the impact of influence models on clusterability in social networks.

An Axiomatic Framework for Community Identification

One can apply physical and statistical intuitions to define clusterability measures. One can also apply game theory, economics, and social-choice theory to derive appealing rules for community identification. However, it remains a fundamental challenge to validate clusterability measures and community identification rules; it also remains difficult to make direct comparisons between different community characterizations.

Axiomatization is a basic mathematical approach to understand these conceptual challenges [33, 207, 66]. It aims at characterizing the maximal set of desirable conditions under which feasible solutions exist. For community identification, the axiomatic framework first defines

[8] One can also define incentive networks using the ordinal framework, which defines u_s not numerically but ordinally by a ranking of all subsets in $V \setminus \{s\}$ in the order of s's preferences in participation.

a set of basic axiomatic properties, such as consistency, fairness, and stability that a desirable community rule should have, and then applies these axioms to analyze community rules holistically [66]. We conclude this section with a brief illustration of the axiomatic study in the recent work of Borgs *et al.* [66], focusing on community rules and community structures in preference networks. Mathematically, a community rule can be defined by a set-theoretical *community function*, which will be the subject of the axiomatic study:

Definition 8.8 (Community Functions)**.** A *community function* $\mathcal{C}$ maps a preference network $A = (V, \Pi)$ to a characteristic function of non-empty subsets of V. In other words, $\mathcal{C}(A) \subseteq 2^V - \{\emptyset\}$. We say a subset $S \subseteq V$ is a *community* according to $\mathcal{C}$ in a preference network $A = (V, \Pi)$ if and only if $S \in \mathcal{C}(A)$.

The axiomatic system of [66] analyzes community rules using a natural set of six axioms. The first two are basic axioms addressing anonymity and monotonicity, inspired by social-choice theory [33]:

- Anonymity: Community functions should be isomorphism-invariant.
- Monotonicity: If S is a community in $A = (V, \Pi)$, then S should remain a community in every preference network $A' = (V, \Pi')$ such that for all $u, s \in S$ and $v \in V$, if $u \succ_{\pi_s} v$ then $u \succ_{\pi'_s} v$.

The next two axioms are basic:

- World Community: For all $A = (V, \Pi)$, V is a community.
- Embedding: If a preference network $A' = (V', \Pi')$ is embedded into a larger one $A = (V, \Pi)$ such that $\pi_i(j) = \pi'_i(j)$ for all $i, j \in V'$, then the community function should preserve the community structure of A', i.e., it should satisfy:

$$\mathcal{C}(A') = \mathcal{C}(A) \cap 2^{V'}.$$

The final two axioms capture the game-theoretical stability inspired by stable marriages and coalition formation [259, 258, 60, 296, 301, 166, 67, 333, 290, 283]. The first stability axiom, Group Stability, states that no subgroup in a community can be replaced by an equal-size group of non-members whom are "unanimously" preferred by the rest of the community members. The second one, Self-Approval, states that a community should have "basic self-respect": No outside group of the same size as S is "unanimously" preferred to S by everyone in S.

Borgs *et al.* [66] proved the following impossibility result, which illustrates the fundamental challenge in directly using the social-choice aggregation to define consistent community rules.

Theorem 8.9 (Impossibility)**.** (1) There is no weighted aggregation for defining a self-certification-based community rule consistent with all six axioms. (2) There is no preference aggregation satisfying Independence of Irrelevant Alternatives for defining a self-certification-based community rule consistent with all six axioms.

Like the impossibility theorems in social choice theory [33], this theorem highlights the fundamental difficulty in defining *consistent* aggregation that summarizes individual preferences into a group preference. Preference aggregation is a form of dimension reduction, in which we can view each individual as a dimension. The aggregation thus can be regarded as the result obtained from reducing multi-dimensional preference data into a "one-dimensional" collective preference. The impossibility results of social choice theory and this theorem all point to the difficulty of reducing the dimensionality of preference data in a consistent manner. In community identification, the difficulty of consistent group summarization of individual preferences is further displayed by the fact that the democratic community rule is not consistent with Axiom Monotonicity. The harmonious community rule enjoys broader consistency — it is only inconsistent with Axiom Group Stability.

The work of [66] is also inspired by Kleinberg's results on data clustering [207]. However, in contrast to Kleinberg's work which focuses on non-overlapping clustering of data in a metric space, the axiomatic

system of [66] characterizes overlapping community structures in preference networks. Thus, this axiomatic system studies community rules, which characterize whether a group is a desirable community, rather than clustering rules, which characterize whether a partition is desirable. The relaxation of the "partition requirement" changes the structure of the solution concept. For example, we can consider the following intuitive community rule. We say a subset $S \subseteq V$ is a *clique* in a preference network $A = (V, \Pi)$ iff $\forall u, s \in S$, $v \notin S$, $u \succ_{\pi_s} v$.

Definition 8.10 (Clique Rule ($\mathcal{C}_{clique}$))**.** Cliques and only cliques are communities in a preference network.

Note that we can apply the standard set-theoretical operators to combine multiple community rules into a new community rule: For two community functions $\mathcal{C}_1$ and $\mathcal{C}_2$, we define the *intersection* and *union*, $\mathcal{C}_1 \cap \mathcal{C}_2$ and $\mathcal{C}_1 \cup \mathcal{C}_2$, as the community functions which, for all preference networks A, respectively:

$$(\mathcal{C}_1 \cap \mathcal{C}_2)(A) := \mathcal{C}_1(A) \cap \mathcal{C}_2(A)$$
$$(\mathcal{C}_1 \cup \mathcal{C}_2)(A) := \mathcal{C}_1(A) \cup \mathcal{C}_2(A).$$

$\mathcal{C}_1 \cap \mathcal{C}_2$ is more *selective* than either $\mathcal{C}_1$ or $\mathcal{C}_2$, and $\mathcal{C}_1 \cup \mathcal{C}_2$ is more *inclusive* than both $\mathcal{C}_1$ and $\mathcal{C}_2$. For example, let $\mathcal{C}_{GS}$ and $\mathcal{C}_{SA}$ be the rules that identify subsets satisfying Axioms Group Stability and Self-Approval, respectively.

Then, the rule

$$\mathcal{C}_{comprehensive} := \mathcal{C}_{GS} \cap \mathcal{C}_{SA}$$

defines the following: $S \in \mathcal{C}_{comprehensive}(A)$ iff S satisfies both Axioms Group Stability and Self-Approval.

The following theorem of [66] further highlights the difference between partition-based clustering in metric spaces [207] and overlapping communities in preference networks. It shows that the family of axiom-conforming community rules is not empty and has a rich algebraic structure.

Theorem 8.11 (Lattice Structures of Community Rules)**.** Let $\mathcal{C}$ denote the set of all community rules satisfying all six axioms. Then, $\mathcal{C}_{clique}, \mathcal{C}_{comprehensive} \in \mathcal{C}$, and the algebraic structure $\mathcal{T} = (\mathcal{C}, \cup, \cap)$ is a bounded lattice under the natural intersection and union operations, with $\mathcal{C}_{clique}$ as the lattice's bottom (most selective) and $\mathcal{C}_{comprehensive}$ as the lattice's top (most inclusive).

The comparative studies of coalition structures in cooperative game theory [160, 166, 67, 48, 333, 290, 283], clustering in metric spaces [207], and community identification in preference networks have raised a fundamental conceptual question:

> *Shall we characterize communities/clusters as coherent subsets on their own, or as parts of a global coalition/clustering structure?*

8.2 Discussions: a Family of Scaling-Invariant Clusterability

A basic task in network analysis is to determine if a clusterability measure, such as conductance or modularity (to be defined below), is effective. For example, Chapters 6 and 7 partially substantiate the *algorithmic importance* of conductances. They show that local clustering with guaranteed conductance leads to the first scalable spectral-sparsification algorithm, SDD linear-systems solver, and effective-resistance query structure.

> *But is conductance a good clusterability measure for community identification in social networks?*

The work of [84, 83] and [146] — discussed in Sections 8.1 and 4.6, respectively — both point out that desirable centrality/clusterability measures should be application specific. Applications may define different interaction models, reflecting the interplay of application-specific processes with the underlying networks.

> *However, selecting and validating a clusterability measure remains an outstanding conceptual challenge.*

In this section, to illustrate the difficulty of this basic question, we will discuss another family of seemingly natural clusterability measures. This family is based on the intuition that a good cluster S should have significantly richer internal connections than its external connections [318, 186], by focusing on a different interpretation of conductance. So far, we have yet to find a dynamic process whose "natural" interaction with the network leads to this family of clusterability measures.

Conductance is widely used in graph partitioning and clustering. and is an intuitively good clusterability measure. However, it still falls short of capturing real-world phenomena.

Consider an academic collaboration network. Suppose S is a group of scholars, say, representing researchers in "Complexity Theory." Suppose $p \in S$ is a leader of the field who has diverse and highly interdisciplinary interests, i.e., having a great deal of collaboration with individuals outside of "Complexity Theory" (for example, say, p = "Papadimitriou"). Then, the conductance of S would be larger than the conductance of $S - \{p\}$, which seems to be a counterintuitive characterization of this research community. In this example, conductance penalizes the "community membership" of the leaders who have high capacity and broad interdisciplinary interests.

Conductance is *scaling-invariant*. Although it is not always a necessary condition for clusterability,[9] scaling-invariant measures are desirable when comparing the clusterability of subsets in different networks:

Definition 8.12 (Scaling-Invariant Clusterability Measures)**.** A clusterability measure ϕ is *scaling invariant* if for any $G = (V, E, \mathbf{W})$, and $\forall S \subseteq V$ and $\forall \alpha > 0$, $\phi(S)$ measured in G is the same as it is measured in $G' = (V, E, \alpha \cdot \mathbf{W})$.

We will now consider a family of scaling-invariant clusterability measures, some of which can better capture what conductance fails to do in the discussion above. To define these measures, recall that for a subset $S \subseteq V$ in a network $G = (V, E, \mathbf{W})$, S divides the edges

[9] For example, the cut-ratio of Section 4.6 and the modularity [261] are not scaling-invariant.

incident to $v \in S$ into two sets: (1) *external edges:* $\partial_v(S) = \{(v,j) \in E : j \in \bar{S}\}$, and (2) *internal edges:* $\{(v,j) \in E : j \in S\}$. Let $d_v^{external(S)} := \sum_{j \in \bar{S}} w_{v,j}$. Then, $\phi_v^S = \frac{d_v^{external(S)}}{d_v}$ measures the fraction of v's weights used for external connection with respect to S.

An intuitive characterization of good clusters is that they have significantly more internal connection than external connection. However, there are many seemingly plausible ways to formalize this simple intuitive characterization. We start with the two natural ones:

$$\begin{aligned} \text{conductance}_{\mathbf{W}}(S) &= \frac{\text{cut}_{\mathbf{W}}(S,\bar{S})}{\text{vol}_{\mathbf{W}}(S)} = \frac{\sum_{v \in S} d_v^{external(S)}}{\sum_{v \in S} d_v} \\ \text{cohesion}_{\mathbf{W}}(S) &= \frac{\sum_{v \in S} \phi_v^S}{|S|} = \frac{\sum_{v \in S} \frac{d_v^{external(S)}}{d_v}}{|S|} \end{aligned} \tag{8.2}$$

While $\text{conductance}_{\mathbf{W}}(S)$ measures the fraction of S's external connection, $\text{cohesion}_{\mathbf{W}}(S)$ measures the average fraction of external connection over nodes in S. Both clusterability measures are scaling-invariant with values in $[0,1]$, where 0 implies that S is disconnected from $\bar{S}$, and 1 implies that S is an independent set of G (i.e., all weights of S are for external connection). Intuitively, if S has small conductance or cohesion, S "statistically favors" internal connection over external connection. However, in spite of their similarity, as illustrated below, these two clusterability measures can be significantly different.

Example 8.13. Let G be the "star" graph where the *center vertex* n is connected with $[n-1]$, and each of them only connects with n.

For all $S \subset V$ in this graph:

$$\text{conductance}(S) \geq 1/2.$$

Thus, this network has no good clusters, according to the conductance measure. However, for $S = [1 : n/2 - 1] \cup \{n\}$:

$$\text{cohesion}_{\mathbf{W}}(S) = 2/n$$

suggesting S is a good cluster according to the cohesion measure. For this graph, conductance and cohesion seem to lead to significantly different conclusions regarding the clusterability of subsets.

Which measure is more relevant of network clusterability? Is conductance overly pessimistic, or is cohesion overly optimistic?

Example 8.14. Suppose G is a graph containing a subset $S = \{v_1, ..., v_k\}$ where (i) $v_1, ..., v_{k-1}$ has no outside links, but each is connected with $\sqrt{k}$ nodes in $S - \{v_k\}$, and (ii) v_k connects with everyone in $v_1, ..., v_{k-1}$. In addition, it connects with k^2 nodes in $\bar{S}$.

Let us consider two subsets $S = \{1, ..., k\}$ and $T = S - \{k\}$. Then:

$$\begin{aligned}
\text{conductance}(S) &= \frac{k^2}{k^2 + (k-1) + (k-1)\cdot(\sqrt{k}+1)} \approx 1 \\
\text{conductance}(T) &= \frac{(k-1)}{(k-1)\cdot(\sqrt{k}+1)} \approx \frac{1}{\sqrt{k}} \\
\text{cohesion}(S) &= \frac{1}{k}\cdot\frac{k^2}{k^2+k-1} \leq \frac{1}{k} \\
\text{cohesion}(T) &= \frac{1}{k-1}\cdot(k-1)\cdot\frac{1}{\sqrt{k}+1} \approx \frac{1}{\sqrt{k}}
\end{aligned}$$

This example shows that cohesion might be more indicative than conductance for S. Note that member v_k holds the most connections within S. However, because it also connects with many nodes outside S, the inclusion of v_k to T greatly increases the conductance measure, which jumps from $1/\sqrt{k}$ to almost 1. So, the conductance indicates that T is a much stronger cluster than S.

Let's return to our "Complexity Theory Scholars". Imagine Example 8.14 is a hypothetical collaboration network, where S represents the "Complexity Theory Scholars" and v_k denotes the (highly interdisciplinary) leader of this field. Then, according to conductance, without the leader v_k, $T = S - \{v_k\}$ becomes a stronger community than S. In contrast, according to cohesion, T is an "ok" cluster. But, with its leader, S is a much better cluster than T.

These two examples show that if one node in the subset has a relatively large $d_v^{external(S)}$ and d_v, then its out-links could dominate in the conductance measure. In contrast, because cohesion measures

the average fraction that nodes in S use for external connection, and because $\forall v \in V,\ 0 \leq d_v^{external(S)}/d_v \leq 1$, nodes with large degrees cannot overly dominate the clusterability measure.

The difference between conductance and cohesion can be further illustrated by their connection with *vertex separators* defined in Section 5.6. The following proposition shows that the separator quality is more closely related to cohesion than with conductance. However, this proposition has no conductance analogue, with Example 8.13 as a counterexample.

Proposition 8.15 (Cohesion and Separators)**.** For $\alpha \in [\frac{1}{2} : 1]$, if a graph $G = (V, E)$ has an α-vertex separator C of size $f(n)$ then there exists a cluster $S \subset V$ with $|S| \in [(1-\alpha)n : \frac{n}{2}]$ such that:

$$\text{cohesion}(S) = \frac{f(n)}{(1-\alpha)n} \tag{8.3}$$

Proof. By definition, $V - C$ can be partitioned into two subsets A and B such that:

1. there exists no edge between A and B, and
2. $\max(|A|, |B|) \leq \alpha n$.

Assume $|A| \leq |B|$. Then, $(1-\alpha)n \leq |A| \leq \frac{n}{2}$. Let $S = A \cup C$ if $|A \cup C| \leq \frac{n}{2}$ and $S = V - A \cup S$, otherwise.

Then, $(1-\alpha)n \leq |S| \leq \frac{n}{2}$, and $\text{cohesion}_{\mathbf{W}}(S) \leq \frac{|C|}{|S|} \leq \frac{f(n)}{(1-\alpha)n}$. □

We will now consider a family of scaling-invariant clusterability measures that contains both conductance and cohesion as special cases.

Definition 8.16 (Parameterized Cohesion)**.** Let $\boldsymbol{\beta} = (\beta_1, ..., \beta_n)$ be a non-negative vector. For a subset $S \subset V$, $(\beta_v : v \in S)$ can be used to define a *convex combination* of $\phi_v^S = d_v^{external(S)}/d_v$:

$$\boldsymbol{\beta}\text{-cohesion}(S) = \frac{\sum_{v \in S} \beta_v \phi_v^S}{\sum_{v \in S} \beta_v} = \frac{\sum_{v \in S} \beta_v \cdot \frac{d_v^{external(S)}}{d_v}}{\sum_{v \in S} \beta_v}. \tag{8.4}$$

We obtain the conductance measure when $\beta_v = d_v$, and the cohesion measure when $\beta_v = 1$. In general, parameters $\beta_1, ..., \beta_n$ can be used to model the significance scores — such as centrality — assigned to nodes in the network. For example, other seemingly reasonable choices of $\boldsymbol{\beta}$ include:

- **PageRank-Cohesion:** $\beta_v = p_v$, the PageRank of node v.
- **Sublinear-Cohesion:** $\beta_v = d_v^\alpha$, for $\alpha \in [0, 1]$. By lowering α, one can reduce the dominance of large degree nodes.

  For example, when $\alpha = 1/2$ we have:

$$\boldsymbol{\beta}\text{-cohesion}(S) = \frac{\sum_{v \in S} \sqrt{d_v} \phi_v^S}{\sum_{v \in S} \sqrt{d_v}} \tag{8.5}$$

$$= \frac{\sum_{v \in S} \frac{d_v^{external(S)}}{\sqrt{d_v}}}{\sum_{v \in S} \sqrt{d_v}}. \tag{8.6}$$

One may quickly note that there is an alternative way to define this family of clusterability measures.

Definition 8.17 ($\boldsymbol{\gamma}$-conductance)**.** Let $\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_n)$ be a non-negative vector (modeling the weights assigned to vertices in the network). For a subset $S \subset V$, let:

$$\boldsymbol{\gamma}\text{-conductance}(S) = \frac{\sum_{v \in S} \gamma_v \cdot d_v^{external(S)}}{\sum_{v \in S} \gamma_v \cdot d_v} \tag{8.7}$$

We obtain conductance when $\gamma_v = 1$ and cohesion when $\gamma_v = 1/d_v$.

Lemma 8.18 (Conductance vs Cohesion)**.** If $\gamma_v = \beta_v / d_v$ then:

$$\boldsymbol{\gamma}\text{-conductance}(S) = \boldsymbol{\beta}\text{-cohesion}(S).$$

Thus, conductance and cohesion can be viewed as dual formulations for capturing the intuitive characterization that good clusters/communities should have significantly more internal connection than external connection.

Remark: Modularity

The above family of clusterability measures does not contain *modularity* — a popular clusterability measure [261] — as a special case. To define modularity, consider the following $n \times n$ matrix:

$$\boldsymbol{M} = \mathbf{W} - \frac{1}{\text{vol}_W(V)} \boldsymbol{d}\boldsymbol{d}^T \tag{8.8}$$

where $\mathbf{d} = (d_1, ..., d_n)$ are the weighted degrees. $\boldsymbol{M}$ is called the *modularity matrix* of G. For a subset S, let $\text{modularity}(S) := \sum_{u,v \in S} \mathbf{M}[u, v]$. Thus, if $\text{modularity}(S) = 0$, the substructure of S is "as expected" in a "random" network that "preserves" the (weighted) degrees of all vertices. The modularity of S measures the discrepancy between the expected and the induced subgraphs over S in G. Newman [261] considers S a good cluster/community if $\text{modularity}(S) \gg 0$. In such cases, S is more connected internally than "expected" by random chance. Not surprisingly, like conductance, the modularity of S can be expressed as a function of $\text{vol}_{\mathbf{W}}(S)$ and $\text{cut}_{\mathbf{W}}(S, \bar{S})$:

$$\begin{aligned} \text{modularity}(S) &= \sum_{u,v \in S} \mathbf{M}[u, v] = \sum_{u,v \in S} w_{u,v} - \sum_{u,v \in S} \frac{d_u d_v}{\text{vol}_{\mathbf{W}}(V)} \\ &= \text{vol}_{\mathbf{W}}(S) \left(\frac{\text{vol}_{\mathbf{W}}(\bar{S})}{\text{vol}_{\mathbf{W}}(V)} - \text{conductance}_{\mathbf{W}}(S) \right). \end{aligned}$$

Particularly, as seen in the last equation, the modularity of S is correlated with the conductance of S. While modularity is not scale-invariant, the following two related quantities are:

$$\begin{aligned} \frac{\sum_{u,v \in S} \mathbf{M}[u, v]}{\sum_{u,v \in S} \frac{d_u d_v}{\text{vol}_{\mathbf{W}}(V)}} &= \left(\frac{\text{vol}_{\mathbf{W}}(\bar{S})}{\text{vol}_{\mathbf{W}}(S)} \right) - \text{conductance}_{\mathbf{W}}(S) \left(\frac{\text{vol}_{\mathbf{W}}(V)}{\text{vol}_{\mathbf{W}}(S)} \right) \\ \frac{\sum_{u,v \in S} \mathbf{M}[u, v]}{\text{vol}_{\mathbf{W}}(S)} &= \left(\frac{\text{vol}_{\mathbf{W}}(\bar{S})}{\text{vol}_{\mathbf{W}}(V)} \right) - \text{conductance}_{\mathbf{W}}(S). \end{aligned}$$

Does a unified parameterization that contains all or most of these popularly-used clusterability measures exists? Such a parameterization would provide a comparative framework for evaluating these correlated formulations, and for designing "optimal" application-specific clusterability measures.

Discussion: Clusterability in Directed Networks

So far, our discussions of clusterability and clustering algorithms have been focused on undirected weighted networks. However, most real-world networks are directed. Both in theory and in practice, the clustering structures in directed networks appear to be more complex. Several mathematical concepts, including modularity and conductance, have directed extensions that are intuitively reasonable. However, the existing mathematical characterizations and algorithmic tools — such as Cheeger's inequality for the former and local clustering and electrical-flow algorithms for the latter — are still limited to the undirected case.

Remarkably, for centrality measures, PageRank applies to both directed and undirected networks. In fact, the personalized PageRank matrix (Definition 3.5) provides the following natural clusterability measures for directed weighted networks:

Definition 8.19 (PageRank Clusterability)**.** Suppose $\alpha > 0$ is a restart constant and $G = (V, E, \mathbf{W})$ is a directed weighted network. For any $S \subseteq V$, let:

$$\text{clusterability}^{\text{IPR}}_{\mathbf{W},\alpha}(S) = \frac{1}{|S|} \cdot \sum_{u,v \in S} \mathbf{PPR}_{\mathbf{W},\alpha}[u, v] \tag{8.9}$$

$$\text{clusterability}^{\text{EPR}}_{\mathbf{W},\alpha}(S) = \frac{1}{|S|} \cdot \sum_{u \in S, v \notin S} \mathbf{PPR}_{\mathbf{W},\alpha}[u, v] \tag{8.10}$$

These two clusterability measures are scaling-invariant, with values in $[0, 1]$. As an analogue of edge-density, $\text{clusterability}^{\text{IPR}}_{\mathbf{W},\alpha}(S)$ measures the fraction of S's PageRank contributions to members within S. Similarly, as an analogue of conductance, $\text{clusterability}^{\text{EPR}}_{\mathbf{W},\alpha}(S) \in [0, 1]$ measures the fraction of S's PageRank contribution to nodes outside S. These two clusterability measures complement each other:

$$\text{clusterability}^{\text{IPR}}_{\mathbf{W},\alpha}(S) + \text{clusterability}^{\text{EPR}}_{\mathbf{W},\alpha}(S) = 1.$$

However, there is a subtle difference between them, which is related to the question of whether a directed network can be "symmetrized" in clusterability measures. The internal version, $\text{clusterability}^{\text{IPR}}_{\mathbf{W},\alpha}$, works

with principal submatrices of $\mathbf{PPR}_{\mathbf{W},\alpha}$, and thus summarizes the "symmetrized" PageRank matrix: $\frac{1}{2}\left(\mathbf{PPR}_{\mathbf{W},\alpha} + \mathbf{PPR}_{\mathbf{W},\alpha}^T\right)$. In other words:

$$\text{clusterability}^{\text{IPR}}_{\mathbf{W},\alpha}(S) = \frac{1}{|S|} \cdot \sum_{u,v \in S} \frac{1}{2} \left(\mathbf{PPR}_{\mathbf{W},\alpha}[u,v] + \mathbf{PPR}_{\mathbf{W},\alpha}[v,u]\right).$$

On the other hand, the external version, $\text{clusterability}^{\text{EPR}}_{\mathbf{W},\alpha}$, works with submatrices of form $\mathbf{PPR}_{\mathbf{W},\alpha}[S,\bar{S}]$. This data is inherently asymmetric, even for undirected networks. In other words, it is the typical case that:

$$\frac{1}{|S|} \cdot \sum_{u \in S, v \notin S} \mathbf{PPR}_{\mathbf{W},\alpha}[u,v] \neq \frac{1}{|S|} \cdot \sum_{u \notin S, v \in S} \mathbf{PPR}_{\mathbf{W},\alpha}[u,v].$$

To further highlight the issue of "network symmetrization," let's consider another relevant clusterability measure. Let $\mathbf{W}^{\text{symPPR}}$ denote the *symmetric PageRank matrix* of $\mathbf{W}$:

$$\mathbf{W}^{\text{symPPR}} = \frac{1}{2}\left(\mathbf{PPR}_{\mathbf{W},\alpha} + \mathbf{PPR}_{\mathbf{W},\alpha}^T\right) \tag{8.11}$$

For $S \subseteq V$, let's define $\text{clusterability}^{\text{symPR}}_{\mathbf{W},\alpha}(S)$ as the conductance of S in undirected weighted network given by $\mathbf{W}^{\text{symPPR}}$:

$$\text{clusterability}^{\text{symPR}}_{\mathbf{W},\alpha}(S) := \text{conductance}_{\mathbf{W}^{\text{symPPR}}}(S).$$

This symmetrized clusterability measure is fundamentally different from $\text{clusterability}^{\text{IPR}}$ because it is usually case that for S:

$$\text{clusterability}^{\text{IPR}}(S) \neq 1 - \text{clusterability}^{\text{symPR}}_{\mathbf{W},\alpha}(S).$$

Both work with symmetrized $\mathbf{PPR}_{\mathbf{W},\alpha}$. However, the former measures the PageRank-contribution density with respect to $|S|$. The latter adaptively measures the PageRank-contribution density with respect to the total of $\mathbf{PPR}_{\mathbf{W},\alpha}[S,S]$, $\mathbf{PPR}_{\mathbf{W},\alpha}[\bar{S},S]/2$ and $\mathbf{PPR}_{\mathbf{W},\alpha}[S,\bar{S}]/2$.

On the other hand, it is usually case that for S:

$$\text{clusterability}^{\text{EPR}}(S) \neq \text{clusterability}^{\text{symPR}}_{\mathbf{W},\alpha}(S).$$

The former is independent of $\mathbf{PPR}_{\mathbf{W},\alpha}[\bar{S},S]$. The latter symmetrically incorporates data from $\mathbf{PPR}_{\mathbf{W},\alpha}[\bar{S},S]$.

Note that Eqn. (8.11) is only one of many ways to symmetrize $\mathbf{PPR}_{\mathbf{W},\alpha}$. By Theorem 3.7, $\mathbf{PPR}_{\mathbf{W},\alpha}$ is a stochastic matrix. Thus, we can also use the following basic result to symmetrize personalized PageRank matrix.

Lemma 8.20 (Markovian Symmetrization)**.** For any stochastic matrix $\mathbf{M}$:

$$\boldsymbol{\Pi} - \frac{\boldsymbol{\Pi}\mathbf{M} + \mathbf{M}^T\boldsymbol{\Pi}}{2} \tag{8.12}$$

is a Laplacian matrix, where $\boldsymbol{\Pi}$ the diagonal matrix associated with $\mathbf{M}$'s stationary distribution.

Proof. Let $\boldsymbol{\pi}$ be the stationary distribution of $\mathbf{M}$. Then:

$$\mathbf{M}^T\boldsymbol{\pi} = \boldsymbol{\pi}, \quad \boldsymbol{\Pi} \cdot \mathbf{1} = \boldsymbol{\pi}, \quad \text{and} \quad \mathbf{M} \cdot \mathbf{1} = \mathbf{1}$$

Therefore:

$$\left(\boldsymbol{\Pi} - \frac{\boldsymbol{\Pi}\mathbf{M} + \mathbf{M}^T\boldsymbol{\Pi}}{2}\right) \cdot \mathbf{1} = \left(\boldsymbol{\pi} - \frac{\boldsymbol{\Pi}\mathbf{1} + \mathbf{M}^T\boldsymbol{\pi}}{2}\right) = \mathbf{0} \tag{8.13}$$

The Lemma then follows from the fact that $\frac{1}{2}(\boldsymbol{\Pi}\mathbf{M} + \mathbf{M}^T\boldsymbol{\Pi})$ is symmetric and non-negative. □

Thus, we can also use the Markovian symmetrization of $\mathbf{PPR}_{\mathbf{W},\alpha}$ to define clusterability measures. Schematically, these personalized PageRank-based clusterability measures are intuitively reasonable and elegant. Mathematically, we need to address the following fundamental questions to better understand them:

> *What do these PageRank clusterability measures capture? Are they relevant to clusterings in real-world networks? Is there a scalable algorithm for identifying groups with significant PageRank clusterability measures? How can we extend these clusterability measures to other network data?*

As processes and graph structures in network phenomena become increasingly asymmetric, deeper mathematical and algorithmic understandings of asymmetric interactions/affinities/preferences are needed. This is also an area where broader network models and solution concepts could be essential.

8.3 Data Clustering: Intuition versus Ground Truth

An important application of multi-way partitioning (Section 5.7) is *clustering*. For a network $G = (V, E, \mathbf{W})$ or a metric space $\mathcal{M} = (V, \text{dist})$, a *clustering* is a partition of V. Particularly, for an integer $K \in [|V|]$, a *K-clustering* divides V into K-non-empty subsets $(V_1, ..., V_K)$.

Many intuitively meaningful objective functions for clustering have been proposed. Even more algorithms have been developed to identify clusterings that approximate these objective functions. However, the central challenging conceptual question remains:

> *What constitutes a desirable clustering for nodes in a network or objects in a metric space?*

The conceptual challenge in this basic question is genuine, as illustrated by Kleinberg [207]. Focusing on data clustering in metric spaces, his famous impossibility theorem for "clustering by partitioning" articulates the difficulty even for defining a "consistent" clustering scheme.

In the history of computing, one (wonderful) thing that has repeatedly been proven is that people never stop trying because of theoretical intractability or impossibility. Many of these attempts started as heuristics, but eventually led to new mathematical questions and theoretical frameworks. In this section, we discuss an insightful theory developed by Balcan, Blum, and Gupta [38]. While their theory can be extended to broader settings, we will focus on clustering data in metric spaces.

Some Popular Objective Functions for Data Clustering

For clustering geometric data in Euclidean spaces (e.g., $P = \{\mathbf{p}_1, ..., \mathbf{p}_n\} \subset \mathbb{R}^d$), a widely used method is the K-means cluster-

ING. Its objective is to find a K-clustering $P_1, ..., P_K$ that minimizes the total variances of the clustering:

$$K\textbf{-means}: \quad \Phi_{\text{means}}((P_1, ..., P_K)) = \sum_{i=1}^{K} \sum_{\mathbf{p} \in P_i} ||\mathbf{p} - \mathbf{c}_i||^2,$$

where $\mathbf{c}_i = \text{mean}(P_i) = \frac{\sum_{\mathbf{p} \in P_i} \mathbf{p}}{|P_i|}$ is the mean of the points in P_i.

Other similar objective functions for minimization include:

$$K\textbf{-median}: \quad \Phi_{\text{median}}((P_1, ..., P_K)) = \sum_{i=1}^{K} \min_{\mathbf{c}_i \in \mathbb{R}^d} \sum_{\mathbf{p} \in P_i} ||\mathbf{p} - \mathbf{c}_i||.$$

$$K\textbf{-sum}: \quad \Phi_{\sum}((P_1, ..., P_K)) = \sum_{i=1}^{K} \sum_{\mathbf{p}, \mathbf{q} \in P_i} ||\mathbf{p} - \mathbf{q}||.$$

We can extend these functions to metric spaces: Suppose $V \subseteq X$ is a set of data points in a metric space $\mathcal{M} = (X, \text{dist})$, and $\{V_1, ..., V_K\}$ is a K-clustering of V. One can extend Φ_{means} as:

$$\Phi_{\text{means}}((V_1, ..., V_K)) = \sum_{i=1}^{K} \min_{\mathbf{c}_i \in X} \sum_{\mathbf{p} \in V_i} ||\mathbf{p} - \mathbf{c}_i||^2$$

and apply $\Phi_{\sum}$ and Φ_{median} directly.

A Spectral Objective Function for Geometric Clustering

In this context, Theorem 5.2 offers a conductance-based objective function, $\Phi_{\text{k-NNG-conductance}}$, for clustering geometric data:

> *Given a point set P in $\mathbb{R}^d$, find a K-clustering $(P_1, ..., P_K)$ that minimizes the maximum fraction of external k-NNG relations over all clusters.*

For low-dimensional data points, Theorem 5.2 provides a remarkable guarantee to this objective function. It shows that every point set in a fixed dimensional space has a K-clustering in which every cluster retains all but $o(1)$-fraction of its k-NNG relations.

Basic Questions

Intuitively, keeping k-NNG relations within the clusters is desirable. Similarly, minimizing K-means, K-median, and K-sum are all desirable. However, returning to the above conceptual question, one must

ask whether partitions with strong guarantees of these objective functions provide good solutions for data clustering. More generally:

> *To what degree do these objective functions capture the "ground-truth" clustering of the underlying data?*

In addition, some of these optimization problems are potentially intractable. As a result, approximation algorithms are used to obtain clustering. For example, K-MEANS optimization is NP-hard in general, and Lloyd's practical algorithm[10] [238, 34] is often used to obtain a locally optimal solution. Thus, another fundamental question arises regarding clustering approaches that approximate objective functions:

> *To what degree do the approximate solutions of an objective function capture the "ground-truth" clustering?*

Stability of an Objective with Respect to Ground Truth

Balcan, Blum, and Gupta [38] present an elegant characterization when addressing this basic question. They introduce a notion of *approximation stability* to connect the *ground-truth clustering* and clusterings found by approximately optimizing an objective function. They use the set-theoretical distance to measure the closeness between clusterings: Suppose $C = \{V_1, ..., V_K\}$ and $C' = \{V'_1, ..., V'_K\}$ are two K-clusterings of $V = [n]$. Let's view them as two K-way classifiers of V. Let $\Delta(C, C')$ denote the fraction of elements of V that C mis-classifies C' under the best possible matching between their class labels.

Definition 8.21 (Distance to Ground-Truth)**.** For a metric space $\mathcal{M} = (X, \text{dist})$ and a data set $V \subset X$, suppose $C^{\text{GT}} = \{V_1^{\text{GT}}, ..., V_K^{\text{GT}}\}$ is the "*ground-truth*" K-clustering. Then, a K-clustering $C = \{V_1, ..., V_K\}$ is ϵ-*close* to the ground-truth if $\Delta(C, C^{\text{GT}}) \leq \epsilon$.

---

[10] Lloyd's algorithm iteratively re-clusters the data: At each step, it computes the "center" of each cluster. It then uses the Voronoi diagram of these centers to define a new partition. This iterative process converges to a local optimum of K-MEANS CLUSTERING.

Balcan, Blum, and Gupta [38] use the following definition.

Definition 8.22 (Approximation Stability)**.** Suppose Φ is an objective function for K-clusterings in a metric space $\mathcal{M} = (X, \text{dist})$. Suppose C^{OPT} is the K-clustering of $V \subseteq X$ that optimizes Φ. For $c > 1$ and $\epsilon > 0$, the input instance $(\mathcal{M}, V)$ is (c, ϵ)-*approximation-stable* respect the ground-truth K-clustering C^{GT} according to Φ if every K-clustering C whose objective value $\Phi(C)$ is within a multiplicative factor $c > 1$ of $\Phi(C^{\text{OPT}})$ is ϵ-close to C^{GT}.

Definition 8.22 explicitly assumes that not only is C^{OPT} ϵ-close to the ground-truth C^{GT}, but also all approximately good K-clusterings — according to Φ — are ϵ-close to C^{GT}.

The rationale behind this definition is the following: Suppose one is committed to using an approximate optimal solution of Φ to formulate a K-clustering of V. Then, such a decision is reasonable *only if* the K-clusterings obtained by approximately optimizing Φ are reasonably close to the target ground-truth. Otherwise, what is the justification for optimizing Φ?

Balcan, Blum, and Gupta [38] make this rationale explicit, and prove the following insightful result: If every approximation of the K-means's objective function is close to the ground-truth, then one can in fact find a clustering close to the ground-truth without actually approximately solving K-MEANS CLUSTERING.

Theorem 8.23 (Approximation Stability)**.** Suppose $(\mathcal{M}, V)$ is $(1 + \alpha, \epsilon)$-approximation-stable with respect to C^{GT} according to Φ_{median}. Then, there exists a polynomial-time clustering algorithm for constructing a K-clustering $O(\epsilon + \epsilon/\alpha)$-close to C^{GT} (without actually finding an approximate solution to Φ_{median}.

Proof. (High-level Sketch). Suppose $C^{\text{OPT}} = \{V_1^{\text{OPT}}, ..., V_K^{\text{OPT}}\}$ and $C^{\text{GT}} = \{V_1^{\text{GT}}, ..., V_K^{\text{GT}}\}$, where V_i^{OPT} is the cluster closest to V_i^{GT}.

Let $H_i = V_i^{\mathrm{OPT}} \cap V_i^{\mathrm{GT}}$ and $\bar{V} = \cup_{i\in[K]} H_i$. Because C^{OPT} is ϵ-close to C^{GT}, we have $|V - \bar{V}| \leq \epsilon n$. Let $r = \frac{1}{n} \cdot \Phi_{\mathrm{median}}(C^{\mathrm{OPT}})$ be the average distance from points in V to their cluster centers in C^{OPT}. Let $r_{critical} = \frac{\alpha \cdot r}{10\epsilon}$. We call a point $v \in \bar{V}$ "good" if (1) v is less than $r_{critical}$ away from its cluster center, and (2) v is at least $5r_{critical}$ away from any other cluster center. Let V_{G} be the set of "good" points. Then, by an averaging argument [38], one can establish the following: Because $(\mathcal{M}, V)$ is $(1+\alpha, \epsilon)$-approximation-stable, $|\bar{V} - V_{\mathrm{G}}| = O(\frac{\epsilon}{\alpha} n)$. Let $\tau = 2r_{critical}$ and $V_{\mathrm{B}} = V - V_{\mathrm{G}}$. Then, the following holds.

1. **Small outliers**: $|V_{\mathrm{B}}| \leq O((\epsilon + \frac{\epsilon}{\alpha})n)$, and
2. **Well-separated clusters**: For each $i \in [K]$, let U_i denote $V_i^{\mathrm{OPT}} \cap V_{\mathrm{G}}$. Then (i) $\forall i$, all pair-wise distances in U_i are less than τ, and (ii) $\forall j \neq i$, U_i and U_j are more than 2τ apart.

Now, let's construct a graph over V by connecting points that are at most τ away from each other. For each i, suppose u_i is an arbitrary point in U_i. Let $\tilde{V}_i$ denote the set consisting of u_i and its neighbors in the constructed graph. The clustering algorithm stated in Theorem 8.23 uses the following key observations:

- (A) $U_i \subseteq \tilde{V}_i \subseteq U_i \cup V_B$, for all i, and
- (B) $U_i \cap U_j = \emptyset$, for all $i \neq j$.

The first part of (A) holds, because U_i forms a clique (implied by Condition (2.i)). The second part of (A) holds, because for all $i \neq j$, U_i and U_j have no edge between them (implied by Condition (2.ii)). Observation (B) follows also from Condition (2.ii), which implies that for all $i \neq j$, U_i and U_j have no common neighbor in the graph constructed. There are two basic cases: (1) Every cluster in C^{GT} is larger than V_B by a constant factor. (2) Some V_i^{GT} is smaller than V_B. For Case (1), we identify $\tilde{V}_i$ iteratively, and obtain a K-clustering by arbitrarily assigning $V - (\cup_i^K \tilde{V}_i)$ into these clusters. The resulting clustering is $O(\epsilon + \epsilon/\alpha)$-close to C^{OPT}, which is $O(\epsilon + \epsilon/\alpha)$-close to C^{GT}. Case (2) requires more careful analyses. We refer readers to [38]. □

This proof can be extended to Φ_{means} and partially extended to $\Phi_{\sum}$ [38, 40]. The approximation-stable point sets according to Φ_{means} have a similar but more subtle clustering structure, which can be used in a graph-based method that adaptively adjusts the threshold of the targeted cluster diameters and the distances between clusters [38].

Adaptive Sampling for Scalable k-Median Clustering

The clustering algorithms of [38] run in polynomial time. However, these algorithms are not scalable because they explicitly construct a threshold graph, which may have a quadratic number of edges. To compute the threshold distance τ, these algorithms also need to know the distances between every pair of points in V. Thus, the straightforward implementation of these algorithms takes $O(n^3)$ time.

Voevodski *et al.* [347] developed a fast sampling scheme to reduce clustering time to $O(Kn \log n)$. In addition to being scalable, this algorithm is preferable for applications, such as genome sequencing, where distances between points — from which pair-wise similarities are derived — are not known *a priori*. In such applications, it is often more cost-effective to make *one-versus-all* distance queries[11] [345, 346]. Voevodski *et al.*'s algorithm makes $O(K)$ *one-versus-all* distance queries.

Theorem 8.24 (Scalable Ground-Truth Clustering)**.** Suppose $V \subset X$ is a data set in an *unknown* metric space $\mathcal{M}$, and (i) $(\mathcal{M}, V)$ is $(1+\alpha, \epsilon)$-approximation stable with respect to C^{GT} according to Φ_{median}, and (ii) every cluster in C^{GT} has size $3 \cdot \frac{\epsilon}{\alpha} \cdot n$. Then, one can construct a K-clustering $O(\epsilon + \frac{\epsilon}{\alpha})$-close to C^{GT}, in expected $O((K + n \log n)$ time, by making $O(K)$ one-versus-all distance queries.

Proof. (Sketch) At the heart of this construction is a *landmark-selection* scheme, which adaptively samples V, without any prior distance infor-

[11] In the *one-versus-all* distance-query model of a data set V in a metric space $\mathcal{M}$, the user specifies a node $v \in V$, and obtains the distance profile from v to all nodes in V.

mation. It builds an "approximate" distance map by performing the one-versus-all distance queries with respect to the landmarks.

Let V_{G}, V_{B}, and $U_i : i \in [K]$ be subsets defined in the proof of Theorem 8.23. The fact that C^{OPT} is ϵ-close to C^{GT} and the assumption that every cluster in C^{GT} has at least $O(\epsilon/\alpha) \cdot n)$ points implies that every U_i has at least $O(\frac{\epsilon}{\alpha} \cdot n)$ points. Let $b = O((\epsilon + \frac{\epsilon}{\alpha})n)$ be an estimated upper bound on $|V_{\mathrm{B}}|$ such that $|U_i| \geq 2b, \forall i \in [K]$.

> *Note that at the start of the algorithm, we don't know V_{G}, V_{B}, $(U_1, ..., U_K)$, or the threshold τ. We only know that they exist.*

The goal of landmark selection for this clustering problem is to identify an *effective landmark set* $L \subset V$ that satisfies the following conditions:

1. **small landmarks**: $|L| = O(K)$
2. **clustering triangulating**: for all $i \in [K]$, $\mathrm{dist}(L, U_i) \leq \tau$.

Note that the second condition is much weaker than the condition achieved by the celebrated embedding theory of Johnson-Lindenstrauss [180], which requires that all pair-wise distances be approximately preserved. For clustering, we only need distinguish distances with respect to cluster radii.

In other words, we don't need the landmarks to estimate the distance between two close points as precisely as in Johnson-Lindenstrauss. Thus, we can use a more aggressive reduction methods to improve efficiency. The challenge is to solve this sampling problem without knowing U_i and τ, and without explicitly constructing U_i. The adaptive sampling algorithm is very simple [347]. It generates landmarks iteratively. Let L_t denote the landmark set after t steps.

1. L_1 contains a random element l_1, chosen uniformly from V.
2. At step t, let X_t be the set of the $2b$ points in $V - L_{t-1}$ that are furthest away from L_{t-1}. We select an element l_t, uniformly at random, from X_t, and set $L_t = L_{t-1} \cup \{l_t\}$.
3. We then make a one-versus-all distance query involving l_t.

Let $h = 4K + 16 \log \frac{1}{\delta}$. By the standard Chernoff-Hoeffding bound, with probability at least $1 - \delta$, the final set $L = L_h$ satisfies $|L \cap V_G| \geq K$, i.e., at least K landmarks in L are good points.

The key observation is that $|L \cap V_G| \geq K$ implies L satisfies the "clustering triangulating" property above. To see this, let $v_1, ..., v_K \in V_G$ be the first K good landmarks added to L. If $\{v_1, ..., v_K\} \cap U_i \neq \emptyset$, $\forall i \in [K]$, then L satisfies the "clustering triangulating" property as $\text{dist}(L, U_i) = 0$. Otherwise, by the pigeonhole principle, there must be $i \in [K]$ and two points $u, v \in \{v_1, ..., v_K\} \cap U_i$, implying $\text{dist}(u, v) \leq \tau$. Suppose $u \in L_{t-1}$, and v is added to L_t at step t. Then:

$$\text{dist}(L_{t-1}, v) \leq \text{dist}(u, v) \leq \tau.$$

Thus, if L does not satisfy the clustering triangulating property, then it must be the case that when v is added, there exists $j \neq i$ such that $\text{dist}(L_{t-1}, U_j) > \tau \geq \text{dist}(L_{t-1}, v)$. We know $U_j \cap L_{t-1} = \emptyset$ and $|U_j| \geq 2b$, which contradicts the assumption that v is among the $2b$ furthest points from L_{t-1}.

Once we obtain an effective landmark set, with a careful search from the landmarks, we can complete the clustering process in $O(Kn \log n)$ time. See [347] for more details. □

Discussion: Clusterability of Low-Dimensional Data

Let's return to Theorem 5.2 regarding the clusterability of low-dimensional data: When d is fixed and $k \cdot K = o(n)$, every data point set in $\mathbb{R}^d$ has a "balanced" K-clustering, in which every cluster keeps all but $o(1)$ fraction of k-NNG relations internal.

> *While this is a strong guarantee, how close are these balanced clusterings to the notions of ground-truth of Balcan-Blum-Gupta?*

The Balcan-Blum-Gupta clustering framework [38] provides some insight into this question. For comparative analysis of this clustering result in relation to K-median and K-mean clustering, let's apply Theorem 5.2 to an approximation-stable low-dimensional data set with respect to Φ_{median} or Φ_{means}.

Suppose $C^{\mathrm{GT}} = \{V_1^{\mathrm{GT}}, ..., V_K^{\mathrm{GT}}\}$ is the ground-truth K-clustering according to Φ_{median} (Φ_{means}).

For illustration, assume further:

$$|V_1^{\mathrm{GT}}| \gg |V_2^{\mathrm{GT}}| = \cdots = |V_K^{\mathrm{GT}}|.$$

For example, $|V_1^{\mathrm{GT}}| = \frac{n}{10}$ and for $i \in [2:K]$, $|V_i^{\mathrm{GT}}| = \frac{9}{10}\frac{n}{K-1}$.

When the dimension is low and $k \cdot K = o(n)$, Theorem 5.2 guarantees that V_1^{GT} can be further partitioned into smaller clusters with excellent k-NNG conductance measures. Thus, although this data set is approximation-stable with respect to the ground-truth clustering C^{GT} according to Φ_{median} (Φ_{means}), the data set is not approximation-stable with respect to C^{GT} according to $\Phi_{\text{k-NNG-conductance}}$. In fact, there is no K-clustering, with respect to which the data set is approximation-stable according to $\Phi_{\text{k-NNG-conductance}}$.

Note that, although both Theorem 8.23 and the K-median method find a clustering close to the ground-truth, they use more information about the data set than Theorem 5.2. To see this, note that each cluster in C^{GT} has size $\Omega(\frac{n}{K})$. The algorithm in Theorem 8.23 identifies K clusters, each of which contains a clique of size $\Omega(\frac{n}{K})$.

The graph structure that the algorithm relies on thus has $\Omega(\frac{n^2}{K})$ edges. In contrast, the geometric partitioning algorithm of Theorem 5.2 uses a graph structure (i.e., the k-NNG of the data set) that has only $\Theta(kn)$ edges.

These two structures have size ratio:

$$O\left(kn : \frac{n^2}{K}\right) = O\left(\frac{k \cdot K}{n}\right) = o(1).$$

As such, the algorithm for Theorem 5.2 attempts to cluster the data set with far less information than the algorithm for Theorem 8.23. In low dimensions, data sets may have many good clusterings under this sparse k-NNG clusterability measure. Consequently, the ground-truth may not stand out.

Thus, for real-world data or networks with intrinsically high dimensions, we need to be concerned with the loss of information, whenever we project the data set to a space with a dimension much lower than

the theory of Johnson-Lindenstrauss [180], which is $\Theta(\log n)$. This is especially the case when we want to approximate a high-dimensional data set by a "sparse" network — such as k-NNGs — constructed from its low-dimensional projection. On the other hand, for applications in which data has an effective low-dimensional projection, we can use the clusterability result of Theorem 5.2 to design scalable divide-and-conquer algorithms.

8.4 Behaviors of Algorithms: Beyond Worst-Case Analysis

In this article, the notion of scalable algorithms and its complexity classes S and RS are defined[12] based on the traditional worst-case complexity. When low, the worst-case complexity provides a performance guarantee for all inputs.

Many remarkable polynomial time algorithms ranging from linear programming [202, 188] to primality test [250, 280, 309, 6] have been developed. Several scalable algorithms discussed in this article such as for solving Significant-PageRank Identification, Spectral Sparsification, Geometric Clustering, and Electrical Flow Computation are also measured by their worst-case complexity.

However, the practical behavior of an algorithm is far more complex than indicated by its worst-case performance measure, because the instance-based complexity of the algorithm across the landscape of inputs can differ greatly [316].

This is particularly true when efficient algorithms according to the worst-case measure are not known or may not exist:

> *"It is commonly believed that practical inputs are usually more favorable than worst-case instances."* [316]

Thus, the practical notion of scalable algorithms — for characterizing efficient algorithms for Big Data — is also more general than given by Definition 1.2. The need to more accurately capture the practical performance of algorithms/heuristics has led to a body of rigorous "beyond worst-case" analyses. These analyses and formulations can be applied to scalability.

[12] Definitions 1.2, 1.4 and 1.5.

Below we briefly summarize some of these approaches, which can be thematically organized into two families.

Input Characterization:

Condition-Based Complexity: In this characterization, the complexity $T_A(x)$ of an algorithm A for solving an input instance $x \in \Omega$ is measured by a function with two parameters, $f(\text{size}(x), \kappa(x))$. The function, $\kappa : \Omega \to \mathbb{R}^+$, measures the *condition* of inputs.

Two classical results are stated in this framework.

- The Conjugate Gradient algorithm solves a symmetric positive definite linear system:
$$\mathbf{Ax} = \mathbf{b}$$
to precision ϵ, in time $O(\text{nnz}(\mathbf{A})\sqrt{\kappa(\mathbf{A})}\log\frac{1}{\epsilon})$, where $\kappa(\mathbf{A})$ denotes the condition number of $\mathbf{A}$.

- The Renegar's linear programming algorithm [285] solves a linear program:
$$\max \mathbf{c}^T\mathbf{x} \quad \text{subject to } \mathbf{Ax} \leq \mathbf{b}$$
in time $O(n^3 \log(\kappa(\mathbf{A}, \mathbf{b}, \mathbf{c})))$, where n is the larger dimension of $\mathbf{A}$, and $\kappa(\mathbf{A}, \mathbf{b}, \mathbf{c})$ is inversely proportional to the distance from the linear program given by $(\mathbf{A}, \mathbf{b}, \mathbf{c})$ to the family of ill-posed linear programs [285, 355, 120].

The following is another example of condition-based complexity:

Theorem 8.25 (PageRank). For any network G and constant $\epsilon > 0$, let $\kappa_{\text{PR}}(G) = n/p_G^{\max}$, where $p_G^{\max}$ denotes the largest PageRank value in G. Then, we can identify a node whose PageRank is at least $(1-\epsilon)\cdot p_G^{\max}$ in time $\widetilde{O}_n(1) \cdot (\kappa_{\text{PR}}(G) \log \kappa_{\text{PR}}(G))$.

We can prove Theorem 8.25 by applying binary search to PageRank, using the scalable algorithm for Significant-PageRank Identification of Chapter 3 as its subroutine.

The theory of Vapnik-Chervonenkis [342] also uses a condition-based complexity measure. It states that if the VC dimension of a

concept space is d, then with probability at least $1-\delta$, the *generalization error* of empirical maximization using m random data points is bounded by $O(\sqrt{d\log(m/\delta)/m})$. See Theorem 5.11 for an example. In other words, this fundamental theorem parameterizes the sampling complexity for statistical learning and geometric approximation by the VC dimension of the classification model. The VC dimension can be viewed as the condition number of the input model.

Average-Case Complexity: Average-case analysis is among the first systematic methods used to complement worst-case analysis. In this method, we first determine an input distribution and then evaluate algorithms by their expected performances, assuming inputs are drawn from this distribution. Ideally, this distribution should "capture" the nature of practical data. Average-case analysis has provided insight into the "typical behavior" of algorithms. However, defining analyzable distributions that can capture data actually encountered in practice is fundamentally challenging [316].

Semi-Random and Smoothed Complexity: These frameworks are motivated by the observation that real data are usually neither *completely random* nor *completely arbitrary* [314, 295, 53, 131, 132, 61]. Examples are physical measures, economical data — e.g., stock values — and information data.

In *Smoothed analysis* [314], we assume that input data to an algorithm is subject to slight random perturbations (either due to measurement errors, or the constantly changing market). Thus, the smoothed performance of an algorithm on an input instance is defined to be the algorithm's expected performance over the perturbations of that instance. The algorithm's smoothed complexity is then measured by its maximum smoothed performance over all inputs.

Smoothed complexity usually measures the performance of algorithms in terms of both input size and perturbation magnitude.

By varying perturbation magnitude, smoothed complexity continuously interpolates between worst-case and average-case complexity. Thus, we can use smoothed and other semi-random complexity to gain additional insight for explaining the practical behaviors of algorithms [314].

We can apply smoothed analysis to scalability formulation. Suppose $\sigma(x)$ is a perturbation model for instances in the input domain Ω. We can define:

$$\begin{aligned} \text{smoothed-scalability}(A, x) &:= \mathrm{E}_{\tilde{x}\sim\sigma(x)}\left[\frac{T_A(\tilde{x})}{\text{size}(\tilde{x})}\right]. \\ \text{smoothed-scalability}_A(n) &:= \sup_{x\in\Omega_n} \text{smoothed-scalability}(A, x). \end{aligned}$$

Subclass-Complexity: Instead of characterizing input instances by their distributions, in this framework, we characterize them by their properties. We then measure the performance of an algorithm on inputs which satisfy certain (desirable) properties. Ideally, these properties should capture the nature of practical data.

For example, Lipton-Tarjan's separator theorem focuses only on planar graphs. Likewise, the sphere separator theorem (Theorem 5.22) only applies to fixed-dimensional neighborhood systems that intersect locally. The planar separator theorem is motivated by scientific computing and VLSI design:

- Computer graphics, animation, and 2D numerical simulation usually work with planar graphs [235].
- The final designs in VLSI are planar-like graphs [226, 339].

The geometric separator theorem is largely motivated by problems in three or higher dimensions, where wireless networks, protein folding [353], finite-element methods [253], and N-body simulation [158, 327], work with geometrically defined graphs that are "locally" connected. Both separator theorems lead to scalable divide-and-conquer algorithms for these applications. In the context of linear systems $\mathbf{Ax} = \mathbf{b}$, the scalability of SDD solvers is characterized by its subclass-complexity, i.e., when $\mathbf{A}$ satisfies the SDD property. This subclass characterization is motivated by that matrices studied in spectral graph theory are graph matrices. Another example is Megiddo's remarkable algorithm [247], which solves any linear program in fixed dimensions in linear time. See also [299] for a simpler, randomized scalable fixed-dimensional linear programming algorithm.

One can also apply distribution-based analysis to subclass complexity measure. For example, when analyzing algorithms for the Web and social networks, one often considers distributions of networks satisfying scale-free power-law degree distributions [42, 94].

Solution Characterization:

Output-Sensitive Complexity: In computational geometry, it is common to measure algorithmic performance by both input and output sizes. This is because, for high-dimensional data, the output size can differ greatly from instance to instance, and can be much larger or smaller than the input size. Thus, it is also relevant to measure the performance of algorithms by the size of outputs.

For example, the elegant *gift-wrapping* method [176] has motivated a wonderful line of research for achieving optimal output-sensitive complexity. The original *Jarvis March* takes $O(nh)$ time to construct the convex hull of n two-dimensional points, where h is the number of points on the hull. Algorithms with optimal *output-sensitive complexity* for this problem were obtained by Kirkpatrick and Seidel [204] and Chan [80]. Their algorithms run in $O(n \log h)$ time. Chan's algorithm, which is simpler, can be extended to three dimensions. Seidel [298] developed a high-dimensional convex hull algorithm with complexity $O(n^2 + h \log n)$. Note that, in d dimensions, h ranges from a fixed constant to $O(n^{\lfloor \frac{d}{2} \rfloor})$.

Algorithms for *mesh generation*, a fundamental task in numerical simulation, computer graphics, and manifold approximation, are also measured more naturally by output size. Several breakthrough algorithms in this area [293, 50, 255, 304, 87, 234] are scalable with respect to the size of the *optimal mesh* for the input domain.

Structures of the Solution: For certain computational problems, the structural properties of the solution might be more characteristic than output size.

For example, the performance of the QR algorithm (for solving algebraic eigenvalue problems [350, 155]) is characterized best by the structure of the solution. Recall that the QR algorithm — first developed by Francis [136] — finds all eigenvalue-eigenvector pairs of a given

$n \times n$ matrix $\mathbf{A}$, where entries of $\mathbf{A}$ could be either complex or real. The basic form of the QR method is very simple:

Algorithm: QR($\mathbf{A}$)

1: Let $\mathbf{A}_0 = \mathbf{A}$.
2: **while** $\mathbf{A}_k$ has not sufficiently converged to the upper triangular matrix in the Schur Decomposition of $\mathbf{A}$ **do**
3: Compute a QR-decomposition $(\mathbf{Q}_{k-1}, \mathbf{R}_{k-1})$ of $\mathbf{A}_{i-1}$, i.e., $\mathbf{A}_{k-1} = \mathbf{Q}_{k-1}\mathbf{R}_{k-1}$, where $\mathbf{Q}_{k-1}$ is a unitary matrix and $\mathbf{R}_{k-1}$ is an upper-triangular matrix.
4: Let $\mathbf{A}_k = \mathbf{R}_{k-1}\mathbf{Q}_{k-1}$.
5: Return $T = \mathbf{A}_k$ (and $\mathbf{Q} = \prod_{i=1}^{k} \mathbf{Q}_i$).

QR($\mathbf{A}$) approximately computes a decomposition of $\mathbf{A}$, whose existence is guaranteed by the following famous Schur decomposition theorem [350, 155]:

Theorem 8.26 (Schur Decomposition)**.** $\forall$ complex matrix $\mathbf{A}$, $\exists$ unitary matrix $\mathbf{Q}$ such that, $\mathbf{Q}^H\mathbf{A}\mathbf{Q} = \mathbf{T}$, where $\mathbf{T}$ is an upper triangular matrix whose diagonals are eigenvalues of $\mathbf{A}$. In addition, $\forall$ real matrix $\mathbf{A}$, $\exists$ orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{n\times n}$ such that:

$$\mathbf{Q}^T\mathbf{A}\mathbf{Q} = \begin{pmatrix} \mathbf{R}_{1,1} & \mathbf{R}_{1,2} & \dots & \mathbf{R}_{1,k} \\ \mathbf{0} & \mathbf{R}_{2,2} & \dots & \mathbf{R}_{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{R}_{k,k} \end{pmatrix},$$

where $\mathbf{R}_{i,i}$ is either a scalar or a 2×2 matrix. When $\mathbf{R}_{i,i}$ is a scalar, it is an eigenvalue of $\mathbf{A}$; when $\mathbf{R}_{i,i}$ is a 2×2 matrix, it is the complex conjugate eigenvalues of $\mathbf{A}$.

The convergence of QR($\mathbf{A}$) depends upon the minimum eigenvalue gap of $\mathbf{A}$. In the complex case when $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$, as shown in [350, 155], the lower off-diagonal (i, j)-entry of $\mathbf{A}_k$ $(i < j)$ is:

$$O\left(\left(\frac{\lambda_i}{\lambda_j}\right)^k\right).$$

Note that the minimum eigenvalue gap is explicitly captured by the solution, that is, the diagonal of matrix $\mathbf{T}$ in the Schur decomposition of $\mathbf{A}$. Thus, this convergence result can be viewed either as a condition-based or solution-structure-based characterization.

Complexity characterization based on the structure of the solution has been used for geometric data. For example, motivated by mesh generation and manifold reconstruction (from point-cloud data), Miller *et al.* [251] proved the following algorithmic results. Consider the following structural property of a d-dimensional point set P: For every simplex T in the Delaunay triangulation of P, the ratio of T's circumradius to the length of its shortest edge is bounded by a fixed constant.

Theorem 8.27 (Scalable Delaunay)**.** For any fixed d, for any instance P that satisfies this solution-structural property, the Delaunay triangulation (and hence the Voronoi diagram) of P can be constructed in $O(n \log n)$ time.

Miller *et al.*'s algorithm uses the geometric-separator based divide-and-conquer scheme given in Section 5.5. The Delaunay triangulation of n points in $\mathbb{R}^d$ can have $\Omega(n^2)$ edges in the worst case when $d \geq 3$. Thus, although Theorem 8.27 does not apply to every point set, it applies to any point sets whose Delaunay-triangulation is a desirable solution for numerical discretization [320]. Inspired by this result, Cheng *et al.* [87] took a further step to obtain the following major result in three-dimensional mesh generation:

Theorem 8.28 (Scalable Meshing)**.** If a (periodic) 3D point set $P = \{\mathbf{p}_1, ..., \mathbf{p}_n\}$ can be triangulated, in which all tetrahedra have bounded *aspect-ratio*, then, in $\widetilde{O}(n)$ time, one can construct a triangulation of P in which all tetrahedra have a bounded aspect-ratio.

Theorem 8.28 is characterized by the structure of the solution. The aspect-ratio of a tetrahedron is the ratio of the radius of its smallest containing sphere to the radius of its largest contained sphere.

Performance characterization based on the structure of the solution is also natural for evaluating approximation algorithms. At the most basic level, polynomial-time approximation schemes (PTAS) for intractable problems are measured by both input size and the objective gap between the computed solution and optimal solutions.

Other features of optimal solutions have also been used to further capture algorithmic efficiency or the difficulty of an optimization problem. For example, Theorem 8.23 — the clustering result of Balcan, Blum, and Gupta [38] — is characterized by the stability of optimal solutions with respect to K-median, K-mean clustering, or K-sum objective functions. In optimization, Kleinberg, Papadimitriou, and Raghavan [208] and Feige *et al.* [129] analyze approximation ratios achievable by polynomial time algorithms for Segmentation and Capacitated Maximum Facility Location, two problems that arise in data mining and resource allocation. The work of [129], which is more general, expresses achievable approximation ratios in terms of a *solution signature*, which captures the *cost-utility-tradeoff* in the optimal allocation.

Algorithms that can adapt to the structure of the solution are widely desired in machine learning. For example, the agnostic learning framework of Kearns, Schapire, and Sellie [194] measures the quality of the learned model in terms of the classification errors of the best model. The recent algorithmic results in sparse coding and machine learning [26, 27, 28, 29] also characterize their performance by properties of the structure of the solution. For a detailed discussion, we refer interested readers to Rong Ge's excellent thesis [143].

In this article, going beyond the worst-case is particularly essential for understanding local network exploration. Local algorithms have to work with limited structures near the starting set without examining entire networks. Thus, scalable local algorithms usually do not exist according to the traditional worst-case scenario. As shown in Section 4.2, scalability of local clustering and personalized PageRank approximation can be naturally defined, according to output-sensitive measures. In fact, the statistical guarantee of local clustering is characterized by the structures of both input and output, that is, by the distribution of starting nodes and the random-walk property of the solution clus-

ter. Likewise, the structure of the solution provides a tight scalability characterization for Significant-PageRank Identification.

We believe beyond-worst-case scalability analysis will continue to enrich our algorithmic understanding of Big Data and Machine Learning, and lead us to algorithms that are more adaptive to emerging real-world applications.

8.5 Final Remarks

Big Data and Scalable Computing are a pragmatic match made on earth. While Big Data takes us into the asymptotic world envisioned by our pioneers, it also challenges their classical notions of efficient algorithms. Today, scalable computing is a rapidly advancing field, both in theory and in the real world.

On the theoretical front, the emergence of algebraic and numerical approaches to network analysis has put a spotlight on the scalability of two most fundamental problems:

> *Whether a linear program*
>
> $$\min \mathbf{c}^T\mathbf{x}, \text{ subject to } \mathbf{A}\mathbf{x} \leq \mathbf{b}$$
>
> *or a linear system*
>
> $$\mathbf{A}\mathbf{x} = \mathbf{b}, \text{ where } \mathbf{A} \text{ is a positive-definite matrix}$$
>
> *can be (approximately) solved in time nearly linear in* $\text{nnz}(\mathbf{A})$.

The Laplacian paradigm and its extensions give rise to the exciting possibility that scalable algorithms exist for linear programming and for solving linear systems with positive-definite matrices. In particular, the recent breakthroughs for approximating undirected maximum flows, minimum cuts, and oblivious routing in undirected networks have been initiated by, but have gone beyond, scalable SDD solvers [303, 197, 282, 275, 220, 219].

In the real world, the Web and social networks continue to expand, as do the computational tasks for machine learning and data/network

analysis [164, 79, 229, 246]. Various data analytics and data management systems have been developed to address business and security needs for harnessing network data. These systems require scalable algorithms.

Interdisciplinary fields, centered around Data and Network Sciences, have emerged, that interconnect all traditional areas of computer science: machine learning, algorithms, computer systems, networking, and AI, and other disciplines, including economics, business, natural sciences, social sciences, and medicine. Studies in many of these fields have produced massive amounts of scientific data, such as DNA sequences, population genomics, brain and medical imaging, climate and space data. More than ever, sciences are now empowered by computation and Big Data. These scientific activities require scalable algorithms.

Much of the current theoretical focus has been on developing provably-good scalable algorithms for solving fundamental problems, such as network clustering, graph partitioning, influence maximization, graph sparsification, matrix approximation, linear systems, and linear programs. However, the future for the field of algorithm design is also to formulate the *right* algorithmic problems, objective functions, and mathematical models. This is particularly the case for scalable algorithm design, which is part of the rapidly evolving frontier of network sciences, data sciences, statistical learning, and optimization.

The holy grail of network science is to understand the network essence that underlies the observed sparse-and-multifaceted network data. While tremendous progress has been made, many fundamental challenges remain for understanding Big Data. Solving these basic challenges requires creative thinking to address both conceptual and algorithmic problems. We conclude this article by quoting [84]:

> *Data and network analysis is not just a mathematical task, but also a computational task. In the age of Big Data, networks are massive. Thus, an effective solution concept in network science should be both mathematically meaningful and algorithmically scalable.*

Acknowledgements

The writing of this article is supported in part by a Simons Investigator Award from the Simons Foundation and the NSF grant CCF-1111270.

References

[1] K. V. Aadithya, B. Ravindran, T. Michalak, and N. Jennings. Efficient computation of the Shapley value for centrality in networks. In *Internet and Network Economics*, volume 6484 of *Lecture Notes in Computer Science*, pages 1–13. Springer Berlin Heidelberg, 2010.

[2] E. Abbe and C. Sandon. Community detection in general stochastic block models: fundamental limits and efficient recovery algorithms. *CoRR*, abs/1503.00609, 2015.

[3] E. Abbe and C. Sandon. Recovering communities in the general stochastic block model without knowing the parameters. *CoRR*, abs/1506.03729, 2015.

[4] I. Abraham, M. Babaioff, S. Dughmi, and T. Roughgarden. Combinatorial auctions with restricted complements. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, pages 3–16, 2012.

[5] I. Abraham, Y. Bartal, and O. Neiman. Nearly tight low stretch spanning trees. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 781–790, Oct. 2008.

[6] M. Agrawal, N. Kayal, and N. Saxena. PRIMES is in P. *Annals of Mathematics*, 2:781–793, 2002.

[7] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[8] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. Elsevier North-Holland, Inc., New York, NY, USA, 1989.

[9] D. Aldous and J. Fill. *Reversible Markov chains and random walks on graphs*. Berkeley, 2002. unifinished monograph.

[10] N. Alon. Problems and results in extremal combinatorics - I. *Discrete Mathematics*, 273(1-3):31–53, Dec. 2003.

[11] N. Alon, R. M. Karp, D. Peleg, and D. West. A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computing*, 24(1):78–100, Feb. 1995.

[12] N. Alon and V. D. Milman. λ_1, Isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory*, 38(Series B):73–88, 1985.

[13] N. Alon, P. Seymour, and R. Thomas. A separator theorem for graphs with an excluded minor and its applications. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, STOC '90, pages 293–299, 1990.

[14] N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley, 3rd edition, 2008.

[15] A. Altman and M. Tennenholtz. Ranking systems: The PageRank axioms. In *Proceedings of the 6th ACM Conference on Electronic Commerce*, EC '05, pages 1–8, 2005.

[16] S. Ambikasaran and E. Darve. An $O(N \log N)$ fast direct solver for partial hierarchically semi-separable matrices. *Journal of Scientific Computing*, 57(3):477–501, 2013.

[17] N. Amenta, M. Bern, D. Eppstein, and S.-H. Teng. Regression depth and center points. *Discrete & Computational Geometry*, 23(3):305–323, 2000.

[18] R. Andersen. A local algorithm for finding dense subgraphs. *ACM Transactions on Algorithms*, 6(4):60:1–60:12, Sep. 2010.

[19] R. Andersen, C. Borgs, J. Chayes, U. Feige, A. Flaxman, A. Kalai, V. Mirrokni, and M. Tennenholtz. Trust-based recommendation systems: An axiomatic approach. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 199–208, 2008.

[20] R. Andersen, C. Borgs, J. Chayes, J. Hopcroft, K. Jain, V. Mirrokni, and S. Teng. Robust PageRank and locally computable spam detection features. In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web*, AIRWeb '08, pages 69–76, 2008.

[21] R. Andersen, C. Borgs, J. T. Chayes, J. E. Hopcroft, V. S. Mirrokni, and S.-H. Teng. Local computation of PageRank contributions. *Internet Mathematics*, 5(1):23–45, 2008.

[22] R. Andersen, F. Chung, and K. Lang. Using PageRank to locally partition a graph. *Internet Mathematics.*, 4(1):1–128, 2007.

[23] R. Andersen, S. O. Gharan, Y. Peres, and L. Trevisan. Almost optimal local graph clustering using evolving sets. *Journal of the ACM*, 63:Article No. 15, 2015.

[24] R. Andersen and Y. Peres. Finding sparse cuts locally using evolving sets. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, STOC '09, pages 235–244, 2009.

[25] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, Jan. 2008.

[26] S. Arora, A. Bhaskara, R. Ge, and T. Ma. Provable bounds for learning some deep representations. In *Proceedings of the International Conference on Machine Learning*, ICML'14, pages 584–592, 2014.

[27] S. Arora, R. Ge, Y. Halpern, D. M. Mimno, A. Moitra, D. Sontag, Y. Wu, and M. Zhu. A practical algorithm for topic modeling with provable guarantees. In *Proceedings of the International Conference on Machine Learning*, ICML"13, pages 280–288, 2013.

[28] S. Arora, R. Ge, T. Ma, and A. Moitra. Simple, efficient, and neural algorithms for sparse coding. *CoRR*, abs/1503.00778, 2015.

[29] S. Arora, R. Ge, A. Moitra, and S. Sachdeva. Provable ICA with unknown Gaussian noise, and implications for Gaussian mixtures and autoencoders. *Algorithmica*, 72(1):215–236, May 2015.

[30] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012.

[31] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, May 1998.

[32] K. Arrow. A difficulty in the concept of social welfare. *Journal of Political Economy*, 58, 1950.

[33] K. J. Arrow. *Social Choice and Individual Values*. Wiley, New York, 2nd edition, 1963.

[34] D. Arthur, B. Manthey, and H. Röglin. Smoothed analysis of the k-means method. *Journal of the ACM*, 58(5):19:1–19:31, Oct. 2011.

[35] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Racke. Optimal oblivious routing in polynomial time. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, STOC '03, pages 383–388, 2003.

[36] L. Babai. Graph isomorphism in quasipolynomial time. *CoRR*, abs/1512.03547, 2015.

[37] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, STOC '91, pages 21–32, 1991.

[38] M.-F. Balcan, A. Blum, and A. Gupta. Approximate clustering without the approximation. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 1068–1077, 2009.

[39] M. F. Balcan, C. Borgs, M. Braverman, J. T. Chayes, and S.-H. Teng. Finding endogenously formed communities. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 767–783, 2013.

[40] M.-F. Balcan and M. Braverman. Finding low error clusterings. In *Conference on Learning Theory*, 2009.

[41] J. F. Banzhaf. Weighted voting doesn't work: A mathematical analysis. *Rutgers Law Review*, 19:317–343, 1965.

[42] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[43] J. Batson, D. A. Spielman, N. Srivastava, and S.-H. Teng. Spectral sparsification of graphs: Theory and algorithms. *Communications of the ACM*, 56(8):87–94, Aug. 2013.

[44] J. D. Batson, D. A. Spielman, and N. Srivastava. Twice-Ramanujan sparsifiers. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, STOC, pages 255–262, 2009.

[45] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, 1st edition, 1998.

[46] A. Bavelas. Communication patterns in task oriented groups. *Journal of the Acoustical Society of America*, 22(6):725–730, Nov. 1950.

[47] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, June 2003.

[48] P. Belleflamme. Stable coalition structures with open membership and asymmetric firms. *Games and Economic Behavior*, 30(1):1–21, 2000.

[49] A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, STOC'96, pages 47–55, New York, NY, USA, 1996. ACM.

[50] M. Bern, D. Eppstein, and J. R. Gilbert. Provably good mesh generation. In *The 31st Annual Symposium on Foundations of Computer Science*, pages 231–241. IEEE, 1990.

[51] M. Bern, D. Eppstein, and S.-H. Teng. Parallel construction of quadtrees and quality triangulations. *International Journal of Computational Geometry & Applications*, 9(06):517–532, 1999.

[52] P. Biswal, J. R. Lee, and S. Rao. Eigenvalue bounds, spectral partitioning, and metrical deformations via flows. *Journal of the ACM*, 57(3):13:1–13:23, Mar. 2010.

[53] A. Blum and J. Spencer. Coloring random and semi-random k-colorable graphs. *Journal of Algorithms*, 19(2):204–234, 1995.

[54] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bulletin of the American Mathematical Society, N. S.*, 21(1):1–46, Jul. 1989.

[55] B. Bollobás. *Modern graph theory*. Springer-Verlag, 1998.

[56] E. G. Boman, D. Chen, O. Parekh, and S. Toledo. On factor width and symmetric H-matrices. *Linear Algebra and Its Applications*, 405:239–248, 2005.

[57] E. G. Boman, B. Hendrickson, and S. Vavasis. Solving elliptic finite element systems in near-linear time with support preconditioners. *SIAM Journal Numerical Analysis*, 46(6):3264–3284, Oct. 2008.

[58] P. Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, 92(5):1170–1182, 1987.

[59] P. Bonacich. Simultaneous group and individual centralities. *Social Networks*, 13(2):155 – 168, 1991.

[60] O. N. Bondareva. Some applications of the methods of linear programming to the theory of cooperative games. *Problemy Kibernet.*, 10:119–139, 1963.

[61] R. B. Boppana. Eigenvalues and graph bisection: An average-case analysis. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 280–285, 1987.

[62] S. P. Borgatti. Centrality and network flow. *Social Networks*, 27(1):55–71, 2005.

[63] S. P. Borgatti and M. G. Everett. A graph-theoretic perspective on centrality. *Social Networks*, 28(4):466–484, 2006.

[64] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 946–957, 2014.

[65] C. Borgs, M. Brautbar, J. Chayes, and S.-H. Teng. Multi-scale matrix sampling and sublinear-time PageRank computation. *Internet Mathematics*, 10(1-2):20–48, 2014.

[66] C. Borgs, J. T. Chayes, A. Marple, and S. Teng. An axiomatic approach to community detection. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA*, pages 135–146, 2016.

[67] S. J. Brams, M. A. Jones, and D. M. Kilgour. Dynamic models of coalition formation: Fallback vs. build-up. In *Proceedings of the 9th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK '03, pages 187–200, 2003.

[68] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 31(138):333–390, Apr. 1977.

[69] A. Brandt. *Multilevel computations: Review and recent developments, edited by McCormick, S.F.*, pages 35–62. Marcel-Dekker, 1988.

[70] M. Braverman, Y. K. Ko, A. Rubinstein, and O. Weinstein. ETH hardness for densest-k-subgraph with perfect completeness. *CoRR*, arXiv:1504.08352, 2015.

[71] R. Brent. The parallel evaluation of general arithmetic expressions. *Journal of the ACM*, 21(2):201–208, Apr. 1974.

[72] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial (2Nd Ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[73] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

[74] A. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences*, SEQUENCES '97, pages 21–29, 1997.

[75] L. E. J. Brouwer. Ü ber Abbildung von Mannigfaltigkeitn. *Math. Annale*, 71:97–115, 1912.

[76] Z. Burda, J. Duda, J. M. Luck, and B. Waclaw. Localization of the maximal entropy random walk. *Physical Review Letters*, 102(16):160602, Apr. 2009.

[77] M. Caesar and J. Rexford. BGP routing policies in ISP networks. *Netwrk. Mag. of Global Internetwkg.*, 19(6):5–11, Nov. 2005.

[78] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, Jan. 1995.

[79] D. Chakrabarti and C. Faloutsos. *Graph mining: laws, tools, and case studies*. Synthesis lectures on data mining and knowledge discovery. Morgan & Claypool, 2012.

[80] T. M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete & Computational Geometry*, 16:361–368, 1996.

[81] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, STOC '02, pages 380–388, 2002.

[82] J. Cheeger. A lower bound for the smallest eigenvalue of the Laplacian. In *Problems in Analysis, Edited by R. C. Gunning*, pages 195–199. Princeton University Press, 1970.

[83] W. Chen and S.-H. Teng. Interplay between influence processes and social networks II: Clusterability and personalized Shapley values. 2016.

[84] W. Chen and S.-H. Teng. Interplay between social influence and network centrality: Shapley values and scalable algorithms. *CoRR*, abs/1602.03780, 2016.

[85] X. Chen and S. Teng. A complexity view of markets with social influence. In *Proceedings in Innovations in Computer Science - Tsinghua University, Beijing, China*, ICS, pages 141–154, 2011.

[86] D. Cheng, Y. Cheng, Y. Liu, R. Peng, and S.-H. Teng. Efficient sampling for Gaussian graphical models via spectral sparsification. In *Proceedings of the 28th Conference on Learning Theory*, COLT '05, 2015.

[87] S.-W. Cheng, T. K. Dey, H. Edelsbrunner, M. A. Facello, and S.-H. Teng. Silver exudation. *Journal of the ACM*, 47(5):883–904, Sept. 2000.

[88] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[89] P. Chew. There are planar graphs almost as good as the complete graph. *JCSS*, 39:205–219, 1989.

[90] P. Chin, A. Rao, and V. Vu. Stochastic block model and community detection in sparse graphs: A spectral algorithm with optimal rate of recovery. In *Proceedings of The 28th Conference on Learning Theory*, pages 391–423, 2015.

[91] P. Christiano, J. A. Kelner, A. Mądry, D. A. Spielman, and S.-H. Teng. Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, pages 273–282, 2011.

[92] F. Chung. The heat kernel as the PageRank of a graph. *Proceedings of the National Academy of Sciences*, 104(50):19735–19740, Dec. 2007.

[93] F. Chung. A local graph partitioning algorithm using heat kernel Pagerank. *Internet Mathematics*, 6(3):315–330, Jan. 2009.

[94] F. Chung and L. Lu. *Complex Graphs and Networks (CBMS Regional Conference Series in Mathematics)*. American Mathematical Society, Boston, MA, USA, 2006.

[95] F. R. K. Chung. *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*. American Mathematical Society, Feb. 1997.

[96] K. Clarkson, D. Eppstein, G. L. Miller, C. Sturtivant, and S.-H. Teng. Approximating center points with and without linear programming. In *Proceedings of 9th ACM Symposium on Computational Geometry*, pages 91–98, 1993.

[97] M. B. Cohen, R. Kyng, G. L. Miller, J. W. Pachocki, R. Peng, A. B. Rao, and S. C. Xu. Solving sdd linear systems in nearly $m \log^{1/2} n$ time. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC, pages 343–352, 2014.

[98] M. B. Cohen, Y. T. Lee, G. L. Miller, J. W. Pachocki, and A. Sidford. Geometric median in nearly linear time. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, STOC, 2016.

[99] J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices and Groups*. Springer-Verlag, 1988.

[100] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158. ACM, 1971.

[101] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19:297–301, 1965.

[102] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, Mar. 1990.

[103] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

[104] D. Johnson and C.H. Papadimitriou and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37(1):79–100, 1988.

[105] S. I. Daitch and D. A. Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 451–460, 2008.

[106] L. Danzer, J. Fonlupt, and V. Klee. Helly's theorem and its relatives. *Proceedings of Symposia in Pure Mathematics, American Mathematical Society*, 7:101–180, 1963.

[107] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th Annual Symposium on Computational Geometry*, SCG '04, pages 253–262, 2004.

[108] I. Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[109] E. David and K. Jon. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, New York, NY, USA, 2010.

[110] P. Debevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and M. Sagar. Acquiring the reflectance field of a human face. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 145–156, 2000.

[111] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[112] X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Math. Oper. Res.*, 19(2):257–266, May 1994.

[113] J. Ding, J. R. Lee, and Y. Peres. Cover times, blanket times, and majorizing measures. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, STOC '11, pages 61–70, 2011.

[114] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 57–66, 2001.

[115] W. E. Donath and A. J. Hoffman. Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices. *IBM Technical Disclosure Bulletin*, 15:938 – 944, 1972.

[116] W. E. Donath and A. J. Hoffman. Lower bounds for the partitioning of graphs. *Journal Research Developments*, 17:420 – 425, 1973.

[117] L. Donetti, P. I. Hurtado, and M. A. Munoz. Entangled networks, synchronization, and optimal network topology. *CoRR*, Oct. 2005.

[118] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, Apr. 2006.

[119] P. G. Doyle and J. L. Snell. *Random Walks and Electric Networks*. Mathematical Association of America, 1984.

[120] J. Dunagan, D. A. Spielman, and S.-H. Teng. Smoothed analysis of condition numbers and complexity implications for linear programming. *Math. Program.*, 126(2):315–350, Feb. 2011.

[121] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Conference on Theory of Cryptography*, TCC'06, pages 265–284, 2006.

[122] J. Edmonds. Maximum matching and a polyhedron with $0,1$ vertices. *Journal of Research at the National Bureau of Standards*, 69 B:125–130, 1965.

[123] Y. Eidelman and I. Gohberg. Inversion formulas and linear complexity algorithm for diagonal plus semiseparable matrices. *Computers & Mathematics with Applications*, 33(4):69 – 79, 1997.

[124] P. Elias, A. Feinstein, and C. E. Shannon. A note on the maximum flow through a network. *IRE Transactions on Information Theory*, 2, 1956.

[125] M. Elkin, Y. Emek, D. A. Spielman, and S.-H. Teng. Lower-stretch spanning trees. *SIAM Journal on Computing*, 32(2):608–628, 2008.

[126] S. Even and R. E. Tarjan. Network flow and testing graph connectivity. *SIAM Journal on Computing*, 4(4):507–518, 1975.

[127] K. Faust. Centrality in affiliation networks. *Social Networks*, 19(2):157–191, Apr. 1997.

[128] U. Feige, M. Feldman, N. Immorlica, R. Izsak, B. Lucier, and V. Syrgkanis. A unifying hierarchy of valuations with complements and substitutes. *CoRR*, abs/1408.1211, 2014.

[129] U. Feige, N. Immorlica, V. S. Mirrokni, and H. Nazerzadeh. Pass approximation: A framework for analyzing and designing heuristics. *Algorithmica*, 66(2):450–478, 2013.

[130] U. Feige and R. Izsak. Welfare maximization and the supermodular degree. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 247–256, 2013.

[131] U. Feige and J. Kilian. Heuristics for finding large independent sets, with applications to coloring semi-random graphs. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, page 674, 1998.

[132] U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM Journal on Computing*, 31:1090–1118, 2002.

[133] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298 – 305, 1973.

[134] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its applications to graph theory. *Czechoslovak Mathematical Journal*, 25(100):619 – 633, 1975.

[135] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[136] J. G. F. Francis. The QR transformation a unitary analogue to the LR transformation: Part 1. *The Computer Journal*, 4(3):256 – 271, 1961.

[137] L. C. Freeman. A set of measures of centrality based upon betweenness. *Sociometry*, 40:35–41, 1977.

[138] L. C. Freeman. Centrality in social networks: Conceptual clarification. *Social Networks*, 1(3):215–239, 1979.

[139] A. M. Frieze, G. L. Miller, and S.-H. Teng. Separator based parallel divide and conquer in computational geometry. In *Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '92, pages 420–429, 1992.

[140] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[141] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[142] H. Gazit and G. L. Miller. Planar separators and the Euclidean norm. In *Proceedings of the International Symposium on Algorithms*, SIGAL '90, pages 338–347, 1990.

[143] R. Ge. *Provable algorithms for machine learning problems*. PhD thesis, Princeton University, 2013.

[144] A. George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10(2):345–363, 1973.

[145] R. Ghosh and K. Lerman. Predicting influential users in online social networks. In *Proceedings of KDD workshop on Social Network Analysis (SNAKDD)*, May 2010.

[146] R. Ghosh, S.-H. Teng, K. Lerman, and X. Yan. The interplay between dynamics and networks: Centrality, communities, and Cheeger inequality. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1406–1415, 2014.

[147] J. R. Gilbert, J. P. Hutchinson, and R. E. Tarjan. A separator theorem for graphs of bounded genus. *Journal of Algorithms*, 5(3):391–407, Sept. 1984.

[148] J. R. Gilbert, G. L. Miller, and S.-H. Teng. Geometric mesh partitioning: Implementation and experiments. *SIAM Journal on Scientific Computing*, 19(6):2091–2110, Nov. 1998.

[149] J. R. Gilbert, C. Moler, and R. Schreiber. Sparse matrices in Matlab: design and implementation. *SIAM Journal on Matrix Analysis and Applications*, 13(1):333–356, Jan. 1992.

[150] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529, 1999.

[151] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, Nov. 1995.

[152] I. Gohberg, T. Kailath, and I. Koltracht. Linear complexity algorithms for semiseparable matrices. *Integral Equations and Operator Theory*, 8(6):780–804, 1985.

[153] O. Goldreich. Introduction to testing graph properties. In O. Goldreich, editor, *Property Testing*, pages 105–141. Springer-Verlag, Berlin, Heidelberg, 2010.

[154] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, July 1998.

[155] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins Press, 2nd edition, 1989.

[156] D. Gómez, E. Gonzalez-Arangüena, C. Manuel, G. Owen, M. del Pozo, and J. Tejada. Centrality and power in social networks: a game theoretic approach. *Mathematical Social Sciences*, 46(1):27–54, 2003.

[157] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, Dec. 1987.

[158] L. F. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems*. PhD thesis, Yale University, New Haven, CT, USA, 1987.

[159] B. Grofman and G. Owen. A game theoretic approach to measuring degree of centrality in social networks. *Social Networks*, 4(3):213 – 224, 1982.

[160] D. Gusfield and R. W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, Cambridge, MA, USA, 1989.

[161] W. Hackbusch. A sparse matrix arithmetic based on H-matrices. part I: Introduction to H-matrices. *Computing*, 62(2):89–108, 1999.

[162] K. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17:219 – 229, 1970.

[163] J. Hammersley and P. Clifford. *Markov fields on finite graphs and lattices*. Unpublished manuscript, 1971.

[164] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[165] S. Hanneke and E. P. Xing. Network completion and survey sampling. In D. A. V. Dyk and M. Welling, editors, *AISTATS*, volume 5 of *JMLR Proceedings*, pages 209–215, 2009.

[166] S. Hart and M. Kurz. Endogenous formation of coalitions. *Econometrica*, 51(4):1047–1064, 1983.

[167] J. Hartmanis and R. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285 – 306, 1965.

[168] T. Haveliwala. Topic-sensitive Pagerank: A context-sensitive ranking algorithm for web search. In *IEEE Transactions on Knowledge and Data Engineering*, volume 15(4), pages 784–âĂŞ796, 2003.

[169] K. He, S. Soundarajan, X. Cao, J. E. Hopcroft, and M. Huang. Revealing multiple layers of hidden community structure in networks. *CoRR*, abs/1501.05700, 2015.

[170] M. T. Heath and W. A. Dick. Virtual prototyping of solid propellant rockets. *Computing in Science and Engineering*, 2(2):21–32, Mar. 2000.

[171] J. Heinonen. *Lectures on analysis on metric spaces*. Universitext. Springer, 2001.

[172] M. Hubert and P. J. Rousseeuw. The catline for deep regression. *J. Multivariate Analysis*, 66:270–296, 1998.

[173] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the Institute of Radio Engineers*, 40(9):1098–1101, 1952.

[174] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, 1998.

[175] M. O. Jackson. *Social and Economic Networks*. Princeton University Press, Princeton, NJ, USA, 2008.

[176] R. A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2(1):18–21, 1973.

[177] G. Jeh and J. Widom. Scaling personalized Web search. In *WWW*, pages 271–279, 2003.

[178] M. Jerrum and A. Sinclair. Conductance and the rapid mixing property for Markov chains: the approximation of permanent resolved. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, STOC, pages 235–244. ACM, 1988.

[179] M. Johnson, J. Saunderson, and A. Willsky. Analyzing hogwild parallel Gaussian Gibbs sampling. In *Advances in Neural Information Processing Systems 26*, pages 2715–2723. Curran Associates, Inc., 2013.

[180] W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26(1-1.1):189–206, 1984.

[181] M. I. Jordan. *Learning in Graphical Models*. The MIT press, 1998.

[182] M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[183] U. Kang, D. H. Chau, and C. Faloutsos. Mining large graphs : Algorithms , inference , and discoveries. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, ICDE '11, pages 243—-254, 2011.

[184] U. Kang and C. Faloutsos. Big graph mining: Algorithms and discoveries. *SIGKDD Explorations Newsletter*, 14(2):29–36, Apr. 2013.

[185] R. Kannan and S. Vempala. Spectral algorithms. *Foundations and Trends® in Theoretical Computer Science*, 4(3-4):157–288, 2009.

[186] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM*, 51(3):497–515, May 2004.

[187] D. R. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, STOC '02, pages 741–750, 2002.

[188] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, STOC '84, pages 302–311, 1984.

[189] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[190] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1), Dec. 1998.

[191] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, Mar. 1953.

[192] M. J. Kearns, M. L. Littman, and S. P. Singh. Graphical models for game theory. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI '01, pages 253–260, 2001.

[193] M. J. Kearns, M. L. Littman, and S. P. Singh. Graphical models for game theory. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, pages 253–260, 2001.

[194] M. J. Kearns, R. E. Schapire, and L. M. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2-3):115–141, Nov. 1994.

[195] J. Kelner, J. R. Lee, G. Price, , and S.-H. Teng. Metric uniformization and spectral bounds for graphs. *Geom. Funct. Anal.*, 21(5):1117–1143, 2011.

[196] J. A. Kelner. Spectral partitioning, eigenvalue bounds, and circle packings for graphs of bounded genus. *SIAM Journal on Computing.*, 35(4):882–902, Apr. 2006.

[197] J. A. Kelner, Y. T. Lee, L. Orecchia, and A. Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'14, pages 217–226, 2014.

[198] J. A. Kelner and A. Levin. Spectral sparsification in the semi-streaming setting. *Theory of Computing Systems*, 53(2):243–262, 2013.

[199] J. A. Kelner and A. Mądry. Faster generation of random spanning trees. In *Proceedings of the 2009 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, pages 13–21, 2009.

[200] J. A. Kelner, L. Orecchia, A. Sidford, and Z. A. Zhu. A simple, combinatorial algorithm for solving SDD systems in nearly-linear time. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, pages 911–920, 2013.

[201] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD '03*, pages 137–146. ACM, 2003.

[202] L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244:1093–1096, 1979.

[203] M. Kim and J. Leskovec. The network completion problem: Inferring missing nodes and edges in networks. In *SDM*, pages 47–58. SIAM / Omnipress, 2011.

[204] D. G. Kirkpatrick and R. Seidel. The ultimate planar convex hull algorithm. *SIAM Journal on Computing.*, 15(1):287–299, Feb. 1986.

[205] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[206] M. Kiwi, D. A. Spielman, and S.-H. Teng. Min-max-boundary domain decomposition. *Theoretical Computer Science*, 261:2001, 1998.

[207] J. Kleinberg. An impossibility theorem for clustering. In *NIPS*, pages 463–470, 2002.

[208] J. Kleinberg, C. Papadimitriou, and P. Raghavan. Segmentation problems. *Journal of the ACM*, 51(2):263–280, Mar. 2004.

[209] J. Kleinberg and E. Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[210] J. Kleinberg and E. Tardos. Balanced outcomes in social exchange networks. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC '08, pages 295–304, 2008.

[211] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, Sept. 1999.

[212] R. Kleinberg. Geographic routing using hyperbolic space. In *Proceedings of the 26th Annual Joint Conference of the IEEE Computer and Communications Societies*, INFOCOM 2007, pages 1902–1909, 2007.

[213] P. Koebe. Kontaktprobleme der konformen abbildung. *Ber. Sächs. Akad. Wiss. Leipzig, Math.-Phys. Kl.*, 88:141âĂŞ164, 1936.

[214] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computa tion and Machine Learning*. The MIT Press, 2009.

[215] I. Koutis, G. Miller, and R. Peng. A nearly-mlogn time solver for SDD linear systems. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science*, FOCS, pages 590–598, 2011.

[216] I. Koutis, G. L. Miller, and R. Peng. Approaching optimality for solving SDD systems. In *Proceedings of the 51st Annual Symposium on Foundations of Computer Science*, FOCS, pages 235–244, 2010.

[217] R. Kyng and S. Sachdeva. Approximate gaussian elimination for laplacians: Fast, sparse, and simple. *CoRR*, abs/1605.02353, 2016.

[218] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 609–616, 2009.

[219] Y. T. Lee and A. Sidford. Following the path of least resistance : An Õ(m sqrt(n)) algorithm for the minimum cost flow problem. *CoRR*, abs/1312.6713, 2013.

[220] Y. T. Lee and A. Sidford. Matching the universal barrier without paying the costs : Solving linear programs with Õ(sqrt(rank)) linear system solves. *CoRR*, abs/1312.6677, 2013.

[221] Y. T. Lee and H. Sun. Constructing linear-sized spectral sparsification in almost-linear time. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 250–269, Oct 2015.

[222] B. Lehmann, D. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. In *Proceedings of the 3rd ACM Conference on Electronic Commerce*, EC '01, pages 18–28, 2001.

[223] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Array, Trees, Hypercubes*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.

[224] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6):787–832, Nov. 1999.

[225] C. Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Transactions on Computers*, 34(10):892–901, Oct. 1985.

[226] C. E. Leiserson. *Area-Efficient VLSI Computation*. MIT Press, Cambridge, MA, USA, 1997.

[227] K. Lerman and R. Ghosh. Network structure, topology and dynamics in generalized models of synchronization. *Physical Review E*, 86(026108), 2012.

[228] K. Lerman, S.-H. Teng, and X. Yan. Network composition from multilayer data. USC Information Sciences Institute.

[229] J. Leskovec. *Dynamics of Large Networks*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2008. AAI3340652.

[230] J. Leskovec and C. Faloutsos. Scalable modeling of real graphs using kronecker multiplication. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 497–504, 2007.

[231] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, pages 29–123, 2009.

[232] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of 17th International World Wide Web Conference (WWW2008)*, 2008.

[233] L. Levin. Universal sorting problems. *Problems of Information Transmission*, 9:265 – 266, 1973.

[234] X. Y. Li and S.-H. Teng. Sliver-free three dimensional Delaunay mesh generation. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 28–37, 2001.

[235] R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM J. on Numerical Analysis*, 16, 1979.

[236] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36:177–189, 1979.

[237] Y. Liu, O. Kosut, and A. S. Willsky. Sampling from Gaussian graphical models using subgraph perturbations. In *Proceedings of the 2013 IEEE International Symposium on Information Theory, Istanbul, Turkey, Jul. 7-12, 2013*, 2013.

[238] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, Sept. 2006.

[239] L. Lovász and M. Simonovits. The mixing rate of Markov chains, an isoperimetric inequality, and computing the volume. In *Proceedings: 31st Annual Symposium on Foundations of Computer Science*, pages 346–354, 1990.

[240] L. Lovász and M. Simonovits. Random walks in a convex body and an improved volume algorithm. *RSA: Random Structures & Algorithms*, 4:359–412, 1993.

[241] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[242] A. Mądry. Fast approximation algorithms for cut-based problems in undirected graphs. In *FOCS'10: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*, pages 245–254, 2010.

[243] A. Mądry. Navigating central path with electrical flows: from flows to matchings, and back. In *Proceedings of the 54th Annual Symposium on Foundations of Computer Science*, FOCS, pages 253–262, 2013.

[244] G. A. Margulis. Explicit group theoretical constructions of combinatorial schemes and their application to the design of expanders and concentrators. *Problems of Information Transmission*, 24(1):39–46, Jul. 1988.

[245] F. Masrour, I. Barjesteh, R. Forsati, A.-H. Esfahanian, and H. Radha. Network completion with node similarity: A matrix completion approach with provable guarantees. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '15, pages 302–307. ACM, 2015.

[246] T. Mattson, D. A. Bader, J. W. Berry, A. Buluç, J. J. Dongarra, C. Faloutsos, J. Feo, J. R. Gilbert, J. Gonzalez, B. Hendrickson, J. Kepner, C. E. Leiserson, A. Lumsdaine, D. A. Padua, S. W. Poole, S. P. Reinhardt, M. Stonebraker, S. Wallach, and A. Yoo. Standards for graph algorithm primitives. *CoRR*, abs/1408.0393, 2014.

[247] N. Megiddo. Linear programming in linear time when the dimension is fixed. *Journal of the ACM*, 31(1):114–127, Jan. 1984.

[248] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[249] T. P. Michalak, K. V. Aadithya, P. L. Szczepanski, B. Ravindran, and N. R. Jennings. Efficient computation of the Shapley value for game-theoretic network centrality. *J. Artif. Int. Res.*, 46(1):607–650, Jan. 2013.

[250] G. L. Miller. Riemann's hypotheis and tests for primality. *Journal of Computer and System Sciences*, 13(3):300–317, Dec. 1976.

[251] G. L. Miller, D. Talmor, S.-H. Teng, and N. Walkington. A Delaunay based numerical method for three dimensions: Generation, formulation, and partition. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, STOC '95, pages 683–692, 1995.

[252] G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Separators for sphere-packings and nearest neighbor graphs. *Journal of the ACM*, 44(1):1–29, Jan. 1997.

[253] G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Geometric separators for finite-element meshes. *SIAM Journal on Scientific Computing*, 19(2):364–386, Mar. 1998.

[254] N. Mishra, R. Schreiber, I. Stanton, and R. Tarjan. Finding strongly-knit clusters in social networks. *Internet Mathematics*, pages 155–174, 2009.

[255] S. A. Mitchell and S. A. Vavasis. Quality mesh generation in three dimensions. In *Proceedings of the ACM Computational Geometry Conference*, pages 212–221, 1992. Also appeared as Cornell C.S. TR 92-1267.

[256] O. Morgenstern and J. von Neumann. *Theory of Games and Economic Behavior*. Princeton University Press, 1947.

[257] R. Narayanam and Y. Narahari. A Shapley value-based approach to discover influential nodes in social networks. *Automation Science and Engineering, IEEE Transactions on*, PP(99):1 –18, 2010.

[258] J. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of the USA*, 36(1):48–49, 1950.

[259] J. Nash. Noncooperative games. *Annals of Mathematics*, 54:289–295, 1951.

[260] M. Newman. *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010.

[261] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 2006.

[262] N. Nisan and A. Ronen. Algorithmic mechanism design (extended abstract). In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, STOC '99, pages 129–140, 1999.

[263] G. Niu, B. Recht, C. Re, and S. J. Wright. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Conference on Neural Information Processing Systems*, NIPS, 2011.

[264] E. Nygren, R. K. Sitaraman, and J. Sun. The Akamai network: A platform for high-performance Internet applications. *SIGOPS Operating Systems Review*, 44(3):2–19, Aug. 2010.

[265] L. Orecchia, S. Sachdeva, and N. K. Vishnoi. Approximating the exponential, the lanczos method and an $\tilde{O}(m)$-time spectral algorithm for balanced separator. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, STOC '12, pages 1141–1160, 2012.

[266] L. Page, S. Brin, R. Motwani, and T. Winograd. The Pagerank citation ranking: Bringing order to the Web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, 1998.

[267] I. Palacios-Huerta and O. Volij. The measurement of intellectual influence. *Econometrica*, 72:963–977, 2004.

[268] C. Papadimitriou. On graph-theoretic lemmata and complexity classes. In *Proceedings 31st Annual Symposium on Foundations of Computer Science*, pages 794–801, 1990.

[269] C. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, pages 498–532, 1994.

[270] C. Papadimitriou. Algorithms, games, and the internet. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, STOC, pages 749–753, 2001.

[271] C. Papadimitriou, A. Schaffer, and M. Yannakakis. On the complexity of local search. In *Proceedings of the 22th Annual ACM Symposium on Theory of Computing*, pages 438–445, 1990.

[272] C. H. Papadimitriou and M. Yannakakis. Multiobjective query optimization. In *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '01, pages 52–59, 2001.

[273] D. Peleg and A. A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13:99–116, 1989.

[274] D. Peleg and J. D. Ullman. An optimal synchronizer for the hypercube. *SIAM Journal on Computing.*, 18(4):740–747, Aug. 1989.

[275] R. Peng. Approximate undirected maximum flows in O(mpolylog(n)) time. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 1862–1867, 2016.

[276] L. S. Penrose. The elementary statistics of majority voting. *Journal of the Royal Statistical Society*, 109(1):53–57, 1946.

[277] M. Piraveenan, M. Prokopenko, and L. Hossain. Percolation centrality: Quantifying graph-theoretic impact of nodes during percolation in networks. *PLoS ONE*, 8(1), 2013.

[278] A. Pothen, H. D. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, May 1990.

[279] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[280] M. O. Rabin. *Probabilistic Algorithms*. Academic Press, New York, 1976.

[281] H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC '08, pages 255–264, 2008.

[282] H. Räcke, C. Shah, and H. Täubig. Computing cut-based hierarchical decompositions in almost linear time. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '14, pages 227–238, 2014.

[283] D. Ray. *A game-theoretic perspective on coalition formation*. The Lipsey lectures. Oxford University Press, Oxford, 2007.

[284] Y. Rekhter and T.Li. A Border Gateway Protocol 4. *RFC 1771, Internet Engineering Task Force*, 1995.

[285] J. Renegar. Incorporating condition measures into the complexity theory of linear programming. *SIAM Journal on Optimization*, 5:5–3, 1995.

[286] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 61–70, 2002.

[287] H. Röglin and S.-H. Teng. Smoothed analysis of multiobjective optimization. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '09, pages 681–690, 2009.

[288] D. Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends® in Theoretical Computer Science*, 5(2):73–205, 2010.

[289] A. E. Roth. The evolution of the labor market for medical interns and residents: A case study in game theory. *Journal of Political Economy*, 92:991–1016, 1984.

[290] A. E. Roth. *Stable Coalition Formation: Aspects of a Dynamic Theory*, pages 228–234. Wuerzberg: Physica-Verlag, 1984.

[291] R. Rubinfeld and A. Shapira. Sublinear time algorithms. *SIAM Journal on Discrete Mathematics*, 25(4):1562–1588, Nov. 2011.

[292] M. Rudelson. Random vectors in the isotropic position. *Journal of Functional Analysis*, 164(1):60 – 72, 1999.

[293] J. Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. In *The 4th ACM-SIAM Symposium on Discrete Algorithms*, pages 83–92. SIAM, 1993.

[294] G. Sabidussi. The centrality index of a graph. *Psychometirka*, 31:581–606, 1996.

[295] M. Santha and U. V. Vazirani. Generating quasi-random sequences from semi-random sources. *Journal of Computer and System Sciences*, 33(1):75–87, 1986.

[296] H. E. Scarf. The core of an N person game. *Econometrica*, 69:35–50, 1967.

[297] A. Schrijver. *Combinatorial Optimization, Volume A*. Number 24 in Algorithms and Combinatorics. Springer, 2003.

[298] R. Seidel. Constructing higher-dimensional convex hulls at logarithmic cost per face. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, STOC '86, pages 404–413, 1986.

[299] R. Seidel. Linear programming and convex hulls made easy. In *Proceedings of the 6th Annual Symposium on Computational Geometry*, SCG '90, pages 211–215, 1990.

[300] L. S. Shapley. A value for n-person games. In H. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games II*, pages 307–317. Princeton University Press, 1953.

[301] L. S. Shapley. On balanced sets and cores. *Naval Research Logistics*, 14, 1967.

[302] L. S. Shapley. Cores of convex games. *International Journal of Game Theory*, 1(1):11 – 26, 1971.

[303] J. Sherman. Nearly maximum flows in nearly linear time. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*, FOCS, pages 263–269, 2013.

[304] J. R. Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1-3):21–74, May 2002.

[305] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[306] H. D. Simon and S.-H. Teng. How good is recursive bisection? *SIAM Journal on Scientific Computing*, 18(5):1436–1445, Sept. 1997.

[307] M. Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1st edition, 1996.

[308] L. M. Smith, K. Lerman, C. Garcia-Cardona, A. G. Percus, and R. Ghosh. Spectral clustering with epidemic diffusion. *Physical Review E*, 88(4):042813, Oct. 2013.

[309] R. M. Solovay and V. Strassen. A fast Monte-Carlo test for primality. *SIAM Journal on Computing*, 6:84–85, 1977.

[310] E. Sperner. Neuer Beweis fur die Invarianz der Dimensionszahl und des Gebietes. *Abhandlungen aus dem Mathematischen Seminar Universitat Hamburg*, 6:265–272, 1928.

[311] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. In *Proceedings of the 40th ACM Symposium on the Theory of Computing*, pages 563–568, 2008.

[312] D. A. Spielman and S.-H. Teng. Disk packings and planar separators. In *Proceedings of the 12th Annual Symposium on Computational Geometry*, SCG '96, pages 349–358, 1996.

[313] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, STOC '04, pages 81–90, 2004.

[314] D. A. Spielman and S.-H. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

[315] D. A. Spielman and S.-H. Teng. Spectral partitioning works: Planar graphs and finite element meshes. *Linear Algebra and its Applications*, 421(2-3):284–305, mar 2007.

[316] D. A. Spielman and S.-H. Teng. Smoothed analysis: an attempt to explain the behavior of algorithms in practice. *Communications of the ACM*, 52(10):76–84, Oct. 2009.

[317] D. A. Spielman and S.-H. Teng. Spectral sparsification of graphs. *SIAM Journal on Computing.*, 40(4):981–1025, July 2011.

[318] D. A. Spielman and S.-H. Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on Computing.*, 42(1):1–26, 2013.

[319] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014.

[320] G. Strang and G. Fix. *An Analysis of the Finite Element Method*. Wiley, 2nd edition, 2008.

[321] J. Suomela. Survey of local algorithms. *ACM Computing Surveys*, 45(2):24:1–24:40, Feb. 2013.

[322] N. R. Suri and Y. Narahari. Determining the top-k nodes in social networks using the Shapley value. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '08, pages 1509–1512, 2008.

[323] P. L. Szczepański, T. Michalak, and T. Rahwan. A new approach to betweenness centrality based on the Shapley value. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pages 239–246, 2012.

[324] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1539–1554, 2015.

[325] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: near-optimal time complexity meets practical efficiency. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, SIGMOD, pages 75–86, 2014.

[326] S.-H. Teng. Combinatorial aspects of geometric graphs. *Computational Geometry: Theory and Applications*, 9(4):277–287, Mar. 1998.

[327] S.-H. Teng. Provably good partitioning and load balancing algorithms for parallel adaptive n-body simulation. *SIAM Journal on Scientific Computing*, 19(2):635–656, Mar. 1998.

[328] S.-H. Teng. *Coarsening, Sampling, and Smoothing: Elements of the Multilevel Method*, pages 247–276. Springer, New York, 1999.

[329] S.-H. Teng. The Laplacian Paradigm: Emerging algorithms for massive graphs. In *Proceedings of the 7th Annual Conference on Theory and Applications of Models of Computation*, TAMC'10, pages 2–14, Berlin, Heidelberg, 2010. Springer-Verlag.

[330] S.-H. Teng. Numerical thinking in algorithm design and analysis. In *Computer Science: The Hardware, Software and Heart of It, ed by Blum, K. Edward and Aho, V. Alfred*, pages 349–384. Springer New York, 2011.

[331] S.-H. Teng. Network essence: Pagerank completion and centrality-conforming Markov Chains. In M. Loebl, J. Nesetril, and R. Thomas, editors, *Journey Through Discrete Mathematis: A Tribute to Jiří Matoušek*. 2016.

[332] S.-H. Teng, Q. Lu, M. Eichstaedt, D. Ford, and T. Lehman. Collaborative Web crawling: Information gathering/processing over Internet. In *Proceedings of the 32nd Annual Hawaii International Conference on System Sciences-Volume 5 - Volume 5*, HICSS '99, page 5044. IEEE Computer Society, 1999.

[333] F. Tohmé and T. Sandholm. Coalition formation processes with belief revision among bounded-rational self-interested agents. *Journal of Logic and Computation*, 9(6):793–815, 1999.

[334] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.

[335] W. Tutte. How to draw a graph. *Proceedings London Mathematical Society*, 52:743–767, 1963.

[336] H. Tverberg. A generalization of Radon's theorem. *Journal London Mathematical Society*, pages 123–128, 1966.

[337] J. C. Urschel, J. Xu, X. Hu, and L. T. Zikatanov. A cascadic multigrid algorithm for computing the Fiedler vector of graph Laplacians. *Journal of Computational Mathematics*, 33(2):209, 2015.

[338] P. M. Vaidya. An optimal algorithm for the all-nearest-neighbors problem. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, FOCS'86, pages 117–122, 1986.

[339] L. Valiant. Universality consideration in VLSI circuits. *IEEE Transactions on Computers*, 30(2):135–140, 1981.

[340] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, Nov. 1984.

[341] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, STOC '81, pages 263–277, 1981.

[342] V. N. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16:264–280, 1971.

[343] S. S. Vempala. *The random projection method*, volume 65 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, Providence, RI, 2004.

[344] N. K. Vishnoi. Lx = b. *Foundations and Trends® in Theoretical Computer Science*, 8(1-2):1 –141, 2013.

[345] K. Voevodski, M.-F. Balcan, H. Röglin, S.-H. Teng, and Y. Xia. Efficient clustering with limited distance information. In *The Conference on Uncertainty in Artificial Intelligence*, UAI'10, pages 632–640, 2010.

[346] K. Voevodski, M.-F. Balcan, H. Röglin, S.-H. Teng, and Y. Xia. Minsum clustering of protein sequences with limited distance information. In *the International Conference on Similarity-based Pattern Recognition*, pages 192–206. Springer-Verlag, 2011.

[347] K. Voevodski, M.-F. Balcan, H. Röglin, S.-H. Teng, and Y. Xia. Active clustering of biological sequences. *Journal of Machine Learning Research*, 13(1):203–225, Jan. 2012.

[348] K. Voevodski, S.-H. Teng, and Y. Xia. Finding local communities in protein networks. *BMC Bioinformatics*, 10:297, 2009.

[349] C. Wang, W. Chen, and Y. Wang. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery*, 25(3):545–576, 2012.

[350] J. H. Wilkinson. *The algebraic eigenvalue problem*. Oxford University Press, Inc., New York, NY, USA, 1988.

[351] V. V. Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, STOC '12, pages 887–898. ACM, 2012.

[352] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li. Fast algorithms for hierarchically semiseparable matrices. *Numerical Linear Algebra with Applications*, 17(6):953–976, 2010.

[353] J. Xu and B. Berger. Fast and accurate algorithms for protein side-chain packing. *Journal of the ACM*, 53(4):533–557, July 2006.

[354] X. Yan, H. Cheng, J. Han, and P. S. Yu. Mining significant graph patterns by leap search. In *the ACM International Conference on Management of Data*, SIGMOD '08, pages 433–444, 2008.

[355] Y. Ye. *Interior Point Algorithms: Theory and Analysis*. John Wiley & Sons, Inc., New York, NY, USA, 1997.

[356] H. P. Young. An axiomatization of Borda's rule. *Journal of Economic Theory*, 9(1):43–52, 1974.

[357] P. S. Yu, J. Han, and C. Faloutsos. *Link Mining: Models, Algorithms, and Applications*. Springer, Incorporated, 1st edition, 2010.

[358] D. Zhou, J. Huang, and B. Schölkopf. Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 1036–1043, 2005.