UNIVERSITY OF CALIFORNIA
SANTA CRUZ

BIG DATA AND SPECTRAL GRAPH THEORY

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

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Andrew Stolman

June 2021

The Dissertation of Andrew Stolman
is approved:

Prof. Seshadhri Comandur, Chair

Prof. Yang Liu

Prof. Daniel Fremont

Quentin Williams
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures iv

List of Tables v

Abstract vi

Dedication vii

Acknowledgments viii

1 Designing algorithms for complex networks 1
- 1.1 Related work: sparse graphs and local computation . . . 2
  - 1.1.1 Cheeger's inequality . . . 3
  - 1.1.2 Local clustering . . . 4
- 1.2 Graph embeddings . . . 7
- 1.3 Contributions . . . 8

2 A near optimal one-sided tester for minor-freeness 9
  - 2.0.1 Related work . . . 11
- 2.1 Main Ideas . . . 12
  - 2.1.1 When do random walks find minors? . . . 13
  - 2.1.2 Returning walks . . . 15
  - 2.1.3 The trapped case: local partitioning to the rescue . . . 16
- 2.2 The algorithm . . . 18
- 2.3 Returning walks and stratification . . . 20
  - 2.3.1 Stratification . . . 21
  - 2.3.2 The correlation lemma . . . 22
- 2.4 Analysis of `FindBiclique` . . . 24
  - 2.4.1 The procedure `FindPath` . . . 24
  - 2.4.2 The procedure `IdealFindBiClique` . . . 26
  - 2.4.3 Criteria for `IdealFindBiClique` to reveal a minor . . . 28
  - 2.4.4 The probabilities of bad events . . . 31
  - 2.4.5 Proof of Theorem 2.4.1 . . . 35
- 2.5 Local partitioning in the trapped case . . . 36
  - 2.5.1 Proof overview . . . 37
  - 2.5.2 Projected Markov chain . . . 38
- 2.6 Wrapping it all up: the proof of the main theorem . . . 44

3 A polynomial-time two sided tester for hyperfinite properties **47**
3.1 The algorithm . . . 47
3.2 Random walks on hyperfinite graphs . . . 48
3.3 The existence of a discoverable decomposition . . . 50
3.4 Proof of main result . . . 51

4 An efficient partition oracle **55**
4.0.1 Consequences . . . 57
4.0.2 Related work . . . 58
4.1 Main Ideas . . . 59
4.1.1 Outline of sections . . . 62
4.2 Global partitioning and its local implementation . . . 62
4.2.1 Truncated diffusion . . . 63
4.2.2 The global partitioning procedure . . . 64
4.2.3 The local implementation . . . 66
4.3 Coordination through the size thresholds: the procedure `findr` . . . 69
4.3.1 The procedure `findr` . . . 71
4.4 Proving the cut bound: the amortization argument . . . 75
4.5 Diffusion Behavior on Minor-Free Families . . . 78
4.6 The proof of Theorem 4.3.10: local partitioning within F . . . 81
4.6.1 The Lovász-Simonovits lemma . . . 82
4.6.2 From leaking timesteps to the dropping of the LS curve . . . 85
4.6.3 Proof of Theorem 4.3.1 . . . 88
4.7 Proofs of applications . . . 90

5 The impossibility of low rank representations for triangle-rich complex networks **92**
5.0.1 Empirical validation . . . 95
5.0.2 Alternate models . . . 95
5.0.3 Broader context . . . 96
5.1 High-level description of the proof . . . 96
5.1.1 Dealing with varying lengths . . . 97
5.2 Proof of Theorem 5.0.2 . . . 98
5.2.1 The basic tools . . . 98
5.2.2 The main argument . . . 100
5.3 Details of empirical results . . . 104
5.3.1 Triangle distributions . . . 106
5.3.2 Alternate graph models . . . 106
5.3.3 Degree distributions . . . 107
5.3.4 Detailed relationship between rank and triangle structure . . . 107

6 On the (in)-effectiveness of matrix factorization-based graph embedding methods for community labeling **109**
6.0.1 Formal description of setting . . . 110
6.0.2 Main results . . . 112
6.0.3 Justification of Setting & Related Work . . . 114
6.1 Mathematical results and interpretation . . . 116
6.1.1 Proof ideas . . . 117
6.2 Empirical verification . . . 118
6.2.1 Community pair prediction . . . 118
6.2.2 Experimental results . . . 119

6.2.3 Experimental setup . . . 119
6.3 Stochastic block models . . . 122
6.3.1 Setup . . . 122
6.3.2 Results . . . 123
6.4 Proof of instability in softmax factorizations . . . 124

7 Evaluating embeddings for link prediction **126**
7.1 The problem with AUC . . . 127
7.2 Reliability curves . . . 128
7.3 Experimental results . . . 128
7.3.1 Experimental setup . . . 129
7.3.2 Datasets . . . 130

Bibliography **132**

List of Figures

1.1 Illustration of Lemma 1.1.5 6

2.1 This figure shows the various subpaths defined in the proof of Claim 2.4.6. The simple path $P'_{a,b}$ (the thick path) from a to b induced by $P_{a,b}$ (in light gray). This path is broken into three contiguous subpaths: the portion "close to" a, the portion close to b, and the remainder. 30

5.1 figure 94

5.2 Table of datasets used 104

5.3 Plots of degree c vs Δ: For each network, we plot c versus the total number of triangles only involving vertices of degree at most c. We divide the latter by the number of vertices, so it corresponds to Δ, as in the main definition. In each subfigure, we plot these both for the original graph, and the maximum Δ in a set of 100 samples from a 100-dimensional embedding. Observe how the embeddings generate graphs with very few triangles among low degree vertices. The gap in Δ for low degree is 2-3 orders of magnitude in all instances. 105

5.4 Plots of degree distributions: For each network, we plot the true degree distribution vs the expected degree distribution of a 100-dimensional embedding. Observe how the embedding does capture the degree distribution quite accurately at all scales. 105

5.5 Plots of degree c vs Δ, for varying rank: For the `Facebook` social network, for varying rank of embedding, we plot c versus the total number of triangles only involving vertices of degree at most c. The embedding is generated by taking the top eigenvectors. Observe how even a rank of 2000 does not suffice to match the true triangle values for low degree. 108

6.1 Each point, (x, y), on the curve represents the approximate fraction of vertices, y, for which the given method produces a precision@10 score of at least x. `LR-Structural` is plotted against the two best performing embedding methods. 1000 vertices are sampled and for each vertex v sampled, the vertices of the graph $u_1, \ldots, u_n$, are ordered by decreasing score assigned by the given classifier. The precision@10 is the fraction of $u_1, \ldots, u_{10}$ which share a community with v. . . 111
6.2 Average precision@10 across 100 samples for all methods across the three datasets. In all cases, `LR-Structural` is the best performing. 120
6.3 Precision@10 reliability curves for sbm datasets. All datasets are stochastically generated datasets with size 10,000 and disjoint communities of size 20. The average degrees are 4, 12 and 20 from left to right. Curves are generated from 100 samples. 123

7.1 ROC-AUC scores of hadamard product embedding models. The scores are calculated on the test set consisting of half true-labeled pairs and half negatively labeled pairs 129
7.2 Reliability curves from 1K vertex samples. For each vertex, v, in the sample, the top d_v neighbors according to the classifier is selected and a precision@d is calculated for each vertex. Each point, (x, y), on the curve represents the approximate fraction of vertices, y, for which the given method produces a precision@d score of at least x. 129

List of Tables

2.1 Stratification notation 20
2.2 Bad Intersections notation 28

6.1 Dataset summary 122

7.1 Summary of datasets 131

Abstract

Big data and spectral graph theory

by

Andrew Stolman

Network data arises naturally in many domains - from protein-protein interaction networks in biology to social networks. Modern storage and collection technology make it possible to collect and analyze networks of unprecedented size. At the same time, the proliferation of machine learning techniques increase the amount of analysis tasks practitioners want to perform. With bigger data and more demands on it, there is a need for faster algorithms tailored to modern datasets.

In this dissertation, we present several new results on algorithms for sparse graphs based on spectral graph theory. From biology to social networks, the networks we encounter in practice are often contain very sparse. Spectral graph theory provides a toolkit which allows us to come up with local procedures for sparse graphs which have desirable global properties, enabling new advances in the field.

In Chapter 2, Chapter 3, and Chapter 4 of this dissertation, new upper bounds in the field of sparse graph property testing are presented. We consider the problem of testing for H-minor-freeness. A near-optimal one-sided tester is presented, as well as a two-sided tester. We also present a polynomial-time algorithm for the related problem of graph partition oracles.

In Chapter 5, Chapter 6, Chapter 7, we examine embeddings of sparse graphs. A popular machine learning tool is to compute geometric embeddings of the vertices of a graph. There has been little principled investigation of the power of these methods. We present several impossibility results which question the efficacy of these embedding methods and follow these up with an empirical investigation.

To my family with inexpressible gratitude.

To my advisor, C. Seshadhri, who introduced me to so many beautiful ideas and people.

To the strangers whose names I have forgotten but whose words still shape my life today. Our lives are chaotic systems and a small kind deed can have incredible consequences.

Acknowledgments

The text of this dissertation includes reprints of the following published material:

- Akash Kumar, C Seshadhri, and Andrew Stolman.

  Random walks and forbidden minors i: An nˆ1/2+o(1)-query one-sided tester for minor closed properties on bounded degree graphs.

  SIAM Journal on Computing, (0):FOCS18–216, 2020

- Akash Kumar, C. Seshadhri, and Andrew Stolman.

  Random walks and forbidden minors II: a poly($d\ \epsilon^{-1}$)-query tester for minor-closed properties of bounded degree graphs.

  In *STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 559–567, 2019

- Akash Kumar, C Seshadhri, and Andrew Stolman.

  Random walks and forbidden minors iii: poly (d/{\epsilon})-time partition oracles for minor-free graph classes.

  arXiv preprint arXiv:2102.00556, 2021

- C Seshadhri, Aneesh Sharma, Andrew Stolman, and Ashish Goel.

  The impossibility of low-rank representations for triangle-rich complex networks.

  Proceedings of the National Academy of Sciences, 117(11):5631–5637, 2020

I would like to thank my co-authors for their contributions which are reprinted here. I am deeply indebted to all of them. Their contributions are enumerated below:

- Chapter 2, Chapter 3, Chapter 4 were co-authored with C. Seshadhri and Akash Kumar. I contributed in writing the manuscript and helping to develop the proofs.

- Chapter 5 was co-authored with C. Seshadhri, Asheesh Goel and Aneesh Sharma. I contributed by helping to develop the proof, write the manuscript and perform the experiments.

- Chapter 6 was co-authored with Caleb Levy, C. Seshadhri and Aneesh Sharma. I contributed parts of the manuscript and performed the experiments.

- Chapter 7 was written in consultation with C. Seshadhri, Aneesh Sharma and Suman Bera. The text and experiments are entirely mine.

Chapter 1

Designing algorithms for complex networks

Understanding the behavior of complex networks is of growing social and economic importance - whether its the rate of diffusion of COVID-19 across the network of human interactions, or the pattern of posts which distinguish a bot account from a genuine one on a social network. Fortunately, the body of work comprising graph algorithms and graph theory provide a formidable toolset. We are (in theory) able to isolate the minimum number of people needed to contain an outbreak or identify spammers with a high degree of accuracy. However, solving these problems efficiently often requires a nuanced understanding of the underlying network data.

Fundamental to dealing with network data is the graph data structure. A graph, $G = (V, E)$, consists of a vertex set, V, and edge set, E, which consists of pairs from V. (We will use the terms vertex and node interchangeably, as well as edge and link.) Graph algorithms have always been a central topic to computer science. Of Karp's 21 NP-complete problems, at least 10 are graph problems. The asymptotic complexity of many natural graph problems are well understood, however the modern rapid increase in size of network datasets and emphasis on fast approximate algorithms for use in machine learning opens new frontiers in the field.

There are essentially two ways to represent a graph: as an adjacency matrix or as an adjacency list. In the adjacency matrix representation, we store an $n \times n$ matrix in memory with an indicator variable in entry (i, j) indicating the presence of edge (v_i, v_j). For the adjacency list model, we store n lists in memory, the ith containing the neighbors of v_i. The memory requirement of the former model is $O(n^2)$ while that of the latter is $O(m)$. Graphs where $m = o(n^2)$ are called *sparse* and adjacency lists are the preferred data structure. Meanwhile graphs which are not sparse are *dense*, and it is usually suggested to use an adjacency matrix

to represent them. While it is possible to convert between the two representations in $O(n^2)$ time, modern datasets are so large that even quadratic time operations can be intolerable. For sublinear algorithms, changing representations is completely out of the question. In this big data world, the two underlying graph representations can suggest totally different algorithmic paradigms.

The speed of edge and neighbor lookups varies across the two graph representations. In an adjacency matrix, answering a query of the form "Is u a neighbor of v?" can be done in constant time - one only need to consult the relevant entry in the array. Whereas in an adjacency list, to answer such a query involves searching for u in v's adjacency list and so has time complexity dependent on d_v. A neighbor query: "what are v's neighbors?" - is $O(d_v)$ in the adjacency list model, but $O(n)$ for adjacency matrices. When there are many edges in the graph and many vertices of degree $\Theta(n)$, the tradeoffs clearly favor the adjacency matrix model since neighbor queries are not much slower than in the adjacency list model and edge queries are much faster. When many vertices have constant degree, the situation is reversed and it makes sense to use the adjacency list model. This tradeoff has deep implications for modern algorithmic design.

The choice of representation goes deeper than polynomial runtime factors when we consider approximation algorithms and notions of distance between graphs. Given two n-vertex graphs, G and H, intuitively, we would like to say that G and H are close when few edge deletion/insertions are required to transform G into H. For G and H with $\Theta(n^2)$ edges, a change of $\Theta(n)$ edges is trivial. If G is any sparse graph, and H is the empty graph, then still only $\Theta(n)$ edge changes are needed to make G isomorphic to H. If we are dealing with mainly sparse graphs, saying that they are all well-approximated by the empty graph is probably not very helpful. Dense and sparse graphs often require completely different frameworks and approaches.

In this thesis, we shall examine several recent results concerning sparse graphs in two very different subfields. In each of these cases, the analogous problem in dense graphs was already solved using well established methods, but sparse graphs present unique challenges requiring new methods. The theme uniting these results is the successful application of spectral graph theory (the name given to the marriage of linear algebra and graph theory). In the first half, we shall see how spectral graph theory can be used to answer problems in property testing, and in the latter half, we apply it in the world of unsupervised machine learning.

1.1 Related work: sparse graphs and local computation

Graph partitioning is an important algorithmic tool Algorithms to provide partitions of graphs with desirable properties can be used as part of a divide-and-conquer style algorithm,

or the partition is useful on its own, such as the partition of a social network into communities, or an electrical network into low capacitance clusters. There are a wide variety of methods and objectives to graph partitioning (see [BMS+16] for a modern survey of uses and techniques). We will focus on a line of graph clustering work which comes from spectral graph theory and see how it unlocks efficient algorithms for sparse networks.

1.1.1 Cheeger's inequality

The following discussion first requires a few definitions. We will always assume that $G = (V, E)$ is a simple undirected graph. For subsets of vertices S, T, $E(S, T)$ denotes the set of edges with endpoint in S and the other in T. For a vertex, v, d_v denotes the degree of v and $\texttt{vol}(S)$ is the sum of degrees of vertices in the set S.

Definition 1.1.1 (conductance)**.** *Define the* conductance *of $S \subseteq V$, $\Phi(S)$ for to be the ratio of edges leaving S over the sums of the degrees of vertices in S, i.e.*

$$\Phi(S) = \frac{|E(S, \bar{S})|}{\min\{\texttt{vol}(S), \texttt{vol}(\bar{S})\}} \tag{1.1}$$

Define the conductance of a graph, $\Phi(G) = \min_{S \subseteq V} \Phi(S)$ to be the minimum conductance of any subset among all subsets of vertices of the graph.

When comparing S and T, if the conductance of S is less than the conductance of T, we say that S is *sparser* than T. **SPARSEST-CUT** is the problem of identifying the minimum conductance cluster in G. It is known to be an NP-hard problem, however Theorem 1.1.3 provides a polynomial time quadratic approximation.

Definition 1.1.2 (sweep set)**.** *Suppose $\hat{v}$ is a real-valued vector with entries indexed by V. Let $u_1, \ldots, u_n$ be elements of V listed in decreasing value assigned by $\hat{v}$. Call $S_t(\hat{v}) = \{u_1, \ldots, u_t\}$ the tth* sweep set *of $\hat{v}$.*

Theorem 1.1.3 (Cheeger's Inequality)**.** *Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of the random walk matrix for a simple undirected graph $G = (V, E)$. Then the following equation holds*

$$1 - \lambda_2 \leq \Phi(G) \leq \sqrt{4(1 - \lambda_2)} \tag{1.2}$$

Moreover, this conductance is achieved by some sweep set of the eigenvector associated with λ_2.

Originating from work on Riemannian geometry, Theorem 1.1.3 is the foundational result in spectral graph theory. It allows us to extract relatively dense clusters from graphs quickly. If a graph, G, has conductance ϕ, Theorem 1.1.3 gives us a polynomial time algorithm to produce a set S such that $\Phi(S) \leq 4\sqrt{\phi}$. One need only compute the eigenvector associated with λ_2, and then output the minimum conductance sweep set among its n sweep sets.

Theorem 1.1.3 also has important consequences for random walks on graphs. Every ergodic Markov chain (such as a lazy random walk on an undirected graph) has a stationary distribution, π. Moreover, π is also an eigenvector associated with $\lambda_1 = 1$, the largest eigenvalue of the transition matrix (or random walk matrix in this case), W. No matter the starting distribution, all walks eventually converge to this distribution. The rate at which this happens is called the *mixing time*. Theorem 1.1.3 allows us to derive a relationship between the conductance of a graph and the mixing time of that graph.

Suppose G is a d-regular graph (i.e. all vertices have degree d), then a well-known fact is that π is uniform over V. Among all probability vectors over V, π has the smallest euclidean 2-norm with $\|\pi\|_2^2 = 1/n$. Given a starting distribution, $\boldsymbol{p}$, since all walks converge to the stationary distribution $\lim t \to +\infty \|W^t \boldsymbol{p}\|_2^2 = 1/n$. Since W is symmetric, it has a set of orthornmal eigenvectors, $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n$ associated with the eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$, and we can write $W^t \boldsymbol{p} = \sum_{i=1}^n (\boldsymbol{p} \cdot \boldsymbol{w}_i) \lambda_i^t \boldsymbol{w}_i$. This makes it easy to track the 2-norm of the t-step probability vector $W^t \boldsymbol{p}$.

$$\begin{aligned}
\left\|W^t \boldsymbol{p}\right\|_2^2 &= \sum_{i=1}^n (\lambda_i^t \boldsymbol{p} \cdot \boldsymbol{w}_i)^2 \\
&\leq \left(\boldsymbol{p} \cdot \frac{\pi}{\|\pi\|_2}\right)^2 + \lambda_2^{2t} \sum_{i=2}^n (\boldsymbol{p} \cdot \boldsymbol{w}_i)^2 \\
&\leq \frac{1}{n} + \lambda_2^{2t}
\end{aligned}$$

Thus in $t = O\left(\frac{\ln(1/\lambda_2)}{\ln(1/\varepsilon)}\right)$ steps, the 2-norm from any start distribution is within ε additive error of stationary. By rearranging Theorem 1.1.3, we can directly relate the conductance of G with its mixing time by replacing λ_2 with the upper bound $1 - \frac{\phi^2}{4}$.

The power of Theorem 1.1.3 is it relates three seemingly far-flung concepts: a combinatorial property of graphs (conductance), a spectral property of matrices and the evolution of Markov processes. The ability to quickly switch between these frames of references has profound algorithmic consequences.

1.1.2 Local clustering

Cheeger's inequality implies an algorithm for partitioning graphs into low conductance clusters: iteratively compute the second eigenvector and remove the minimum conductance cluster. The time complexity of this algorithm is polynomial in n. A natural question to ask is can this bound be improved to linear in n?

It turns out the answer is affirmative for sparse graphs. Local clustering algorithms produce clusters in time proportional to the number of edges touched by the cluster. In sparse

graphs, with $O(n)$ edges, this leads to linear algorithms for clustering the whole graph. For a reference see Chapter 22 of [Spib].

The ideas of local clustering heavily influence the algorithms of Chapter 2, Chapter 3, and Chapter 4 of this thesis, and are later used to provide competitive ML models in the remaining chapters. In the remainder of this section, we give an overview of the state-of-the-art in local clustering algorithms.

Local clustering was first introduced in [ST04, ST13] as a part of near linear time approximation algorithm for solving certain classes of linear equations. Inspired by the work in [LS90a, LS93] on analyzing the mixing rate of random walks, [ST04] gives an algorithm for extracting low conductance cuts by using random walks in time that depends polynomially on the cut size, and only logarthmically in the size of the graph. Since then, there have been various improvements by refining certain methods used [ACL06, AGPT16].

The local partitioning results are obtained by a familiar pattern we call the **Lovász-Simonovits curve technique**. This technique involves studying the value of a certain potential function of random walk vectors as the random walk vector evolves over time.

Definition 1.1.4. *For a probability vector over V, $\boldsymbol{p}$, an integer $k \in [0, n]$, and positive integer t, let S_k^t denote the k^{th} sweep set of $W^t\boldsymbol{p}$. The Lovász-Simonovits potential function is defined for integer values of k as follows:*

$$h_t(x, \boldsymbol{p}) = \sum_{v \in S_x} \boldsymbol{p}(v)$$

When $\boldsymbol{p}$ is clear from context (it is often just an indicator vector for the random walk start point), we will omit the second argument. For non-integral x, h_t is linearly interpolated from $h_t(\lfloor x \rfloor, \lceil x \rceil)$.

This potential function has several important properties. It is a monotone increasing curve such that $h_t(0) = 0$ and $h_t(n) = 1$. Moreover, h_t must be concave in the first argument since by definition $h_t(x) - h_t(x-1) \leq h_t(x-1) - h_t(x-2)$ for all x. When $\boldsymbol{p} = 1_v$, $h_t(x, \boldsymbol{p})$ sharply rises from $(0, 0)$ to $(1, 1)$, and then is flat until $(n, 1)$. As t increases, and the walk approaches the stationary distribution, $h_t(x)$ converges to the straight line connecting $(0, 0)$ to $(n, 1)$. Lemma 2.5.6 states the rate at which this convergence happens in terms of the conductance of the sweep sets of the random walk vector.

Lemma 1.1.5. *For some start distributionm, $\boldsymbol{p}$, let $L_{k,t}$ denote the kth sweep set of the vector $W^t\boldsymbol{p}$, and let $k^* = \min\{k, n-k\}$. The following holds for all k and all t.*

$$h_t(k) \leq \frac{1}{2}\left[h_{t-1}(k - 2k^*\Phi(L_{k,t})) + h_{t-1}(k + 2k^*\Phi(L_{k,t}))\right]$$

Fig. 1.1 illustrates Lemma 1.1.5. It says that every point on the curve $h_t(x)$ lies below some chord drawn on the curve h_{t-1}. The length of that chord in the horizontal dimension is

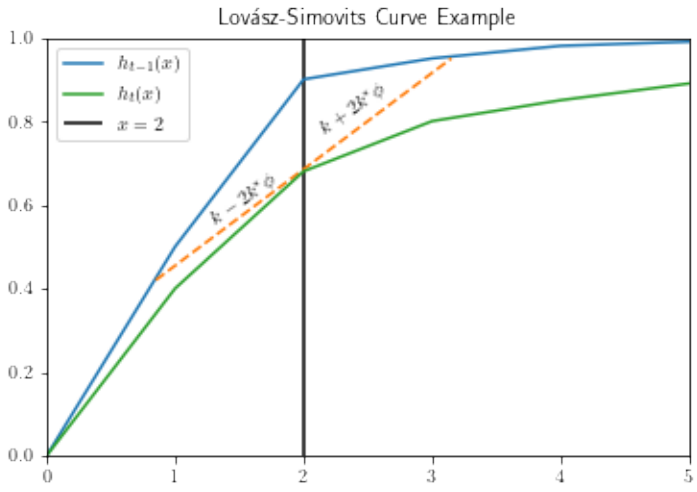

Figure 1.1: Illustration of Lemma 1.1.5

proportional to the conductance of the level sets of the random walk vectors. Therefore, if the conductances of all level sets are large, the chords are wide, and because of convexity, $h_{t+1}(x)$ approaches stationarity relatively quickly. Thus Lemma 1.1.5 was used originally to bound the mixing time of graphs which do not have any low conductance cuts.

Local partitioning, as pioneered in [ST04], draws on the contra-positive of Lemma 1.1.5: if the curves $h_1, h_2, \ldots, h_t$ do not approach the stationary distribution quickly, then there must exist some $t' \leq t$ for which $W^t \boldsymbol{p}$ has a sparse sweep set.

Lemma 1.1.6. *If $h_t(k) > \sqrt{k}\left(1 - \phi^2/2\right)^t + pk$ for some $\phi, p \in [0, 1]$, then there must exist some $t' \leq t$ such that some sweep cut of $W^{t'}\boldsymbol{p}$ has conductance at most ϕ and minimum probability at least p.*

So long as we can quickly simulate random walks, Lemma 1.1.6 implies a local partitioning algorithm. If $h_t(k)$ is sufficiently large for some k, we need only consider the $\frac{t}{p}$ eligible sweep sets of $W\boldsymbol{p}, W^2\boldsymbol{p}, \ldots W^t\boldsymbol{p}$, and we are guaranteed to find a low conductance cut. Assuming we have oracle access to $W\boldsymbol{p}, W^2\boldsymbol{p}, \ldots W^t\boldsymbol{p}$, this takes time $O(\frac{t}{p}\frac{d}{p})$ in a d-bounded degree graph to consider all $\frac{t}{p}$ subsets of volume at most $\frac{d}{p}$.

In order to actually produce a low conductance cut, a local partitioning algorithm needs a good *seed*, i.e. a vertex such that $h_t(k, 1_v) > \sqrt{k}\left(1 - \phi^2/2\right)^t + pk$ for some p and (preferably small) t. For a vertex, v, let $W_t v$ be the t-step random walk vector for walks starting from v. Then $h_t(1) = \|W_t v\|_\infty$, and we can see the tight relationship between the spectral properties of W as a linear operator and the presence of small, easily detectable sparse cuts.

Besides a good seed from which to start, a local partitioning algorithm also needs an

efficient way to simulate random walks. Since computing the matrix vector product $W^t \boldsymbol{p}$ takes time which depends on n, doing so would make the algorithm no longer "local." Much of the work in local partitioning comes down to constructing new approximation methods and showing that the resulting vectors still follow a Lovász-Simonovits-style property. [ST04] use random walks where small entries in the vector are ignored, while [ACL06] use approximate personal pagerank vectors, and [AGPT16] use a more exotic construction.

1.2 Graph embeddings

Networks pose a challenge to many machine learning methods. The "classical" machine learning setting assumes a fixed number of features per observation, and the goal is to learn some function with domain in this feature space. Network data, however, usually does not have this structure. Instead of taking a fixed number of features per observation, in network data, we observe interactions between objects. For a sample of n nodes, there are n possible interactions for each node. With the number of features growing linearly with n, the well known "curse of dimensionality" rules out many off-the-shelf machine learning methods.

One of the most popular attempts to circumnavigate the curse of dimensionality has been unsupervised feature learning. This approach, also called feature extraction or graph embedding, seeks to learn a function, $f : V \to \mathbb{R}^d$, such that distances in $\mathbb{R}^d$ somehow reflect distances in the graph and $d << n$. (See [HYL18, CAEHP+20] for surveys). There are myriad methods employed toward low-dimensional embeddings, however there is limited principled understanding of the power of low-dimensional embeddings in general.

We commence a study of graph embeddings. In order to carry out a principled analysis, we focus on one popular class of embeddings based on matrix factorization. This class of embeddings is characterized by trying to learn some $n \times d$ embedding matrix, E, such that $L(M, EE^T)$ is minimized for some loss function, L and $n \times n$ matrix, M, derived from the graph structure. Prominent examples of such methods are the various SVD-based methods, DeepWalk [PARS14], Node2Vec [GL16a] and NetMF [QDM+18].

The idea behind matrix factorization methods (MFM) is to choose some matrix, M which is meant to capture node similarity, i.e. $M_{u,v}$ is some similarity measure of the vertex pair (u, v). Common choices of M include the adjacency matrix, matrices derived from the graph laplacian and low powers of the random walk matrix. The hope is that these matrices accurately represent the adjacency information and/or community structure of the graph.

We can divide the class of matrix factorization embedding methods based on the loss function, L. When $L(M, EE^T) = \|M - EE^T\|_2$, the MFM is called a *direct factorization method*. Analyzing such methods is relatively straightforward since a singular value decomposition of M is known to be an optimal solution. In Chapter 5, we explore the limitations of this class. The

other loss function which appears in the literature is based on cross entropy. In cross entropy embeddings, M is a stochastic matrix and the goal is to minimize $H(M, f(EE^T))$ where H is the average cross entropy between rows and f is some function that maps the rows of EE^T to probability vectors. Chapter 6 explores some limitations of this class.

1.3 Contributions

Here we will briefly summarize the contributions and outline the structure of this thesis.

In Chapter 2, Chapter 3 and Chapter 4, we present a series of three papers that answer a series of related open problems in bounded-degree graph property testing [KSS18, KSS19b, KSS21]. This work applies local clustering to algorithms for deciding graph planarity. Planarity is the property of being "planar" and is characterized by graphs which can be drawn on the plane without crossing edges. It is a known fact that all dense graphs are non-planar, however there are also many sparse graphs which are not. In Chapter 2, we present the first sublinear algorithm for distinguishing non-planar sparse graphs from planar ones and also provide a certificate of non-planarity. This algorithm operates in time $O(\sqrt{n})$. In Chapter 3, we show that this bound can be improved to lose the dependence on n if we foregoe the requirement to produce a certificate of non-planarity and allow two-sided error. We are even able to extend the methods slightly beyond planarity to cover the property of *hyperfinite* graphs. In Chapter 4, we show how to further generalize the method and provide the first ever partition oracle with runtime polynomial in the distance parameter.

In Chapter 5, Chapter 6 and Chapter 7, we turn our attention to other frontiers in sparse graph algorithms. In particular, we examine the effectiveness of geometric graph embeddings [SSSG20a]. These algorithms, often framed as unsupervised learning or feature extraction algorithms, seek to represent the graph as a point cloud in some low dimensional space with the distance between two vertices correlating with their distance in the graph. Here we find a rich field to apply linear algebra in novel ways. In Chapter 5, we show that a popular class of embedding methods is incapable of representing the sparse, yet triangle-rich structure that we observe in many real world complex networks. In Chapter 6, we show that a popular extension of these embedding methods still is unlikely to succeed to capture these networks. In both chapters, the focus is as much on the real world empirical evidence of these limitations as it is on the theoretical statements.

Chapter 2

A near optimal one-sided tester for minor-freeness

Deciding if an n-vertex graph G is planar is a classic algorithmic problem solvable in linear time [HT74]. The Kuratowski-Wagner theorem asserts that any non-planar graph must contain a K_5 or $K_{3,3}$-minor [Kur30, Wag37]. Thus, certifying non-planarity is equivalent to producing such a minor, which can be done in linear time. Can we beat the linear time bound if we knew that G was "sufficiently" non-planar?

Assume random access to an adjacency list representation of a bounded degree graph, G. Suppose, for some constant $\varepsilon > 0$, one had to remove εn edges from G to make it planar. Can one find a forbidden (K_5 or $K_{3,3}$) minor in $o(n)$ time? It is natural to ask this question for any property expressible through forbidden minors. By the famous Robertson-Seymour graph minor theorem [RS04], any graph property $\mathcal{P}$ that is closed under taking minors can be expressed by a finite list of forbidden minors. We desire sublinear time algorithms to find a forbidden minor in any G that requires εn edge deletions to make it have $\mathcal{P}$.

This problem was first posed by Benjamini-Schramm-Shapira [BSS10] in the context of property testing on bounded degree graphs. We follow the model of property testing on bounded degree graphs as defined by Goldreich-Ron [GR02]. Fix a degree bound d. Consider $G = (V, E)$, where $V = [n]$, and G is represented by an adjacency list. We have random access to the list through *neighbor queries*. There is an oracle that, given $v \in V$ and $i \in [d]$, returns the ith neighbor of v (if no neighbor exists, it returns $\perp$).

Given any property $\mathcal{P}$ of graphs with degree bound d, the distance of G to $\mathcal{P}$ is defined as the minimum number of edge additions/removals required to make G have $\mathcal{P}$ divided by dn. This ensures that the distance is in $[0, 1]$. We say that G is ε-far from $\mathcal{P}$ if the distance to $\mathcal{P}$ is more than ε.

A property tester for $\mathcal{P}$ is a randomized procedure that takes as input (query access to) G and a proximity parameter $\varepsilon > 0$. If $G \in \mathcal{P}$, the tester must accept with probability at least $2/3$. If G is ε-far from $\mathcal{P}$, the tester must reject with probability at least $2/3$. A one-sided tester must accept $G \in \mathcal{P}$ with probability 1, and thus must provide a certificate of rejection.

We are interested in properties expressible through *forbidden minors*. Fix a finite graph H. The property $\mathcal{P}_H$ of *H-minor-freeness* is the set of graphs that do not contain H as a minor. Observe that one-sided testers for $\mathcal{P}_H$ have a special significance since they must produce an H-minor whenever they reject. One can cast one-sided property testers for $\mathcal{P}_H$ as sublinear time procedures that find forbidden minors. Our main theorem follows.

Theorem 2.0.1. *Fix a finite graph H with $|V(H)| = r$ and arbitrarily small $\delta > 0$. Let $\mathcal{P}_H$ be the property of H-minor-freeness. There is a randomized algorithm that takes as input (oracle access to) a graph G with maximum degree d, and a parameter $\varepsilon > 0$. Its running time is $dn^{1/2+O(\delta r^2)} + d\varepsilon^{-2\exp(2/\delta)/\delta}$. If G is ε-far from $\mathcal{P}_H$, then, with probability $> 2/3$, the algorithm outputs an H-minor in G.*

Equivalently, there exists a one-sided property tester for $\mathcal{P}_H$ with the above running time.

The graph minor theorem of Robertson and Seymour [RS04] asserts the following. Consider any property $\mathcal{Q}$ that is closed under taking minors. There is a finite list $\boldsymbol{H}$ of graphs such that $G \in \mathcal{Q}$ iff G is H-minor-free for all $H \in \boldsymbol{H}$. If G is ε-far from $\mathcal{Q}$, then G is $\Omega(\varepsilon)$-far from $\mathcal{P}_H$ for some $H \in \boldsymbol{H}$. Thus, a direct corollary of Theorem 2.0.1 is the following.

Corollary 2.0.2. *Let $\mathcal{Q}$ be any minor-closed property of graphs with degree bound d. For any $\delta > 0$, there is a one-sided property tester for $\mathcal{Q}$ with running time $O(dn^{1/2+\delta} + d\varepsilon^{-2\exp(2/\delta)/\delta})$.*

In the following discussion, we suppress dependences on ε and n^δ by $O^*(\cdot)$. Previously, the only graphs H for which an analogue of Theorem 2.0.1 was known are the following: $O^*(1)$ time for H being a forest, $O^*(\sqrt{n})$ for H being a cycle [CGR+14], and $O^*(n^{2/3})$ for H being $K_{2,k}$, the $(k \times 2)$-grid, and the k-circus [FLVW17, FLVW18]. No sublinear time bound was known for planarity.

Corollary 2.0.2 implies that properties such as planarity, series-parallel graphs, embeddability in bounded genus surfaces, and bounded treewidth are all one-sided testable in $O^*(\sqrt{n})$ time.

We note a particularly pleasing application of Theorem 2.0.1. Suppose a bounded degree graph, G, has more than $(3+\varepsilon)n$ edges. Then it is guaranteed to be ε-far from being planar, and thus, there is an algorithm to find a forbidden minor in G in $O^*(\sqrt{n})$ time. Since all minor-closed properties have constant average degree bounds, analogous statements can be made for all such properties.

2.0.1 Related work

Graph minor theory is a deep topic, and we refer the reader to Chapter 12 of Diestel's book [Die10] and Lovász's survey [Lov06]. For our purposes, we use as a black-box a polynomial time algorithm that finds fixed minors in a graph. A result of Kawarabayashi-Kobayashi-Reed provides an $O(n^2)$ time algorithm [KKR12].

Property testing on graphs is an immensely rich area of study, and we refer the reader to Goldreich's recent textbook for more details [Gol17]. There is a significant difference between the theory of property testing for dense graphs and that of bounded degree graphs. For the former, there is a complete characterization of properties (one-sided, non-adaptive) testable in query complexity independent of graph size. There is a deep connection between property testing and the Szemeredi regularity lemma [AFNS06]. Property testing for bounded degree graphs is much less understood. This study was initiated by Goldreich-Ron, and the first results focused on connectivity properties [GR02]. Czumaj-Sohler-Shapira proved that hereditary properties of non-expanding graphs are testable [CSS09]. A breakthrough result of Benjamini-Schramm-Shapira (henceforth BSS) proved that all minor-closed (more generally, hyperfinite) properties are two-sided testable in constant time. The dependence on ε was subsequently improved by Hassidim et al., using the concept of local partitioning oracles [HKNO09]. A result of Levi-Ron [LR15] significantly simplified and improved this analysis, to get a final query complexity quasi-polynomial in $1/\varepsilon$. Indeed, it is a major open question to get polynomial dependence on $1/\varepsilon$ for two-sided planarity testers. Towards this goal, Ito and Yoshida give such a bound for testing outerplanarity [YI15], or Edelman et al. generalize for bounded treewidth graphs [EHNO11].

In contrast to dense graph testing, there is a significant jump in complexity for one-sided testers. BSS first raised the question of one-sided testers for minor-closed properties (especially planarity) and conjectured that the bound is $O(\sqrt{n})$. Czumaj et al. [CGR+14] made the first step by giving an $\widetilde{O}(\sqrt{n})$ one-sided tester for the property of being C_k-minor-free [CGR+14]. For $k = 3$, this is precisely the class of forests. This tester is obtained by a reduction to a much older result of Goldreich-Ron for one-sided bipartiteness testing for bounded degree graphs [GR99] (the results in Czumaj et al. are obtained by black-box applications of this result). Czumaj et al. adapt the one-sided $\Omega(\sqrt{n})$ lower bound for bipartiteness and show an $\Omega(\sqrt{n})$ lower bound for one-sided testers for H-minor-freeness when H has a cycle [CGR+14]. This is complemented with a constant time tester for H-minor-freeness when H is a forest.

Recently, Fichtenberger-Levi-Vasudev-Wötzel give an $\widetilde{O}(n^{2/3})$ tester for H-minor-freeness when H is one of the following graphs: $K_{2,k}$, the $(k \times 2)$-grid or the k-circus graph (a wheel where spokes have two edges) [FLVW17, FLVW18]. This subsumes the properties of outerplanarity and cactus graphs. This result uses a different, more combinatorial (as opposed to random walk based) approach than Czumaj et al. A one step random walk in a graph G that begins at a vertex v consists of picking a neighbor $u \sim N(v)$ u.a.r and going to that vertex.

The use of random walks in property testing was pioneered by Goldreich-Ron [GR99] and was then (naturally) used in testing expansion properties and clustering structure [GR11, CS10, KS08, NS10, KPS13, CPS15]. Our approach is inspired by the Goldreich-Ron analysis, and we discuss more in the next section. A number of previous results have used random walks for routing in expanders [BFU99, KR96]. We use techniques from Kale-Seshadhri-Peres to analyze random walks on projected Markov Chains [KPS13]. We also employ the local partitioning methods of Spielman-Teng [ST12], which is in turn derived from the Lovász-Simonovits analysis technique [LS90a].

2.1 Main Ideas

We give an overview of the proof strategy and discuss the various moving parts of the proof. Assume that G is a d-regular graph. It is instructive to understand the method of Goldreich-Ron (henceforth GR) for one-side bipartiteness testing [GR99]. The basic idea to perform $O(\sqrt{n})$ *lazy random walks* of $\text{poly}(\log n)$ length from a u.a.r vertex s. Recall that a lazy random walk stays at the current vertex with probability $1/2$, and moves to a uniform random neighbor (since we assume d-regularity) with probability $1/2$. An odd cycle is discovered when two walks end at the same vertex v, through path of differing parity (of length).

The GR analysis first considers the case when G is an expander (and ε-far from bipartite). In this case, the walks from s reach the stationary distribution. One can use a standard collision argument to show that $O(\sqrt{n})$ suffice to hit the same vertex v twice, with different parity paths. The deep insight is that any graph G can be decomposed into pieces where the algorithm works, and each piece P has a small cut to $\overline{P}$. This has connections with decomposing a graph into expander-like pieces [Tre05, AGPT16]. Famously, the Arora-Barak-Steurer algorithm [ABS15] for unique games basically proves such a statement. We note that GR does *not* decompose into expanders, but rather into pieces where the expander analysis goes through. So, one might hope to analyze the algorithm by its behavior on each component. Unfortunately, the algorithm cannot produce the decomposition; it can only walk in G and hope that performing random walks in G suffice to simulate the procedure within P. This is extremely challenging, and is precisely what GR achieve (this is the bulk of the analysis). The main lemma produces a decomposition into such pieces, such that for each piece P, there exists $s \in P$ wherein short random walks (in G) from s reach all vertices in P with sufficient probability. One can think of this as a simulation argument: we would like to simulate the random walk algorithm running only on P, through random walks in G.

The challenge of general minors: With planarity in mind, let us focus on finding K_5 minors. It is highly unlikely that random walks from a single vertex will find a such a minor. Intuitively, we would need to find 5 different vertices, launch random walks from all of them

and hope these walks will produce a minor. Thus, we would need to simulate a much more complex procedure than the (odd) cycle finder of GR. Most significantly, we need to understand the random walks behavior from multiple sources within P simultaneously. The GR analysis actually constructs the pieces P by a local partitioning looking at the random walk distribution from a single vertex. There is no guarantee on random walk behavior from other vertices in P.

There is a more significant challenge from arbitrary minors. The simulation does not say anything about the specific structure of the paths generated. It only deals with the probability of reaching v from s by a random walk in G when v and s are in the same piece. For bipartiteness, as long as we find two paths of differing parity, we are done. They may intersect each other arbitrarily. For finding a K_5 minor, the actual intersection matter. We would need paths between all pairs of 5 seed vertices to be "disjoint enough" to give a K_5 minor. This appears extremely difficult using the GR analysis. Even if we did understand the random walk behavior (in G) from all vertices in P, we have little control over their behavior when they leave P. (Based on the parameters, the walks leave P with high probability.) They may intersect arbitrarily, and thus destroy any minor structure.

2.1.1 When do random walks find minors?

Inspired by GR, let us start with an algorithm to find a K_5 minor in an expander G (variants of these ideas were present in a result of Kleinberg-Rubinfeld that expanders contain an H-minor for any H with $n/\text{poly}(\log n)$ edges [KR96]). Let ℓ denote the mixing time. Pick u.a.r. a vertex, s, and launch 5 random walks each of length ℓ to reach $v_1, v_2, \ldots, v_5$. From each v_i, launch $\sqrt{n}$ random walks each of length ℓ. With high probability, a walk from v_i and a walk from v_j will "collide" (end at the same vertex). We can collect these collisions to get paths between all v_i, v_j, and one can, with some effort, show that these form a K_5-minor.

Our main insight is to show that this algorithm, with minor modifications, works even when random walks have extremely slow mixing properties. When the random walks mix even more slowly than the requisite bound, we can essentially perform local partitioning to pull out very small (n^δ for arbitrarily small $\delta > 0$) pieces that have low conductance cuts. We can simply query all edges in this piece and run a planarity test.

There is a parameter $\delta > 0$ that can be set to an arbitrarily small constant. Let us set the random walk length ℓ to n^δ, and let $\mathbf{p}_{s,\ell}$ be the random walk distribution after ℓ steps from s. Our proof splits into two cases, where $\alpha = c\delta$ for explicit constant $c > 1$:

- Case 1 (the leaky case): For at least εn vertices s, $\|\mathbf{p}_{s,\ell}\|_2^2 \leq 1/n^\alpha$.
- Case 2 (the trapped case): For at least $(1-\varepsilon)n$ vertices s,$\|\mathbf{p}_{s,\ell}\|_2^2 > 1/n^\alpha$.

In the leaky case, the upper bounds is too weak to argue about convergence of the random walk. We merely have the property that a random walk of length n^δ (roughly speaking) spreads to a

set of size $n^{c\delta}$.

We prove that, in the leaky case, the procedure described in the first paragraph succeeds in finding a K_5 with high probability. We give an outline of this proof strategy.

Let us assume that $\mathbf{p}_{v,\ell/2} = \mathbf{p}_{v,\ell}$ (so $\ell/2$-length walks have "stabilized"). Let us make a slight modification to the algorithm. We pick $v_1, \ldots, v_5$ as before, with ℓ-length random walks from s. We will perform $O(\sqrt{n})$ $\ell/2$ length random walks from each v_i to produce the K_5 minor. By symmetry of the random walks, the probability that a single walk from v_i and one from v_j collide (to produce a path) is exactly $\mathbf{p}_{v_i,\ell/2} \cdot \mathbf{p}_{v_j,\ell/2}$. Thus, we would like these dot products to be large. By the symmetry of the random walk, the probability of an ℓ-length random walk starting from s and ending at v is $\mathbf{p}_{s,\ell/2} \cdot \mathbf{p}_{v,\ell/2}$. In other words, the entries of $\mathbf{p}_{s,\ell}$ are precisely these dot products, and $\|\mathbf{p}_{s,\ell}\|_2^2 = \sum_{v \in V} (\mathbf{p}_{s,\ell/2} \cdot \mathbf{p}_{v,\ell/2})^2 = \mathbf{E}_{v \sim \mathbf{p}_{s,\ell/2}}[\mathbf{p}_{s,\ell/2} \cdot \mathbf{p}_{v,\ell/2}]$. Since $\mathbf{p}_{s,\ell/2} = \mathbf{p}_{s,\ell}$, we rewrite to get $\mathbf{p}_{s,\ell/2} \cdot \mathbf{p}_{s,\ell/2} = \mathbf{E}_{v \sim \mathbf{p}_{s,\ell/2}}[\mathbf{p}_{s,\ell/2} \cdot \mathbf{p}_{v,\ell/2}]$.

Think of the dot products as correlations between distributions. We are saying that the average correlation (over some distribution on vertices) of $\mathbf{p}_{v,\ell/2}$ with $\mathbf{p}_{s,\ell/2}$ is exactly the self-correlation of $\mathbf{p}_{s,\ell/2}$. If the distributions mostly had low ℓ_2-norm (as in the leaky case), we might hope that these distributions are reasonably correlated with each other. Indeed, this is what we prove in Lemma 2.3.11. Under some conditions, we show that $\mathbf{E}_{v_i,v_j \sim \mathbf{p}_{s,\ell/2}}[\mathbf{p}_{v_i,\ell/2} \cdot \mathbf{p}_{v_j,\ell/2}]$ can be lower bounded, where $\mathbf{p}_{s,\ell/2}$ is exactly the distribution the algorithm picks the v_i and v_j from. This is evidence that $\ell/2$-length random walks will connect the v_i's through collisions.

There are four difficulties in increasing order of worry.

1. We only have a lower bound of the average $\mathbf{p}_{v_i,\ell/2} \cdot \mathbf{p}_{v_j,\ell/2}$. We would need bounds for all (or most) pairs to produce a minor.

2. $\mathbf{p}_{v,\ell}$ might be very different from $\mathbf{p}_{v,\ell/2}$.

3. The expected number of collisions between walks from v_i and v_j is controlled by the dot product above, but the variance (which really controls the probability of getting a collision) can be large. There are instances where the dot product is high, but the collision probability is extremely low.

4. There is no guarantee that these paths will produce a minor since we do not have any obvious constraints on the intermediate vertices in the path.

The first problem is surmounted by a technical trick. It turns out to be cleaner to analyze the probability of getting a biclique minor. Observe that $K_{r,r}$ has K_r as a minor. (Contract all the edges in any perfect matching of $K_{r,r}$ to obtain K_r.) So, we perform 10 random walks from s to get sets $A = \{a_1, a_2, \ldots, a_5\}$ and an analogous B. We launch $\ell/2$-length

random walks from each vertex in $A \cup B$. The average lower bound on the dot product suffices to get a lower bound on the probability of getting a $K_{5,5}$-minor, which contains a K_5-minor.

For the second problem, it turns out that the weaker bound of $\|\mathbf{p}_{v,\ell}\|_2 = \Omega(n^{-\delta}\|\mathbf{p}_{v,\ell/2}\|_2)$ suffices. We could try to search for some value of ℓ where this happens. If there was no (small) value of ℓ where this bound held, then it suggest that $\|\mathbf{p}_{v,n^\delta}\|_2$ is extremely small (say $\Theta(1/n)$). This kind of reasoning is detailed more in the next subsection.

The third problem requires bounds on the variance, or higher norms, of $\mathbf{p}_{v,\ell/2}$. Unfortunately, there appears be no handle on these. At a high level, our idea is to truncate $\mathbf{p}_{v,\ell/2}$ by ignoring large entries. This truncated vector is not a probability vector any more, but we can hope to redo the analysis for such vectors.

Now for the fourth problem. Naturally, if the vertices $v_1, \ldots, v_5$ are close to each other, we do not expect to get a minor by connecting them. Suppose they were sufficiently "spread out", One could hope that the paths connecting the v_i, v_j pairs would only intersect "near" the v_i. The portion of the paths nears the v_i's could be contracted to get a K_5-minor. We can roughly quantify how far the v_i's will be by the variance of $\mathbf{p}_{v,\ell/2}$. Thus, the third and fourth problem are coupled.

2.1.2 Returning walks

The main technical contribution of our work is in defining R-returning walks. These are walks that periodically return to a given set of vertices, R. A careful analysis of these walks provides the tools to handle the various problems discussed above.

Fix ℓ as before. Formally, an R-returning walk of length $j\ell$ (for $j \in \mathbb{N}$) is a walk that encounters R at every $i\ell$ step $\forall i \in [j]$. While random walk distributions can have poor variance, we can carefully choose R to ensure that the distribution of R-returning walks is well-behaved. We will quantify this as approximate "support uniformity" (being approximatedly uniform on the support).

In the leaky case, there is some (large) set, R, such that $\forall s \in R$, $\|\mathbf{p}_{s,\ell/2}\|_2^2 \leq 1/n^\alpha$. Let $\mathbf{p}_{[R],s,\ell}$ be the random walk distribution restricted to R. Suppose for some $s \in R$, $\|\mathbf{p}_{[R],s,\ell}\|_2^2 \geq 1/n^{\alpha+\delta}$. Observe that each entry in $\mathbf{p}_{[R],s,\ell}$ is $\mathbf{p}_{s,\ell/2} \cdot \mathbf{p}_{v,\ell/2}$, for $s, v \in R$. By Cauchy-Schwartz, this is at most $1/n^\alpha$. For any distribution $\mathbf{v}$, the condition $\|\mathbf{v}\|_2^2 = \|\mathbf{v}\|_\infty$ is equivalent to support uniformity. While $\mathbf{p}_{[R],s,\ell}$ is not a distribution, if one assumes that $\|\mathbf{p}_{[R],s,\ell}\|_1$ is sufficiently large, we deduce that $\mathbf{p}_{[R],s,\ell}$ is approximately support uniform. (When R is sufficiently large, one can prove that $\|\mathbf{p}_{[R],s,\ell}\|_1$ is large.) The math discussed in the previous section goes through for any such $s \in R$. In other words, if the random walk algorithm started from s, it succeeds in finding a K_5 minor.

Suppose only a negligible fraction of vertices satisfied this condition, and so our algorithm would not actually find such a vertex. Let us remove all these vertices from R

(abusing notation, let R be the resulting set). Now, $\forall s \in R$, $\|\mathbf{p}_{[R],s,\ell}\|_2^2 \leq 1/n^{\alpha+\delta}$. So, the bound on the l_2-norm has fallen by an n^δ factor. What does $\mathbf{p}_{[R],s,\ell} \cdot \mathbf{p}_{[R],v,\ell}$ signify? This is the probability of a 2ℓ-length random walk starting from s, ending at v, and encountering R at the ℓ-th step. This is an R-returning walk of length 2ℓ. Let $\mathbf{q}_{[R],s,2\ell}$ denote the vector of R-returning walk probabilities. Suppose for some s, $\|\mathbf{q}_{[R],s,2\ell}\|_2^2 \geq 1/n^{\alpha+2\delta}$. By Cauchy-Schwartz, $\|\mathbf{q}_{[R],s,2\ell}\|_\infty \leq 1/n^{\alpha+\delta}$, implying that $\mathbf{q}_{[R],s,2\ell}$ is approximately support uniform. Again, the math of the previous section goes through for such an s.

We remove all vertices that have this property, and end up with R such that $\forall s \in R$, $\|\mathbf{q}_{[R],s,2\ell}\|_2^2 \leq 1/n^{\alpha+2\delta}$. Observe that $\mathbf{q}_{[R],s,2\ell} \cdot \mathbf{q}_{[R],v,2\ell}$ is a probability of a 4ℓ R-returning walk. We then iterate this argument.

In general, this argument goes through phases. In the ith phase, we find all $s \in R$ that satisfy $\|\mathbf{q}_{[R],s,2^i\ell}\|_2^2 \geq 1/n^{\alpha+i\delta}$. We show that the random walk procedure of the previous section (with some modifications) finds a K_5-minor starting from such vertices. We remove all such vertices from R, increment i and continue the argument. The vertices removed at the ith phase are called the *ith stratum*, and we refer to this entire process as stratification. Intuitively, for vertices in the ith stratum, the R-returning (for the setting of R at that phase) walk probabilities roughly form a uniform distribution of support $n^{\alpha+i\delta}$. Thus, for vertices in higher strata, the random walks are spreading to larger sets.

There is a major problem. The $\mathbf{q}$ vectors are *not* distributions, and the vast majority of walks are not R-returning. Indeed, the reduction in norm as we increase strata might simply be an artifact of the lower probability of a longer R-returning walk (note that the walks lengths are increasing exponentially in the phase number). We prove a spectral lemma asserting that this is not the case. As long as R is sufficiently large, the probabilities of R-returning walks are sufficiently high. Unfortunately, these probabilities (must) decrease exponentially in the number of returns. In the ith phase, the walk length is $2^i\ell$ and it must return to R 2^i times. Here is where the n^δ decay in l_2-norm condition saves us. After $1/\delta$ phases, the $\|\mathbf{q}_{[R],s,2^i\ell}\|_2^2$ is basically $1/n$. The spectral lemma tells us that if R is still large, the probability that a $2^{1/\delta}\ell$ length walk is R-returning is sufficiently large. Thus, the norm cannot decrease, and almost all vertices end up in the very next stratum. If R was small, then there is an earlier stratum containing $\Omega(\delta\varepsilon n)$ vertices. Regardless of the case, there exists a $i \leq 1/\delta + O(1)$ such that the ith stratum contains $\Omega(\delta\varepsilon n)$ vertices. For all these vertices, the random walk algorithm to find minors succeeds with non-trivial probability.

2.1.3 The trapped case: local partitioning to the rescue

In this case, for almost all vertices $\|\mathbf{p}_{s,\ell}\|_2^2 \geq 1/n^\alpha$. The proofs of the (contrapositive of the) Cheeger inequality basically imply the existence of a set of low condutance cut P_s "around" s. By local partitioning methods such as those of Spielman-Teng and Anderson-

Chung-Lang [ST12, ACL06], we can actually find P_s in roughly n^α time. We expect our graph to basically decompose into $O(n^\alpha)$ sized components with few edges between them. Our algorithm can simply find these pieces P_s and run a planarity test on them. We refer to this as the *local search* procedure.

While the intuition is correct, the analysis is difficult. The main problem is that actual partitioning of the graph (into small components connected by low conductance cuts) is fundamentally iterative. It starts by finding a low conductance set P_{s_1}, then finding a low conductance set P_{s_2} in $\overline{P_{s_1}}$, then P_{s_3} in $\overline{P_{s_1} \cup P_{s_2}}$, and so on. In general, this requires conditions on the random walk behavior inside $\overline{\bigcup_{j<i} P_{s_j}}$. On the other hand, our algorithm and the trapped case condition only refer to random walk behavior in all of G. Furthermore, $\overline{\bigcup_{j<i} P_{s_j}}$ can be as small as $\Theta(\varepsilon n)$, and so we do expect the random walk behavior to be quite different.

The GR bipartiteness analysis surmounts this problem and performs such a decomposition, but their parameters do not work for us. Starting from a source vertex s, their analysis discovers P_s such that probabilities of reaching any vertex in P_s (from s) is roughly uniform *and* smaller than $1/\sqrt{n}$. On the other hand, we would like to discover all of P_s in $n^{O(\delta)}$ time so that we can run a full planarity test.

We employ a collection of tools, and use the methods of Kale-Peres-Seshadhri to analyze "projected" Markov Chains [KPS13]. In the analysis above, we have some set S ($\overline{\bigcup_{j<i} P_{s_j}}$) and want to find a low conductance set P completely contained in S. Moreover, we wish to discover P using random walks in G. We construct a Markov chain, M_S, with vertex set S, and include new transitions that correspond to walks in G whose intermediate vertices are not in S. Each such transition has an associated "cost," corresponding to the actual length in G. (GR also have a similar idea, although their Markov chain introduces extra vertices to track the length of the walk in G. This makes the analysis somewhat unwieldy, since low conductance cuts in M_S may include these extra vertices.)

Using bounds on the return time of random walks, we have relationships between the average length of a walk in G whose endpoints are in S and the corresponding length when "projected" to M_S. On average, an ℓ-length walk in G with endpoints in S corresponds to an $\ell|S|/n$-length walk in M_S. Roughly speaking, we hope that for many vertices s, an $\ell|S|/n$-length walk in M_S is trapped in a set of size n^α.

We employ the Lovász-Simonovits curve technique to produce a low conductance cut P_s in M_S [LS90a]. We can guarantee that all vertices in P_s are reachable with roughly $n^{-\alpha}$ probability from s through $\ell|S|/n$-length random walks in M_S. Using the average length correspondence between walks in M_S to G, we can make a similar statement in G - albeit with a longer length. We basically iterate over this entire argument to produce the decomposition into low conductance pieces.

In our analysis, we use the stratification itself to (implicitly) distinguish between the

leaky and trapped case. Stratification peels the graph into $1/\delta + O(1)$ strata. If a vertex s lies in a stratum numbered at least some fixed constant b, we can show that the algorithm finds a K_r-minor with s as the starting vertex. Thus, if at least (say) $n^{1-\delta}$ vertices lie in stratum b or higher, we are done. If s is in a low stratum, we have a lower bound on the random walks norm. This allows for local partitioning around s.

2.2 The algorithm

We are given a bounded degree graph $G = (V, E)$, with max degree d. We assume that $V = [n]$. We follow the standard adjacency list model of Goldreich-Ron for (random) access to the graph. This model allows an algorithm to sample u.a.r. vertices and perform *edge queries*. Given a pair $(v, i) \in [n] \times [d]$, the output of an edge query is the ith neighbor of v according to the adjacency list ordering. If the degree of v is smaller than i, the output is $\perp$.

In the algorithm, the phrase "random walk" refers to a lazy random walk on G. Given a current vertex, v, with probability $1/2$, the walk remains at v. With probability $1/2$, the procedure generates u.a.r. $i \in [d]$. It performs the edge query for (v, i). If the output is $\perp$, the walk remains at v, otherwise the walk visits the output vertex. This is a symmetric, ergodic Markov chain with a uniform stationary distribution.

Our main procedure `FindMinor`(G, ε, H), tries to find a H-minor in G. We prove that it succeeds with high probability if G is ε-far from being H-minor-free. There are three subroutines:

- `LocalSearch`(s): This procedure perform a small number of short random walks to find the piece described in §2.1.3. This produces a small subgraph of G, where an exact H-minor finding algorithm is used.
- `FindPath`(u, v, k, i): This procedure tries to find a path from u to v. The parameter i decides the length of the walk, and the procedure performs k walks from u and v. If any pair of these walks collide, this path is output.
- `FindBiclique`(s): This is the main procedure mostly as described in §2.1.1. It attempts to find a sufficiently large biclique minor. First, it generates seed sets A and B by performing random walks from s. Then, it calls `FindPath` on all pairs in $A \times B$.

We fix a collection of parameters.

- δ: An arbitrarily small constant.
- r: The number of vertices in H.
- ℓ: The random walk length. This will be $n^{5\delta}$.
- $\varepsilon_{\mathsf{CUTOFF}}$: $\varepsilon_{\mathsf{CUTOFF}} = n^{\frac{-\delta}{exp(2/\delta)}}$. If $\varepsilon < \varepsilon_{\mathsf{CUTOFF}}$, the algorithm just queries the whole graph.

• $\texttt{KKR}(F, H)$: This refers to an exact H-minor finding process (in F). For concreteness, we use $O(|V(G)|^2)$ procedure of Kawarabayashi-Kobayashi-Reed [KKR12].

$\texttt{FindMinor}(G, \varepsilon, H)$

1. If $\varepsilon < \varepsilon_{\textsf{CUTOFF}}$, query all of G, and output $\texttt{KKR}(G, H)$
2. Else
   (a) Repeat $\varepsilon^{-2} n^{35\delta r^2}$ times:
      i. Pick u.a.r $s \in V$
      ii. Call $\texttt{LocalSearch}(s)$ and $\texttt{FindBiclique}(s)$.

$\texttt{LocalSearch}(s)$

1. Initialize set $B = \emptyset$.
2. For $h = 1, \ldots, n^{7\delta r^2}$:
   (a) Perform $\varepsilon^{-1} n^{30\delta r^2}$ independent random walks of length h from s. Add all destination vertices to B.
3. Determine $G[B]$, the subgraph induced by B.
4. Run $\texttt{KKR}(G[B], H)$. If it returns an H-minor, output that and terminate.

$\texttt{FindBiclique}(s)$

1. For $i = 5r^2, \ldots, 1/\delta + 4$:
   (a) Perform $2r$ independent random walks of length $2^{i+1}\ell$ from s. Let the destinations of the first r walks be multiset A, and the destinations of the remaining walks be B.
   (b) For each $a \in A$, $b \in B$:
      i. Run $\texttt{FindPath}(a, b, n^{\delta(i+18)/2}, i)$
   (c) If all calls to $\texttt{FindPath}$ return a path, then let the collection of paths be the subgraph F. Run $\texttt{KKR}(F, H)$. If it returns an H-minor, output that and terminate.

$\texttt{FindPath}(u, v, k, i)$

1. Perform k random walks of length $2^i\ell$ from u and v.
2. If a walk from u and v terminate at the same vertex, return these paths. (Otherwise, return nothing.)

Theorem 2.2.1. *If G is ε-far from being H-minor-free, $\texttt{FindMinor}(G, \varepsilon, H)$ finds an H-minor of G with probability at least $2/3$. Furthermore, $\texttt{FindMinor}$ has a running time of $dn^{1/2+O(\delta r^2)} + d\varepsilon^{-2\exp(2/\delta)/\delta}$.*

The query complexity is fairly easy to compute. The total queries made in the $\texttt{LocalSearch}$ calls is $dn^{O(\delta r^2)}$. The main work happens in the calls of $\texttt{FindPath}$, within $\texttt{FindBiclique}$. Observe that k is set to $n^{\delta(i+18)/2}$, where $i \leq 1/\delta + 4$. This leads to the $\sqrt{n}$ in the final complexity. (In general, a setting of $\delta < 1/\log(\varepsilon^{-1} \log\log n)$ suffices for an $n^{1/2+o(1)}$ running time.)

Outline: There are a number of moving parts in the proof, which we relegate to their own subsections. We first develop the notion of R-returning walks and the stratification process in §2.3. In §2.4, we use these techniques to prove that `FindBiclique` discovers a sufficiently large biclique-minor in the leaky case. In §2.5, we prove a local partitioning lemma that will be used to handle the trapped case. Finally, in §2.6, we put the tools together to complete the proof of Theorem 2.2.1.

2.3 Returning walks and stratification

We introduce the concept of R-returning random walks for any $R \subseteq V$. These definitions are with respect to a fixed length ℓ.

Notation	Meaning	Where defined
$\boldsymbol{q}^{(i)}_{[R],s}$	R-returning probability vector	§2.3, Def 2.3.1
$q^{(i)}_{[R],s}(u)$	Probability of R-returning walk ending at $u \in R$	§2.3, Def 2.3.1
$\hat{q}^{(i)}_{[R_i],s}(u)$	Distribution on R_i induced by $\boldsymbol{q}^{(i)}_{[R_i],s}$	Def 2.3.10

Table 2.1: Stratification notation

Definition 2.3.1. *For any set of vertices R, $s \in R$, $u \in R$, and $i \in \mathbb{N}$, we define the* R-returning probability *as follows. We denote by $q^{(i)}_{[R],s}(u)$ the probability that a $2^i\ell$-length random walk from s ends at u, and encounters a vertex in R at every $j\ell^{th}$ step, for all $1 \leq j \leq 2^i$. The R-returning probability vector, denoted by $\boldsymbol{q}^{(i)}_{[R],s}$, is the $|R|$-dimensional vector of returning probabilities.*

Proposition 2.3.2. $q^{(i+1)}_{[R],s}(u) = \boldsymbol{q}^{(i)}_{[R],s} \cdot \boldsymbol{q}^{(i)}_{[R],u}$

Proof. We use the symmetry of (returning) random walks in G.

$$q^{(i+1)}_{[R],s}(u) = \sum_{w \in S} q^{(i)}_{[R],s}(w) q^{(i)}_{[R],w}(u) = \sum_{w \in R} q^{(i)}_{[R],s}(w) q^{(i)}_{[R],u}(w) = \boldsymbol{q}^{(i)}_{[R],s} \cdot \boldsymbol{q}^{(i)}_{[R],u}$$

□

Let M be the transition matrix of the lazy random walk on G. Let $\mathbb{P}_R$ be the $n \times |R|$ matrix on R, where each column is the unit vector for some $s \in R$. For any set U, we use $\mathbf{1}_U$ for the indicator vector on U. If no subscript is given, it is the all ones vector, for the appropriate dimension.

Proposition 2.3.3. $\boldsymbol{q}^{(i)}_{[R],s} = (\mathbb{P}_R^T M^\ell \mathbb{P}_R)^{2^i} \mathbf{1}_s$

Now for a critical lemma. We can lower bound the total probability of an R-returning random walk. If R contains at least a β-fraction of vertices, the average R-returning walk probability, for t returns, is at least β^t.

Lemma 2.3.4. $|R|^{-1}\sum_{s\in R}\|\boldsymbol{q}^{(i)}_{[R],s}\|_1 \geq (|R|/n)^{2^i}$

Proof. We will express $\sum_{s\in R}\|\boldsymbol{q}^{(i)}_{[R],s}\|_1 = \mathbf{1}^T(\mathbb{P}_R^T M^\ell \mathbb{P}_R)^{2^i}\mathbf{1}$. Let us first prove the lemma for $i = 0$. Observe that

$$\sum_{s\in R}\|\boldsymbol{q}^{(0)}_{[R],s}\|_1 = \mathbf{1}_R^T M^\ell \mathbf{1}_R = ((M^T)^{\ell/2}\mathbf{1}_R)^T(M^{\ell/2}\mathbf{1}_R) = \|M^{\ell/2}\mathbf{1}_R\|_2^2.$$

Since $M^{\ell/2}$ is a stochastic matrix, $\|M^{\ell/2}\mathbf{1}_R\|_1 = \|\mathbf{1}_R\|_1 = |R|$. By a standard norm inequality, $\|M^{\ell/2}\mathbf{1}_R\|_2^2 \geq \|M^{\ell/2}\mathbf{1}_R\|_1^2/n = |R|^2/n$. This completes the proof for $i = 0$.

Let $N = \mathbb{P}_R^T M^\ell \mathbb{P}_R$, which is a symmetric matrix. We have just proven that $\mathbf{1}^T N\mathbf{1} \geq |R|^2/n$. Let the eigenvalues of N be $1 \geq \lambda_1 \geq \lambda_2 \ldots \lambda_{|R|}$, with corresponding eigenvectors $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_s}$. We can express $\mathbf{1} = \sum_{k\leq|R|}\alpha_k\mathbf{u_k}$, where $\sum_k \alpha_k^2 = |R|$. Observe that $N^{2^i}\mathbf{1} = \sum_{k\leq|R|}\alpha_k\lambda_k^{2^i}\mathbf{u_k}$

Let $\mu_k = \alpha_k^2/\sum_j \alpha_j^2$, noting that $\sum_k \mu_k = 1$. We apply Jensen's inequality below.

$$\frac{\mathbf{1}^T N^{2^i}\mathbf{1}}{|R|} = \frac{\sum_k \alpha_k^2\lambda_k^{2^i}}{\sum_j \alpha_j^2} = \sum_k \mu_k\lambda_k^{2^i} \geq (\sum_k \mu_k\lambda_k)^{2^i}$$

For $i = 0$, we already proved that $\mathbf{1}^T N\mathbf{1}/|R| = \sum_k \mu_k\lambda_k \geq |R|/n$. We plug this bound to complete the proof for general i. □

2.3.1 Stratification

Stratification results in a collection of disjoint sets of vertices denoted by $S_0, S_1, \ldots$ which are called *strata*. The corresponding *residue* sets denoted by $R_0, R_1, \ldots$. The zeroth residue R_0 is initialized before stratification and subsequent residues are defined by the recurrence $R_i = R_0 \setminus \bigcup_{j<i} S_j$. The definitions and claims may seem technical, and the proofs are mostly norm manipulations. But these provide the tools to analyze our main algorithm.

Definition 2.3.5. *Suppose R_i has been constructed. A vertex $s \in R_i$ is placed in S_i if $\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_2^2 \geq 1/n^{\delta i}$.*

We have an upper bound for the length of R_i-returning walk vectors.

Claim 2.3.6. *For all $s \in R_i$ and $1 \leq j \leq i$, $\|\boldsymbol{q}^{(j)}_{[R_i],s}\|_2^2 \leq 1/n^{\delta(j-1)}$.*

Proof. Suppose $\exists j \leq i, \|\boldsymbol{q}^{(j)}_{[R_i],s}\|_2^2 > 1/n^{\delta(j-1)}$. By assumption, $s \in R_i \subseteq R_{j-1}$. An R_i-returning walk from s is also an R_{j-1}-returning walk. Thus, every entry of $\boldsymbol{q}^{(j)}_{[R_{j-1}],s}$ is at least that of $\boldsymbol{q}^{(j)}_{[R_i],s}$. So $\|\boldsymbol{q}^{(j)}_{[R_{j-1}],s}\|_2^2 \geq \|\boldsymbol{q}^{(j)}_{[R_i],s}\|_2^2 > 1/n^{\delta(j-1)}$. This implies that $s \in S_{j-1}$ or an earlier stratum, contradicting the assumption that $s \in R_i$. □

We prove an ℓ_∞ bound on the returning probability vectors. Note that we allow j to be $i + 1$ in the following bound.

Claim 2.3.7. *For all $s \in R_i$ and $2 \le j \le i+1$, $\|\boldsymbol{q}^{(j)}_{[R_i],s}\|_\infty \le 1/n^{\delta(j-2)}$.*

Proof. By Prop. 2.3.2, for any $v \in R_i$, $q^{(j)}_{[R_i],s}(v) = \boldsymbol{q}^{(j-1)}_{[R_i],s} \cdot \boldsymbol{q}^{(j-1)}_{[R_i],v}$. Note that $1 \le j-1 \le i$. By Cauchy-Schwartz and Claim 2.3.6, $q^{(j)}_{[R_i],s}(v) \le 1/n^{\delta(j-2)}$. □

As a consequence of these bounds, we are able to bound the amount of probability mass retained by R_i-returning walks.

Claim 2.3.8. *For all $s \in S_i$, $||\boldsymbol{q}^{(i+1)}_{[R_i],s}||_1 \ge n^{-\delta}$.*

Proof. Since $s \in S_i$, $||\boldsymbol{q}^{(i+1)}_{[R_i],s}||_2^2 \ge n^{-i\delta}$, and by Claim 2.3.7, $||\boldsymbol{q}^{(i+1)}_{[R_i],s}||_\infty \le n^{-\delta(i-1)}$. Since, $||\boldsymbol{q}^{(i+1)}_{[R_i],s}||_2^2 \le ||\boldsymbol{q}^{(i+1)}_{[R_i],s}||_1 ||\boldsymbol{q}^{(i+1)}_{[R_i],s}||_\infty$, we conclude $||\boldsymbol{q}^{(i+1)}_{[R_i],s}||_1 \ge n^{-i\delta} n^{\delta(i-1)} = n^{-\delta}$. □

We prove that most vertices lie in "early" strata.

Lemma 2.3.9. *Suppose $\varepsilon \ge \varepsilon_{\mathsf{CUTOFF}}$. At most $\varepsilon n/\log n$ vertices are in $R_{1/\delta+3}$.*

Proof. We prove by contradiction. Suppose that $R_{1/\delta+3}$ has at least $\varepsilon n/\log n$ vertices. The previous residue, $R_{1/\delta+2}$, is only bigger and thus $|R_{1/\delta+2}| \ge \varepsilon n/\log n$ as well. By Lemma 2.3.4,

$$|R_{1/\delta+2}|^{-1} \sum_{s \in R_{1/\delta+2}} \|\boldsymbol{q}^{(1/\delta+3)}_{[R_{1/\delta+2}],s}\|_1 \ge \left(\frac{\varepsilon}{\log n}\right)^{2^{1/\delta+3}}. \tag{2.1}$$

By averaging and Cauchy-Schwartz (to relate l_1 and l_2 norms),

$$||\boldsymbol{q}^{(1/\delta+3)}_{[R_{1/\delta+2}],s}\|_2^2 \ge n^{-1} \left(\frac{\varepsilon}{\log n}\right)^{2^{1/\delta+4}}. \tag{2.2}$$

By assumption, $\varepsilon \ge \varepsilon_{\mathsf{CUTOFF}} \ge n^{-\delta/\exp(1/\delta)}$. For sufficiently small δ,

$$\delta/\exp(1/\delta) < 2\delta/2^{1/\delta+4}$$

Thus, $\varepsilon \ge (\log n) n^{-2\delta/(2^{1/\delta+4})}$. Plugging into the RHS of the previous equation, $||\boldsymbol{q}^{(1/\delta+3)}_{[R_{1/\delta+2}],s}\|_2^2 \ge 1/n^{1+2\delta} = 1/n^{\delta(1/\delta+2)}$. This implies that $v \in S_{1/\delta+2}$ - a contradiction. □

2.3.2 The correlation lemma

The following lemma is an important tool in our analysis. Here is an intuitive explanation. Fix some $s \in S_i$. By Prop. 2.3.2, the probability $q^{(i+1)}_{[R_i],s}(v)$ is the correlation between the vectors $\boldsymbol{q}^{(i)}_{[R_i],s}$ and $\boldsymbol{q}^{(i)}_{[R_i],v}$. If many of these probabilities are large, then there are many v such that $\boldsymbol{q}^{(i)}_{[R_i],v}$ is correlated with $\boldsymbol{q}^{(i)}_{[R_i],s}$. We then expect many of these vectors are correlated among themselves.

Definition 2.3.10. *For $s \in R_i$, the distribution $\mathcal{D}_{s,i}$ has support R_i, and the probability of $u \in R_i$ is $\hat{q}^{(i+1)}_{[R_i],s}(v) = q^{(i+1)}_{[R_i],s}(v)/\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1$.*

Lemma 2.3.11. *Fix arbitrary $s \in R_i$.*

$$\mathbf{E}_{u_1,u_2\sim\mathcal{D}_{s,i}}[\boldsymbol{q}^{(i)}_{[R_i],u_1}\cdot\boldsymbol{q}^{(i)}_{[R_i],u_2}] \geq \frac{1}{\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1^2}\cdot\frac{\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_2^4}{\|\boldsymbol{q}^{(i)}_{[R_i],s}\|_2^2}$$

Proof.

$$\begin{aligned}
&\mathbf{E}_{u_1,u_2\sim\mathcal{D}_{s,i}}[\boldsymbol{q}^{(i)}_{[R_i],u_1}\cdot\boldsymbol{q}^{(i)}_{[R_i],u_2}]\\
&=\sum_{u_1,u_2\in R_i}\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1^{-2}q^{(i+1)}_{[R_i],s}(u_1)q^{(i+1)}_{[R_i],s}(u_2)\boldsymbol{q}^{(i)}_{[R_i],u_1}\cdot\boldsymbol{q}^{(i)}_{[R_i],u_2}\\
&=\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1^{-2}\sum_{u_1,u_2\in R_i}(\boldsymbol{q}^{(i)}_{[R_i],s}\cdot\boldsymbol{q}^{(i)}_{[R_i],u_1})(\boldsymbol{q}^{(i)}_{[R_i],s}\cdot\boldsymbol{q}^{(i)}_{[R_i],u_2})(\boldsymbol{q}^{(i)}_{[R_i],u_1}\cdot\boldsymbol{q}^{(i)}_{[R_i],u_2})\\
&=\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1^{-2}\sum_{u_1,u_2\in R_i}(\boldsymbol{q}^{(i)}_{[R_i],s}\cdot\boldsymbol{q}^{(i)}_{[R_i],u_1})(\boldsymbol{q}^{(i)}_{[R_i],s}\cdot\boldsymbol{q}^{(i)}_{[R_i],u_2})\sum_{w\in R_i}q^{(i)}_{[R_i],u_1}(w)q^{(i)}_{[R_i],u_2}(w))\\
&=\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1^{-2}\sum_{\substack{w\in R_i\\u_1,u_2\in R_i}}[(\boldsymbol{q}^{(i)}_{[R_i],s}\cdot\boldsymbol{q}^{(i)}_{[R_i],u_1})q^{(i)}_{[R_i],u_1}(w)][(\boldsymbol{q}^{(i)}_{[R_i],s}\cdot\boldsymbol{q}^{(i)}_{[R_i],u_2})q^{(i)}_{[R_i],u_2}(w)]\\
&=\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1^{-2}\sum_{w\in R_i}\left[\sum_{u\in R_i}(\boldsymbol{q}^{(i)}_{[R_i],s}\cdot\boldsymbol{q}^{(i)}_{[R_i],u})q^{(i)}_{[R_i],u}(w)\right]^2 \qquad (2.3)
\end{aligned}$$

We now write out $\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_2^2=\sum_{u\in R_i}q^{(i+1)}_{[R_i],s}(u)^2=\sum_{u\in R_i}(\boldsymbol{q}^{(i)}_{[R_i],s}\cdot\boldsymbol{q}^{(i)}_{[R_i],u})^2$, by Prop. 2.3.2. We expand further below. The only inequality is Cauchy-Schwartz.

$$\begin{aligned}
\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_2^2&=\sum_{u\in R_i}(\boldsymbol{q}^{(i)}_{[R_i],s}\cdot\boldsymbol{q}^{(i)}_{[R_i],u})\sum_{w\in R_i}q^{(i)}_{[R_i],s}(w)q^{(i)}_{[R_i],u}(w)\\
&=\sum_{w\in R_i}q^{(i)}_{[R_i],s}(w)\Big[\sum_{u\in R_i}(\boldsymbol{q}^{(i)}_{[R_i],s}\cdot\boldsymbol{q}^{(i)}_{[R_i],u})q^{(i)}_{[R_i],u}(w)\Big]\\
&\leq\sqrt{\sum_{w\in R_i}q^{(i)}_{[R_i],s}(w)^2}\sqrt{\sum_{w\in R_i}\Big[\sum_{u\in R_i}(\boldsymbol{q}^{(i)}_{[R_i],s}\cdot\boldsymbol{q}^{(i)}_{[R_i],u})q^{(i)}_{[R_i],u}(w)\Big]^2}\\
&=\|\boldsymbol{q}^{(i)}_{[R_i],s}\|_2\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1\sqrt{\mathbf{E}_{u_1,u_2\sim\mathcal{D}_{s_i}}[\boldsymbol{q}^{(i)}_{[R_i],u_1}\cdot\boldsymbol{q}^{(i)}_{[R_i],u_2}]}
\end{aligned}$$

The last line is because of (2.3). We rearrange and take squares to complete the proof. □

We can apply previous norm bounds to get an explicit lower bound. To see the significance of the following lemma, note that by Claim 2.3.6 and Cauchy-Schwartz, $\forall u_1,u_2\in R_i$, $\boldsymbol{q}^{(i)}_{[R_i],u_1}\cdot\boldsymbol{q}^{(i)}_{[R_i],u_2}\leq 1/n^{\delta(i-1)}$ (fairly close to the lower bound below).

Lemma 2.3.12. *Fix arbitrary $s\in S_i$.*

$$\mathbf{E}_{u_1,u_2\sim\mathcal{D}_{s,i}}[\boldsymbol{q}^{(i)}_{[R_i],u_1}\cdot\boldsymbol{q}^{(i)}_{[R_i],u_2}]\geq 1/n^{\delta(i+1)}$$

Proof. By Lemma 2.3.11, the LHS is at least $\frac{1}{\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1^2}\cdot\frac{\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_2^4}{\|\boldsymbol{q}^{(i)}_{[R_i],s}\|_2^2}$. Note that $\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1\leq 1$. By Def 2.3.5, $\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_2^2\geq 1/n^{\delta i}$. Since $s\in S_i\subseteq R_i$, by Claim 2.3.6, $\|\boldsymbol{q}^{(i)}_{[R_i],s}\|_2^2\leq 1/n^{\delta(i-1)}$. □

2.4 Analysis of FindBiclique

This is the central theorem of our analysis. It shows that the FindBiclique(s) procedure discovers a $K_{r,r}$ minor with non-trivial probability when s is in a sufficiently high stratum.

Theorem 2.4.1. *Suppose $s \in S_i$, for $5r^2 \le i \le 1/\delta + 3$. The probability that the paths discovered in* FindBiclique(s) *contain a $K_{r,r}$ minor is at least $n^{-4\delta r^2}$.*

Theorem 2.4.1 is proved in §2.4.5. Towards the proof, we will need multiple tools. In §2.4.1, we perform a standard calculation to bound the success probability of FindPath. In §2.4.2, we use this bound to show that the sets A and B sampled by FindBiclique are successfully connected by paths as discovered by FindPath. In §2.4.3, we argue that the intersections between these paths are "well-behaved" enough to induce a $K_{r,r}$ minor.

We note that the $\sqrt{n}$ in the final running time comes from the calls to FindPath in FindBiclique.

2.4.1 The procedure FindPath

For convenience, we reproduce the procedure FindPath. It is a relatively straightforward application of a birthday paradox argument for bidirectional path finding.

FindPath(u, v, k, i)

1. Perform k random walks of length $2^i\ell$ from u and v.
2. If a walk from u and v terminate at the same vertex, return these paths.

Lemma 2.4.2. *Let c be a sufficiently large constant. Consider $u, v \in R_i$. Suppose there exist $\alpha \le \beta$ such that $\max(\|\boldsymbol{q}^{(i)}_{[R_i],u}\|_2^2, \|\boldsymbol{q}^{(i)}_{[R_i],v}\|_2^2) \le 1/n^\alpha$ and $\boldsymbol{q}^{(i)}_{[R_i],u} \cdot \boldsymbol{q}^{(i)}_{[R_i],v} \ge 1/2n^\beta$. Then, with $k \ge cn^{\beta/2+4(\beta-\alpha)}$, FindPath$(u, v, k, i)$ returns an R_i-returning path of length $2^{i+1}\ell$ with probability $\ge 2/3$.*

Proof. First, define $W = \{w | q^{(i)}_{[R_i],u}(w)/q^{(i)}_{[R_i],v}(w) \in [1/(8n^{\beta-\alpha}), 8n^{\beta-\alpha}]\}$.

$$\sum_{w \notin W} q^{(i)}_{[R_i],u}(w) q^{(i)}_{[R_i],v}(w) \le (8n^{\beta-\alpha})^{-1} \sum_{w \notin W} \max(q^{(i)}_{[R_i],u}(w), q^{(i)}_{[R_i],v}(w))^2$$
$$\le (8n^{\beta-\alpha})^{-1}(\|\boldsymbol{q}^{(i)}_{[R_i],u}\|_2^2 + \|\boldsymbol{q}^{(i)}_{[R_i],v}\|_2^2) \le 1/4n^\beta$$

Therefore, $\sum_{w \in W} q^{(i)}_{[R_i],u}(w) q^{(i)}_{[R_i],v}(w) \ge 1/4n^\beta$.

For $a, b \le k$, let $X_{a,b}$ be the indicator for the following event: the ath $2^i\ell$-length random walk from u is an R_i-returning walk that ends at some $w \in W$, and the bth random walk from v is also R_i-returning, ending at the same w. Let $X = \sum_{a,b \le k} X_{a,b}$. Observe that the probability that FindPath(u, v, k, i) returns a path is at least $\Pr[X > 0]$.

We can bound

$$\mathbf{E}[\sum_{a,b\leq k} X_{a,b}] = k^2 \sum_{w\in W} q^{(i)}_{[R_i],u}(w) q^{(i)}_{[R_i],v}(w) \geq k^2/4n^\beta \geq (c^2/4)n^{8(\beta-\alpha)}.$$

Let us now bound the variance. First, let us expand out the expected square.

$$\begin{aligned}
&\mathbf{E}[(\sum_{a,b} X_{a,b})^2] \\
&= \sum_{a,b} \mathbf{E}[X^2_{a,b}] + 2\sum_{a\neq a',b} \mathbf{E}[X_{a,b}X_{a',b}] + 2\sum_{a,b\neq b'} \mathbf{E}[X_{a,b}X_{a',b}] + 2\sum_{a\neq a',b\neq b'} \mathbf{E}[X_{a,b}X_{a',b'}] \qquad (2.4)
\end{aligned}$$

Observe that $X^2_{a,b} = X_{a,b}$. Furthermore, for $a \neq a', b \neq b'$, by independence of the walks, $\mathbf{E}[X_{a,b}X_{a',b'}] = \mathbf{E}[X_{a,b}]\mathbf{E}[X_{a',b'}]$. (This term will cancel out in the variance.) By symmetry, $\sum_{a\neq a',b} \mathbf{E}[X_{a,b}X_{a',b}] \leq k^3\mathbf{E}[X_{1,1}X_{2,1}]$ (and analogously for the third term in (2.4)). Plugging these in and expanding out the $\mathbf{E}[X]^2$,

$$\mathbf{var}[X] \leq \mathbf{E}[X] + 2k^3\mathbf{E}[X_{1,1}X_{2,1}] + 2k^3\mathbf{E}[X_{1,1}X_{1,2}]$$

Note that $X_{1,1}X_{2,1} = 1$ when the first and second walks from u end at the same vertex where the first walk from v ends. Thus, $\mathbf{E}[X_{1,1}X_{2,1}] = \sum_{w\in W} q^{(i)}_{[R_i],u}(w)^2 q^{(i)}_{[R_i],v}(w)$. Since $w \in W$, we have $q^{(i)}_{[R_i],u}(w)/8n^{\beta-\alpha} \leq q^{(i)}_{[R_i],v}(w) \leq 8n^{\beta-\alpha} q^{(i)}_{[R_i],u}(w)$. Plugging this bound in,

$$\begin{aligned}
2k^3\mathbf{E}[X_{1,1}X_{2,1}] &\leq 16k^3n^{\beta-\alpha} \sum_{w\in W} q^{(i)}_{[R_i],u}(w)^3 \\
&\leq 16k^3n^{\beta-\alpha}[\sum_{w\in W} q^{(i)}_{[R_i],u}(w)^2]^{3/2} \qquad (l_2\text{-}l_3 \text{ norm inequality}) \\
&= 16n^{\beta-\alpha}[k^2 \sum_{w\in W} q^{(i)}_{[R_i],u}(w)^2]^{3/2}
\end{aligned}$$

We can apply the bound $q^{(i)}_{[R_i],u}(w) \leq 8n^{\beta-\alpha} q^{(i)}_{[R_i],v}(w)$.

$$\begin{aligned}
2k^3\mathbf{E}[X_{1,1}X_{2,1}] &\leq 16n^{\beta-\alpha}[k^2 \sum_{w\in W} q^{(i)}_{[R_i],u}(w) q^{(i)}_{[R_i],v}(w) \cdot 8n^{\beta-\alpha}]^{3/2} \\
&\leq 512n^{5(\beta-\alpha)/2}[k^2 \sum_{w\in W} q^{(i)}_{[R_i],u}(w) q^{(i)}_{[R_i],v}(w)]^{3/2} \qquad (2.5)
\end{aligned}$$

Note that $k^2 \sum_{w\in W} q^{(i)}_{[R_i],u}(w) q^{(i)}_{[R_i],v}(w)]^{3/2}$ is exactly $\mathbf{E}[X]$. We have previously bounded $\mathbf{E}[X] \geq (c^2/4)n^{8(\beta-\alpha)}$. Thus, $512n^{5(\beta-\alpha)/2} \leq \mathbf{E}[X]^{1/2}/(c/100)$. Applying the bounds in (2.5), we deduce that

$$2k^3\mathbf{E}[X_{1,1}X_{2,1}] \leq (\mathbf{E}[X]^{1/2}/(c/100))(\mathbf{E}[X]^{3/2}) = \mathbf{E}[X]^2/(c/100)$$

We get an identical bound for $2k^3\mathbf{E}[X_{1,1}X_{1,2}]$. Putting it all together, we can prove that $\mathbf{var}[X] \leq 4\mathbf{E}[X]^2/c'$, for $c' = \Theta(c)$. An application of Chebyshev proves that $\Pr[X > 0] > 2/3$. □

2.4.2 The procedure `IdealFindBiClique`

We describe an "ideal" variant of `FindBiclique` below. It is not possible to directly implement `IdealFindBiClique`. On the other hand, we can directly analyze the probability that it produces a $K_{r,r}$-minor. We will eventually prove that the `FindBiclique` procedure can efficiently simulate `IdealFindBiClique`.

`IdealFindBiClique`(s)

1. For $i = 5r^2, \ldots, 1/\delta + 4$:
   (a) Perform $2r$ independent draws from the distribution $\mathcal{D}_{s,i}$ (Def 2.3.10). Let the first r draws be multiset A, and the remaining draws be B.
   (b) For each $a \in A$, $b \in B$:
      i. Run `FindPath`$(a, b, n^{\delta(i+18)/2}, i)$
   (c) If all calls to `FindPath` return a path, then let the collection of paths be the subgraph F. Run `KKR`(F, H). If it returns an H-minor, output that and terminate.

The next lemma asserts that `IdealFindBiClique` finds (with non-trivial probability) paths between all pairs of vertices between two sets of r vertices. Ignoring the gnarly problem of the paths intersecting internally, this structure looks like a $K_{r,r}$-minor. Lemma 2.4.2 gives bounds on finding a single path between one pair of vertices, whose truncated random walk vectors satisfy some technical conditions. Somewhat naively, we could hope to find two sets of vertices A, B where all pairs in $A \times B$ satisfied these conditions. Then, on applying Lemma 2.4.2 for each pair, we could find a subgraph that looks like a $K_{r,r}$ minor.

It turns out that if A and B are themselves generated by performing random walks from a fixed vertex, the probability conditions of Lemma 2.4.2 hold "on average" for the pairs in $A \times B$. This crucially uses the correlation lemma. The proof of the next lemma further shows that this average condition suffices to lower bound the success probability of `FindBiclique`. We can basically assume that each call of `FindPath` within `IdealFindBiClique` has an independent success probability.

Lemma 2.4.3. *Suppose $s \in S_i$, for some $i \leq 1/\delta + 4$. With probability $(4n^{2\delta})^{-r^2}$, the calls to* `FindPath` *in* `IdealFindBiClique`(s) *output paths from every $a \in A$ to every $b \in B$, where each path is an R_i-returning walk of length $2^{i+1}\ell$.*

Proof. Recall that each element in $A \cup B$ is drawn according to $\hat{q}^{(i+1)}_{[R_i],s}(u)$. For any $a, b \in V$, let $\tau_{a,b}$ be the probability that `FindPath`$(a, b, n^{\delta(i+18)/2}, i)$ succeeds in finding an R_i-returning walk between a and b (of length $2^{i+1}\ell$). The probability of success for `FindBiclique`(s) conditioned

on $A, B \subseteq R_i$ is at least

$$\begin{aligned}
&\sum_{A\in R_i^r}\sum_{B\in R_i^r}\prod_{a\in A}\hat{q}^{(i+1)}_{[R_i],s}(a)\prod_{b\in B}\hat{q}^{(i+1)}_{[R_i],s}(b)\tau_{a,b}\\
&=\sum_{A\in R_i^r}\sum_{B\in R_i^r}\prod_{a\in A}\hat{q}^{(i+1)}_{[R_i],s}(a)\Big(\prod_{b\in B}\hat{q}^{(i+1)}_{[R_i],s}(b)\Big)\Big(\prod_{b\in B}\tau_{a,b}\Big)\\
&=\sum_{B\in R_i^r}\Big(\prod_{b\in B}\hat{q}^{(i+1)}_{[R_i],s}(b)\Big)\sum_{A\in R_i^r}\prod_{a\in A}\Big[\hat{q}^{(i+1)}_{[R_i],s}(a)\Big(\prod_{b\in B}\tau_{a,b}\Big)\Big]\\
&=\sum_{B\in R_i^r}\prod_{b\in B}\hat{q}^{(i+1)}_{[R_i],s}(b)\Big(\sum_{a\in R_i}\hat{q}^{(i+1)}_{[R_i],s}(a)\prod_{b\in B}\tau_{a,b}\Big)^r.
\end{aligned}$$

Observe that $\prod_{b\in B}\hat{q}^{(i+1)}_{[R_i],s}(b)$ is a probability distribution over R_i^r. By Jensen, we lower bound.

$$\begin{aligned}
&\sum_{B\in R_i^r}\prod_{b\in B}\hat{q}^{(i+1)}_{[R_i],s}(b)\Big(\sum_{a\in R_i}\hat{q}^{(i+1)}_{[R_i],s}(a)\prod_{b\in B}\tau_{a,b}\Big)^r\\
&\qquad\qquad\geq\Big[\sum_{B\in R_i^r}\Big(\prod_{b\in B}\hat{q}^{(i+1)}_{[R_i],s}(b)\Big)\sum_{a\in R_i}\hat{q}^{(i+1)}_{[R_i],s}(a)\prod_{b\in B}\tau_{a,b}\Big]^r \qquad (2.6)
\end{aligned}$$

We manipulate and expand the right hand side of (2.6) further.

$$\begin{aligned}
\text{RHS of (2.6)} &= \Big[\sum_{a\in R_i}\sum_{B\in R_i^r}\hat{q}^{(i+1)}_{[R_i],s}(a)\Big(\prod_{b\in B}\hat{q}^{(i+1)}_{[R_i],s}(b)\Big)\Big(\prod_{b\in B}\tau_{a,b}\Big)\Big]^r\\
&= \Big[\sum_{a\in R_i}\hat{q}^{(i+1)}_{[R_i],s}(a)\sum_{B\in R_i^r}\prod_{b\in B}\hat{q}^{(i+1)}_{[R_i],s}(b)\tau_{a,b}\Big]^r\\
&= \Big[\sum_{a\in R_i}\hat{q}^{(i+1)}_{[R_i],s}(a)\Big(\sum_{b\in R_i}\hat{q}^{(i+1)}_{[R_i],s}(b)\tau_{a,b}\Big)^r\Big]^r\\
&\geq \Big[\sum_{a\in R_i}\sum_{b\in R_i}\hat{q}^{(i+1)}_{[R_i],s}(a)\hat{q}^{(i+1)}_{[R_i],s}(b)\tau_{a,b}\Big]^{r^2} \quad \text{(Jensen)}\\
&= \Big[\mathbf{E}_{a,b\sim\mathcal{D}_{s,i}}[\tau_{a,b}]\Big]^{r^2}. \qquad (2.7)
\end{aligned}$$

Towards lower bounding $\tau_{a,b}$, we first lower bound $\boldsymbol{q}^{(i)}_{[R_i],a}\cdot\boldsymbol{q}^{(i)}_{[R_i],b}$. By Lemma 2.3.12, $\mathbf{E}_{a,b}[\boldsymbol{q}^{(i)}_{[R_i],a}\cdot\boldsymbol{q}^{(i)}_{[R_i],b}]\geq 1/n^{\delta(i+1)}$. Claim 2.3.6 states that $\|\boldsymbol{q}^{(i)}_{[R_i],a}\|_2^2\leq 1/n^{\delta(i-1)}$ and $\|\boldsymbol{q}^{(i)}_{[R_i],b}\|_2^2\leq 1/n^{\delta(i-1)}$. By Cauchy-Schwartz, $\boldsymbol{q}^{(i)}_{[R_i],a}\cdot\boldsymbol{q}^{(i)}_{[R_i],b}\leq 1/n^{\delta(i-1)}$. Let p be the probability (over a,b) that $\boldsymbol{q}^{(i)}_{[R_i],a}\cdot\boldsymbol{q}^{(i)}_{[R_i],b}\geq 1/2n^{\delta(i+1)}$.

$$1/n^{\delta(i+1)}\leq\mathbf{E}_{a,b}[\boldsymbol{q}^{(i)}_{[R_i],a}\cdot\boldsymbol{q}^{(i)}_{[R_i],b}]\leq(1-p)/2n^{\delta(i+1)}+p/n^{\delta(i-1)}$$

Thus, $p\geq 1/2n^{2\delta}$.

By Claim 2.3.6, for every $a\in R_i$, $\|\boldsymbol{q}^{(i)}_{[R_i],a}\|_2^2\leq 1/n^{\delta(i-1)}$ (similarly for $b\in R_i$). Suppose $\boldsymbol{q}^{(i)}_{[R_i],a}\cdot\boldsymbol{q}^{(i)}_{[R_i],b}\geq 1/2n^{\delta(i+1)}$. Let us apply Lemma 2.4.2, with $\alpha=\delta(i-1)$ and $\beta=\delta(i+1)$. The number of walks performed in calls made to `FindPath` by the `FindBiclique` procedure, (the value of k) is $n^{\delta(i+18)/2}$. Note that $\delta(i+18)/2>\delta(i+1)/2+8\delta=\beta/2+4(\alpha-\beta)$. By

Lemma 2.4.2, $\tau_{a,b} \geq 1/2$. As argued in the previous paragraph, this will happen with probability $1/2n^{2\delta}$ (over the choice of $a, b \sim \mathcal{D}_{s,i}$). We plug in (2.7) and deduce that the probability of success is at least $(1/4n^{2\delta})^{r^2}$. □

2.4.3 Criteria for `IdealFindBiClique` to reveal a minor

Fix $s \in S_i$, as in Lemma 2.4.3. This lemma only asserts that all pairs in $A \times B$ are connected by `IdealFindBiClique` (with non-trivial probability). We need to argue that these paths will actually induce a $K_{r,r}$-minor.

The overall proof is highly technical, and we provide Tab. 2.2 for reference.

As in Lemma 2.4.3, let us focus on the ith iteration within `IdealFindBiClique`. For every $a \in A, b \in B$, there is a call to $\texttt{FindPath}(a, b, n^{\delta(i+18)/2}, i)$. Within each such call, a set of walks is performed from both a and b, with the hope of connecting a to b. We use a, a' (resp. b, b') to refer to elements in A (resp. B).

Notation	Meaning	Where defined
$\boldsymbol{W}_a^b$	R_i-returning walks from a in $\texttt{FindPath}(a, b, \cdot, \cdot)$	Def 2.4.4
$\boldsymbol{W}_a$	$\bigcup_{b \in B} \boldsymbol{W}_a^b$	Def 2.4.4
$P_{a,b}$	A path from a to b discovered in `FindPath`	Def 2.4.4
τ	Middle step of the walks in $\boldsymbol{W}_a$ or $\boldsymbol{W}_b$	Def 2.4.5
$\boldsymbol{\sigma}_{s,S,t}$	Prob. vector of S-returning t-length walk from s	§2.4.4, Def 2.4.7

Table 2.2: Bad Intersections notation

Definition 2.4.4.

- *Let $\boldsymbol{W}_a^b$ denote the set of R_i-returning walks from a performed in the call to* $\texttt{FindPath}(a, b, n^{\delta(i+18)/2}, i)$.
- *Let $\boldsymbol{W}_a$ denote the set of all vertices in $\bigcup_{b \in B} \boldsymbol{W}_a^b$.*
- *Let $P_{a,b}$ be a single path from a to b discovered by* `FindPath` $(a, b, n^{\delta(i+18)/2}, i)$*, that consists of a walk in $\boldsymbol{W}_a^b$ and a walk $\boldsymbol{W}_b^a$ that end at the same vertex. If there are many possible such paths, pick the lexicographically smallest.*

We stress that walks in $\boldsymbol{W}_a^b$ do not necessarily end at b, and come from a distribution independent of b (but we wish to track the specific call of `FindPath` where these walks were performed). Note that $\boldsymbol{W}_b^a$ is the set of R_i-returning walks starting from b performed in the same call.

Note that any of the paths/sets described above could be empty. We will think of paths as sequences, rather than sets, since the order in which the path is constructed is relevant. For any path, P, we use $P(t)$ to denote the t-th element in the sequence. We use $P(\geq t)$ to denote

the sequence of elements with index at least t. When we refer to intersections of paths being empty/non-empty, we refer to sets induced by the corresponding sequences.

For $s \in S_i$, conditioned on $A, B \subseteq R_i$, Lemma 2.4.3 gives a lower bound on $\Pr[\forall a \in A, b \in B, (P_{a,b} \neq \emptyset)]$. We will now define some *bad* events that interfere with minor structure.

Recall that A and B are multisets (it is convenient to think of them as sequences). The same vertex may appear multiple times in $A \cup B$, but we think of each occurrence as a distinct multiset element. Therefore, when we speak of equality of vertices, we refer to vertices at the same index in A (or B). By definition, elements in A are disjoint from B.

Definition 2.4.5. *The following events are referred to as* bad events *of Type 1, 2, or 3. We set* $\tau = 2^{i-1}\ell$.

1. $\exists a, b, c \in A \cup B$, $c \notin \{a, b\}$, *such that* $\boldsymbol{W}_c \cap P_{a,b} \neq \emptyset$.
2. $\exists a, b, b'$ *(all distinct) such that* $\exists W \in \boldsymbol{W}_a^b$ *where* $W(\geq \tau) \cap P_{a,b'} \neq \emptyset$. *(Or,* $\exists a, a' \in A, b \in B$, *all distinct, such that* $\exists W \in \boldsymbol{W}_b^a$ *where* $W(\geq \tau) \cap P_{a',b} \neq \emptyset$.*)*
3. $\exists a, b, W_a \in \boldsymbol{W}_a^b, W_b \in \boldsymbol{W}_b^a$ *such that* W_a, W_b *end at the same vertex and* $\exists t_1, t_2$ *such that* $\min(t_1, t_2) \leq \tau$ *and* $W_a(t_1) = W_b(t_2)$.

For clarity, let us express the above bad events in plain English. Note that τ is the index of the midpoint of the walks, so it splits walks into halves.

1. A walk from $c \in A \cup B$ intersects $P_{a,b}$, where $c \neq a, b$.
2. The second half of a walk in $\boldsymbol{W}_a^b$, which starts from a, intersects $P_{a,b'}$ for $b \neq b'$. Or, the second half of a walk in $\boldsymbol{W}_b^a$, which starts at b, intersects $P_{a',b}$ for $a \neq a'$.
3. A walk in $\boldsymbol{W}_a^b$ and a walk in $\boldsymbol{W}_b^a$ intersect at least twice. Note that this is a pair of walks, one from a and the other from b. The first intersection is in the first half of either of the walks. The walks also end at the same vertex.

Claim 2.4.6. *If all* $P_{a,b}$ *sets are non-empty and there is no bad event, then* $\bigcup_{a,b} P_{a,b}$ *contains a* $K_{r,r}$*-minor.*

Proof. Each $P_{a,b}$ is formed by $W_a \in \boldsymbol{W}_a^b$ and $W_b \in \boldsymbol{W}_b^a$ that end at the same vertex. Since there is no Type 3 bad event, $W_a(\leq \tau)$ is disjoint from W_b (and vice versa). Since $P_{a,b}$ is not necessarily a simple path, let us remove all self-intersections to form a simple path $P'_{a,b}$. We construct three vertex disjoint contiguous subpaths of $P'_{a,b}$ (also refer to Fig. **??**).

1. $Q_{a,b}$ is the contiguous subpath of $P'_{a,b}$ contained in $W_a(\leq \tau)$. This is the "first half" of the walk from a that forms $P_{a,b}$.

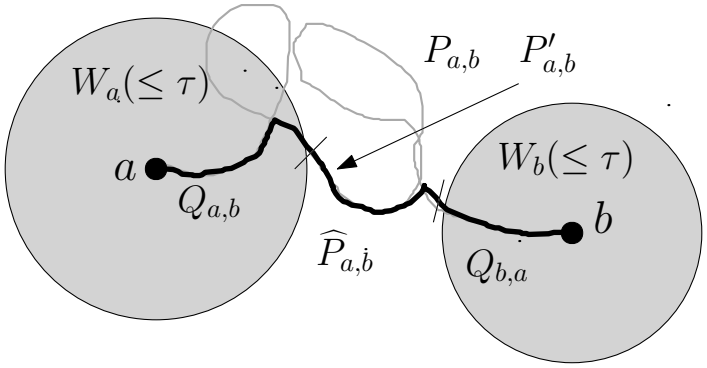

Figure 2.1: This figure shows the various subpaths defined in the proof of Claim 2.4.6. The simple path $P'_{a,b}$ (the thick path) from a to b induced by $P_{a,b}$ (in light gray). This path is broken into three contiguous subpaths: the portion "close to" a, the portion close to b, and the remainder.

2. $Q_{b,a}$ is the contiguous subpath of $P'_{a,b}$ contained in $W_b(\leq \tau)$. This is the "first half" of the walk from b that forms $P_{a,b}$.

3. $\widehat{P_{a,b}}$ is the contiguous subpath of $P'_{a,b}$ formed by removing vertices of $Q_{a,b} \cup Q_{b,a}$. Note that $\widehat{P_{a,b}} \subseteq W_a(> \tau) \cup W_b(> \tau)$.

In each bullets below, we state a disjointness condition on all the paths defined above. Each statement is followed by its corresponding proof. Subsequently, we will show how these disjointness conditions imply the existence of the desired minor.

We consider $a, a' \in A$ and $b, b' \in B$, where the elements in A (or B) might be equal. In what follows, when we say paths intersect, we mean that they share a common vertex. (Thus, disjoint paths always means vertex disjoint paths.) The idea is to show that whenever the disjointness condition fails, a bad intersection must occur.

- If $a \neq a'$, $Q_{a,b} \cap Q_{a',b'} = \emptyset$. If $b \neq b'$, $Q_{b,a} \cap Q_{b',a'} = \emptyset$.

  Consider the first statement. (Note that we allow $b = b'$.) Suppose $Q_{a,b} \cap Q_{a',b'} \neq \emptyset$. Observe that $Q_{a,b} \subseteq \boldsymbol{W}_a$ and $Q_{a',b'} \subseteq P_{a',b'}$. So $\boldsymbol{W}_a \cap P_{a',b'} \neq \emptyset$, implying a Type 1 bad event. Contradiction. The second statement follows analogously.

- $Q_{a,b} \cap Q_{b',a'} = \emptyset$.

  First, suppose that $a = a'$ and $b = b'$. As argued in the beginning of the proof, $Q_{a,b}$ and $Q_{b,a}$ are vertex disjoint. Now, consider the case $a \neq a'$. As before, $Q_{a,b} \subseteq \boldsymbol{W}_a$ and $Q_{b',a'} \subseteq P_{a',b'}$.

  Since no Type 1 bad events occur, $\boldsymbol{W}_a \cap P_{a',b'} = \emptyset$. The case $b \neq b'$ is analogous.

- If $a \neq a'$ or $b \neq b'$, $\widehat{P_{a,b}} \cap P_{a',b'} = \emptyset$. For contradiction's sake, suppose $\widehat{P_{a,b}} \cap P_{a',b'} \neq \emptyset$. We will show that some bad intersection occurs. Wlog, assume $a \neq a'$ (and b may or not be the same as b'). Note that $\widehat{P_{a,b}} \subseteq W_a(> \tau) \cup W_b(> \tau)$, where $W_a \in \boldsymbol{W}_a^b$ and $W_b \in \boldsymbol{W}_b^a$. Either $W_a(> \tau)$ or $W_b(> \tau)$ intersects $P_{a',b'}$, leading to two cases.
  - If $W_a(> \tau) \cap P_{a',b'} \neq \emptyset$, then $\boldsymbol{W}_a \cap P_{a',b'} \neq \emptyset$. This is a Type 1 bad event.
  - If $W_b(> \tau) \cap P_{a',b'} \neq \emptyset$: If $b \neq b'$, then $\boldsymbol{W}_b \cap P_{a',b'} \neq \emptyset$. This is a Type 1 bad event. If $b = b'$, then we have $W_b(> \tau) \cap P_{a',b} \neq \emptyset$. Since $W_b \in \boldsymbol{W}_b^a$ (for $a \neq a'$), $\boldsymbol{W}_b \cap P_{a'}b \neq \emptyset$. This is a Type 2 bad event.

We construct the minor. Let $C(a) = \bigcup_{b \in B} Q_{a,b}$ and $C(b) = \bigcup_{a \in A} Q_{b,a}$. Each $C(a), C(b)$ forms a connected subgraph. By the disjointness properties of the $Q_{a,b}$ sets, all the $C(a), C(b)$ sets/subgraphs are vertex disjoint. Note that $\widehat{P_{a,b}}$ is disjoint from all other $P_{a',b'}$ paths *and* all the $C(a), C(b)$ sets. (We construct $\widehat{P}_{a,b}$ to be disjoint from $Q_{a,b}$ and $Q_{b,a}$ in the first paragraph. Every other $Q_{a',b'}$ is contained in $P_{a',b'}$.) Moreover, each $\widehat{P}_{a,b}$ has an edge to $C(a)$ and $C(b)$, since it is contained in $P_{a,b}$. Thus, we have disjoint paths from each $C(a)$ to $C(b)$, which gives a $K_{r,r}$-minor. □

2.4.4 The probabilities of bad events

In this section, we bound the probability of bad events, as detailed in Def 2.4.5. As before, we fix $s \in S_i$.

We require some technical definitions of random walk probabilities.

Definition 2.4.7. *Let $\sigma_{s,S,t}(v)$ be the probability of a walk from s to v of length t being S-returning. (We allow $\ell \nmid t$ [1], and require that the walk encounters S at every $j\ell$-th step, for $j \leq \lfloor t/\ell \rfloor$.)*

We use $\boldsymbol{\sigma}_{s,S,t}$ to denote the vector of these probabilities. More generally, given any distribution vector $\boldsymbol{x}$ on V, $\boldsymbol{\sigma}_{\boldsymbol{x},S,t}$ denotes the vector of S-returning walk probabilities at time t.

We stress that this is not a conditional probability. Note that if $t = 2^i\ell$, then $\boldsymbol{\sigma}_{s,S,t} = \boldsymbol{q}^{(i)}_{[S],s}$. We show some simple propositions on these vectors. Let $\mathbb{I}_S$ denote the $n \times n$ matrix that preserves all coordinates in S and zeroes out other coordinates.

Proposition 2.4.8. *The vector $\boldsymbol{\sigma}_{\boldsymbol{x},S,t}$ evolves according to the following recurrence. Firstly, $\boldsymbol{\sigma}_{\boldsymbol{x},S,0} = \boldsymbol{x}$. For $t \geq 1$ such that $\ell \nmid t$, $\boldsymbol{\sigma}_{\boldsymbol{x},S,t} = M\boldsymbol{\sigma}_{\boldsymbol{x},S,t-1}$. For $t \geq 1$ such that $\ell \mid t$, $\boldsymbol{\sigma}_{\boldsymbol{x},S,t} = \mathbb{I}_S M \boldsymbol{\sigma}_{\boldsymbol{x},S,t-1}$*

Proposition 2.4.9. *For all $\boldsymbol{x}$ and all $t \geq 1$, $\|\boldsymbol{\sigma}_{\boldsymbol{x},S,t}\|_\infty \leq \|\boldsymbol{\sigma}_{\boldsymbol{x},S,t-1}\|_\infty$.*

[1] this means ℓ does not divide t.

Proof. Since M is a symmetric random walk matrix, it computes the "new" value at a vertex by averaging the values of the neighbors (and itself). This can never increase the maximum value. Furthermore, $\mathbb{I}_S$ only zeroes out some coordinates. This proves the proposition. □

In what follows, we fix the walk length to $2^i\ell$. To reduce clutter, we drop notational dependencies on this length.

Definition 2.4.10. *The distribution of $2^i\ell$-length walks from u is denoted $\mathcal{W}_u$. For any walk W, $W_u(t)$ denotes the t-th vertex of the walk. The Boolean predicate $\rho(W_u)$ is true if W_u is R_i-returning.*

Recall that $\mathcal{D}_{s,i}$ is the distribution with support R_i, where the probability of $u \in R_i$ is $\hat{q}^{(i+1)}_{[R_i],s}(v)$ (Def 2.3.10). This is the distribution that A, B are drawn from in `IdealFindBiClique`. We set i to be such that $s \in R_i$. Since i is fixed, we will simply write this as $\mathcal{D}_s$.

Claim 2.4.11. *For any $F \subseteq V$:*

1. $$\Pr_{a\sim\mathcal{D}_s, W_a\sim\mathcal{W}_a}[\rho(W_a) \wedge W_a \cap F \neq \emptyset] \leq 2^i\ell|F|/(n^{\delta(i-1)}\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1).$$

2. *For any $a \in R_i$,*
$$\Pr_{W_a\sim\mathcal{W}_a}[\exists t \geq \tau \mid \rho(W_a) \wedge W_a(t) \in F] \leq 2^i\ell|F|/n^{\delta(i-2)}$$

Proof. We prove the first part. Let $\boldsymbol{x} = \boldsymbol{\sigma}_{s,R_i,i+1}$ be the probability vector corresponding to $\mathcal{D}_s$. So $\|\boldsymbol{x}\|_\infty = \|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_\infty/\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1$. We proceed with a union bound over F and the walk length and write.

$$\Pr_{a\sim\mathcal{D}_s, W_a\sim\mathcal{W}_a}[\rho(W_a) \wedge W_a \cap F \neq \emptyset] \leq \sum_{t\leq 2^i\ell}\sum_{v\in F} \Pr_{a\sim\mathcal{D}_s, W_a\sim\mathcal{W}_a}[\rho(W_a) \wedge W_a(t) = v] \tag{2.8}$$

Note that for any $v \in V$ and $t \leq 2^i\ell$, we can write

$$\begin{aligned}
\Pr_{a\sim\mathcal{D}_s, W_a\sim\mathcal{W}_a}[\rho(W_a) \wedge W_a(t) = v] &= \sum_{u\in R_i} \Pr_{a\sim\mathcal{D}_s}[a = u] \Pr_{W_a\sim\mathcal{W}_a}[\rho(W_a) \wedge W_a(t) = v | a = u] \\
&\overset{(1)}{\leq} \sum_{u\in R_i} \sigma_{s,R_i,2^{i+1}\ell}(u) \cdot \sigma_{u,R_i,t}(v) \\
&\overset{(2)}{=} \sigma_{s,R_i,2^{i+1}\ell+t}(v) \\
&\overset{(3)}{\leq} \|\boldsymbol{x}\|_\infty
\end{aligned}$$

The first inequality, (**1**) above follows because we can upperbound the probability that a $2^i\ell$ length walk from a random a is R_i-returning *and* hits v at the t-th step by the probability that

just the t-step "prefix" of the walk from a that ends at v was R_i-returning. The last equality, (**2**), is immediate from the defintion of $\mathcal{D}_s$, and $\boldsymbol{\sigma}_{s,R_i,t}$. For (**3**), note that by Prop. 2.4.9, $\forall t \geq 1$, $\|\boldsymbol{\sigma}_{\boldsymbol{x},R_i,t}\|_\infty \leq \|\boldsymbol{x}\|_\infty$. Further, using Claim 2.3.7, this is at most $1/(n^{\delta(i-1)}\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1)$. Together with (2.8), this gives

$$\begin{aligned}\Pr_{a\sim\mathcal{D}_s, W_a\sim\mathcal{W}_a}[\rho(W_a) \wedge W_a \cap F \neq \emptyset] &\leq \sum_{t\leq 2^i\ell}\sum_{v\in F}\|\boldsymbol{x}\|_\infty \\ &\leq 2^i\ell|F|/(n^{\delta(i-1)}\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1)\end{aligned}$$

Now for the second part. By the union bound, the probability is bounded above by

$$\sum_{t\geq 2^{i-1}\ell}\sum_{u\in F}\Pr_{W_a\sim\mathcal{W}_a}[\rho(W_a) \wedge W_a(t) = u] \leq \sum_{t\geq 2^{i-1}\ell}\sum_{u\in F}\|\boldsymbol{\sigma}_{a,R_i,t}\|_\infty \tag{2.9}$$

By Prop. 2.4.9, the infinity norm is bounded above by $\|\boldsymbol{\sigma}_{a,R_i,2^{i-1}\ell}\|_\infty = \|\boldsymbol{q}^{(i-1)}_{[R_i],a}\|_\infty$. By Claim 2.3.7, the latter is at most $1/n^{\delta(i-2)}$. Plugging in (2.9), we get an upper bound of $2^{i-1}\ell|F|/n^{\delta(i-2)}$. □

Claim 2.4.12. *For any $a \in R_i$, we have the following.*

$$\begin{aligned}&\Pr\big[\rho(W_a) \wedge \rho(W_b) \wedge W_a(2^i\ell) = W_b(2^i\ell) \wedge \\ &\qquad \exists t_a, t_b, \min(t_a, t_b) \leq \tau, W_a(t_a) = W_b(t_b))\big] \qquad \leq \frac{2^{2i}\ell^2}{(n^{\delta(2i-2)}\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1)}\end{aligned}$$

Here, the probability is taken over the following joint distribution: $b \sim \mathcal{D}_s, W_a \sim \mathcal{W}_a, W_b \sim \mathcal{W}_b$

Proof. Let us write out the main event in English. We fix an arbitrary a, and pick $b \sim \mathcal{D}_s$. We perform walks of length $2^i\ell$ from both a and b. We are bounding the probability that these walks are R_i-returning, and that the "initial half" (less than $2^{i-1}\ell$ steps) of one of the walks intersects with the other. Subsequently, both walks end at the same vertex(and both of these walks happen to be R_i-returning).

To that end, let us define two vertices w_1, w_2. We want to bound the probability of that both walks first encounter w_1, and then end at w_2. It will be very useful to treat the latter part simply as two walks from w_1, where one of them is at least of length $2^{i-1}\ell$. Note that w_1 might not be in R_i.

Let $Z_{a,t}$ be the random variable denoting the t-th vertex of a random walk from a. Let us also define R_i-returning walks with an offset g, starting from w_1. Basically, such a walk starts from w (that may not be in R_i) and performs g steps to end up in R_i. Subsequently, it behaves as an R_i-returning walk. Observe that the second parts of the walks are R_i-returning

walks from w_1, with offsets of $\ell - [t_a (\mathrm{mod}\ \ell)]$, $\ell - [t_b (\mathrm{mod}\ \ell)]$. Let $Y_{w,t}$ be the random variable denoting the t-th vertex of an R_i-returning walk from w, with the offset $\ell - [t (\mathrm{mod}\ \ell)]$. We use primed notation (e.g, something like $Y'_{w,t}$) for an independent copy of such variables.

Let us fix values for t_a, t_b such that $\min(t_a, t_b) \leq \tau = 2^{i-1}\ell$. (We will eventually union bound over all such values.) The probability we wish to bound is the following. We use independence of the walks to split the probabilities. There are four independent walks under consideration: one from a, one from b, and two from w.

$$\sum_{w_1 \in V} \sum_{w_2 \in V} \Pr_{\substack{b \sim \mathcal{D}_s \\ \mathcal{W}_a, \mathcal{W}_b, \mathcal{W}_{w_1}}} [Z_{a,t_a} = w_1 \wedge Z_{b,t_b} = w_1 \wedge Y_{w_1, 2^i\ell - t_a} = w_2 \wedge Y'_{w_1, 2^i\ell - t_b} = w_2]$$

$$= \sum_{w_1 \in V} \sum_{w_2 \in V} \Pr_{\mathcal{W}_a}[Z_{a,t_a} = w_1] \Pr_{\substack{b \sim \mathcal{D}_s \\ \mathcal{W}_b}} [Z_{b,t_b} = w_1] \Pr_{\mathcal{W}_{w_1}}[Y_{w_1, 2^i\ell - t_a} = w_2] \Pr_{\mathcal{W}_{w_1}}[Y_{w_1, 2^i\ell - t_b} = w_2] \quad (2.10)$$

Consider $\Pr_{b \sim \mathcal{D}_s, \mathcal{W}_b}[Z_{b,t_b} = w_1]$. This is exactly the w_1th entry in $\boldsymbol{\sigma}_{\boldsymbol{x}, R_i, t_b}$ where $\boldsymbol{x}$ is the distribution given by $\mathcal{D}_s$. By Prop. 2.4.9, this is at most $\|\boldsymbol{x}\|_\infty$, which is at most $1/(n^{\delta(i-1)} \|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1)$ (as argued in the second pat of the proof of Claim 2.4.11).

Since $\min(t_a, t_b) \leq \tau$, at least one of $2^i\ell - t_a$ or $2^i\ell - t_b$ is at least $2^{i-1}\ell$. Thus, one of $\Pr_{\mathcal{W}_{w_1}}[Y_{w_1, 2^i\ell - t_a} = w_2]$ or $\Pr_{\mathcal{W}_{w_1}}[Y_{w_1, 2^i\ell - t_b} = w_2]$ refers to a walk of length at least $2^{i-1}\ell$. Let us bound $\Pr_{\mathcal{W}_{w_1}}[Y_{w_1,t} = w_2]$ for $t \geq 2^i\ell$. We can break such a walk into two parts: the first $\ell - [t (\mathrm{mod}\ \ell)]$ steps lead to some $v \in R_i$, and the second part is an R_i-returning walk of length at least $2^i\ell$ from v to w. Recall that $p_{x,d}(y)$ is the standard random walk probability of starting from x and ending at y after d steps. For some $t' \geq 2^i\ell$,

$$\begin{aligned}
\Pr_{\mathcal{W}_{w_1}}[Y_{w_1,t} = w_2] &= \sum_{v \in R_i} p_{w_1, \ell - [t (\mathrm{mod}\ \ell)]}(v) \sigma_{v, R_i, t'}(w_2) \\
&\leq \sum_{v \in R_i} p_{w_1, \ell - [t (\mathrm{mod}\ \ell)]}(v) \|\boldsymbol{\sigma}_{v, R_i, t'}\|_\infty \\
&\leq \sum_{v \in R_i} p_{w_1, \ell - [t (\mathrm{mod}\ \ell)]}(v) \|\boldsymbol{q}^{(i)}_{[R_i],v}\|_\infty \\
&\leq \sum_{v \in R_i} p_{w_1, \ell - [t (\mathrm{mod}\ \ell)]}(v) n^{-\delta(i-1)} \\
&= n^{-\delta(i-1)}.
\end{aligned}$$

Plugging these bounds in (2.10), for fixed t_a, t_b, there exists $t \in \{2^i\ell - t_a, 2^i\ell - t_b\}$ such that the probability of the main event is at most

$$\begin{aligned}
&\frac{1}{n^{\delta(i-1)} \|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1} \cdot \frac{1}{n^{\delta(i-1)}} \sum_{w_1 \in V} \sum_{w_2 \in V} \Pr_{\mathcal{W}_a}[Z_{a,t_a} = w_1] \Pr_{\mathcal{W}_{w_1}}[Y_{w_1,t} = w_2] \\
&\leq \frac{1}{n^{\delta(2i-2)} \|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1} \sum_{w_1 \in V} \Pr_{\mathcal{W}_a}[Z_{a,t_a} = w_1] \sum_{w_2 \in V} \Pr_{\mathcal{W}_{w_1}}[Y_{w_1,t} = w_2] \\
&= \frac{1}{n^{\delta(2i-2)} \|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1}
\end{aligned}$$

A union bound over all pairs of t_a, t_b completes the proof. □

We now bound the total probability of bad events. Most of the technical work is already done in the previous lemmas; we only need to perform some union bounds.

Lemma 2.4.13. *Total probability of bad events (in a call to* `IdealFindBiClique`(s)*) is at most*

$$\frac{2^{2i+4} r^4 n^{30\delta}}{n^{\delta i/2} \|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1} \tag{2.11}$$

Proof. We bound the bad events by type. Recall that $\ell = n^{5\delta}$.

Type 1: $\exists a, b, c \in A \cup B$, $c \neq a, b$, such that $\boldsymbol{W}_c \cap P_{a,b} \neq \emptyset$.

Fix a choice of $a \in A, b \in B$. Any $c \neq a, b$ is drawn from $\mathcal{D}_s$. In Claim 2.4.11, set $F = P_{a,b}$. By the first part of Claim 2.4.11, the probability that a single walk drawn from $\mathcal{W}_c$ is R_i-returning and intersects $P_{a,b}$ is at most $2^i \ell (2^{i+1}\ell)/n^{\delta(i-1)} \|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1$. The set $\boldsymbol{W}_c$ consists of at most $r n^{\delta(i+18)/2}$ such walks. We union bound over all these walks, and all r^2 choices of a, b, and plug in $\ell = n^{5\delta}$ to get an upper bound of

$$\frac{2^{2i+1} \ell^2 r^3 n^{\delta(i+18)/2}}{n^{\delta(i-1)} \|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1} = \frac{2^{2i+1} r^3 n^{20\delta}}{n^{\delta i/2} \|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1}$$

Type 2: $\exists a, b, b'$ (all distinct) such that $\exists W \in \boldsymbol{W}^b_a$ where $W(\geq \tau) \cap P_{a,b'} \neq \emptyset$. (Or, $\exists a, a' \in A, b \in B$ with analogous conditions.)

Fix a, b, b'. Set $F = P_{a,b'}$ in Claim 2.4.11. By the second part of Claim 2.4.11, the probability that a single walk from $\mathcal{W}_a$ is R_i-returning and intersects F at step $\geq \tau$ is at most $2^i \ell (2^{i+1}\ell)/n^{\delta(i-2)}$. We union bound over all the $r n^{\delta(i+18)/2}$ walks in $\boldsymbol{W}_a$ and all r^3 choices of a, b, b'. (We also union bound over choosing b, b' or a, a'.) The upper bound is $2^{2i+1} r^3 n^{21\delta}/n^{\delta i/2}$.

Type 3: $\exists a, b, W_a \in \boldsymbol{W}^b_a, W_b \in \boldsymbol{W}^a_b$ such that W_a, W_b end at the same vertex and $\exists t_1, t_2$ such that $\min(t_1, t_2) \leq \tau$ and $W_a(t_1) = W_b(t_2)$.

This case is qualitatively different. We will take a union bound over *pairs* of walks, and require the stronger bound of Claim 2.4.12.

Fix $a \in A$. Observe that $b \sim \mathcal{D}_s$. For a single walk $W_a \sim \mathcal{W}_a$ and a single walk $W_b \sim \mathcal{W}_b$, the probability of a Type 3 bad event is bounded by Claim 2.4.12. The upper bound is $2^{2i} \ell^2 / (n^{\delta(2i-2)} \|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1)$. We union bound over the $r^2 n^{\delta(i+18)}$ *pairs* of walks from a and b, and then over the r^2 choices of a, b. The final bound is:

$$\frac{2^{2i} r^4 \ell^2 \; n^{\delta(i+18)}}{n^{\delta(2i-2)} \|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1} = \frac{2^{2i} r^4 n^{30\delta}}{n^{\delta i} \|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1}$$

We complete the proof by taking a union bound over the three types. Note that $\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1 \leq 1$, so we can upper bound the probability of each type of bad event by $\frac{2^{2i+1} r^4 n^{30\delta}}{n^{\delta i/2} \|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1}$. □

2.4.5 Proof of Theorem 2.4.1

Proof. Fix $s \in S_i$. We consider a run of the `FindBiclique`(s) procedure, and not the procedure `IdealFindBiClique`(s). Nonetheless, we can argue that the former efficiently simulates the latter. Let $\mathcal{C}$ be the event that the multisets A and B generated in `FindBiclique`(s) come from $2r$ independent draws from $\mathcal{D}_{s,i}$. In this case, `FindBiclique`(s) behaves exactly like `Ideal-FindBiClique`(s).

Let $\mathcal{E}$ be the event $\bigcap_{a\in A, b\in B} P_{a,b} \neq \emptyset$, and let $\mathcal{F}$ be the union of bad events. By Claim 2.4.6, the probability that `FindBiclique`(s) find a minor is at least $\Pr[\mathcal{E} \cap \overline{\mathcal{F}}]$. We lower bound as follows: $\Pr[\mathcal{E} \cap \overline{\mathcal{F}}] \geq \Pr[\mathcal{C}] \Pr[\mathcal{E} \cap \overline{\mathcal{F}}|\mathcal{C}] \geq \Pr[\mathcal{C}]\ (\Pr[\mathcal{E}|\mathcal{C}] - \Pr[\mathcal{F}|\mathcal{C}])$.

The probability of a single random walk from s of length $2^i\ell$ being R_i-returning is $\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1$. Thus, $\Pr[\mathcal{C}] = \|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1^{2r}$. By Claim 2.3.8, $\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1 \geq n^{-\delta}$, so $\Pr[\mathcal{C}] \geq n^{-2\delta r}$.

Lemma 2.4.3 provides a lower bound for $\Pr[\mathcal{E}|\mathcal{C}]$, and Lemma 2.4.13 provides an upper bound for $\Pr[\mathcal{F}|\mathcal{C}]$. We plug these bounds in below.

$$\Pr[\mathcal{E}|\mathcal{C}] - \Pr[\mathcal{F}|\mathcal{C}] \quad \geq \quad \frac{1}{(4n^{2\delta})^{r^2}} - \frac{2^{2i+4} r^4 n^{30\delta}}{n^{\delta i/2} \|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1} \tag{2.12}$$

Observe how the positive term is independent of i, while the negative term decays exponentially in i. This is crucial to argue that for a sufficiently large (constant) i, the lower bound is non-trivial.

When $i \geq 5r^2$, $n^{i\delta/2} \geq n^{2\delta r^4 + \delta r^4/2} \geq n^{2\delta r^2 + 40\delta}$ (note that, r, the number of vertices in H, is at least 3). By Claim 2.3.8, $\|\boldsymbol{q}^{(i+1)}_{[R_i],s}\|_1 \geq n^{-\delta}$. Thus, for sufficiently large n, $\Pr[\mathcal{F}|\mathcal{C}] \leq 1/(2(4n^{2\delta})^{r^2})$. Putting it all together, the probability of finding a $K_{r,r}$-minor is at least $n^{-4\delta r^2}$. □

2.5 Local partitioning in the trapped case

Theorem 2.4.1 tells us that if there are $\Omega(n^{1-\delta})$ vertices in strata numbered $5r^2$ and above, then `FindMinor` finds a biclique minor with high probability. We deal with the case when most vertices lie in low strata, i.e, random walks from most vertices are trapped in a very small subset.

We will argue that (almost) all vertices in low strata can be partitioned into "pieces", $P_1, P_2, \ldots P_b$ such that each piece is a low conductance cut that is "easily discoverable". We mean that a superset of each piece $P_i, i \in [b]$ can be found by performing short random walks in G. If `FindMinor` fails to find a minor, this lemma can be iteratively applied to make G H-minor-free by removing few edges (this argument is given in §2.6).

We use $p_{s,t}(v)$ to denote the probability that a t length random walk from s ends at v.

Lemma 2.5.1. *Let $i \leq 5r^2$ and $\delta < 1/20r^2$. Let $\alpha \geq n^{-\delta/2}$. Consider some subset $S \subseteq V$ and $i \in \mathbb{N}$ such that $\forall s \in S, \|\boldsymbol{q}^{(i)}_{[S],s}\|_2^2 \leq 1/n^{\delta(i-1)}$. Define $S' \subseteq S$ to be $\{s \in S \mid \|\boldsymbol{q}^{(i+1)}_{[S],s}\|_2^2 \geq 1/n^{\delta i}\}$.*

Suppose $|S'| \geq \alpha n$. Then, there is a subset $\widetilde{S} \subseteq S'$, $|\widetilde{S}| \geq \alpha n/8$ such that for $\forall s \in \widetilde{S}$: there exists a subset $P_s \subseteq S$ where

- *(Low conductance cut) $|E(P_s, S \setminus P_s)| \leq 2n^{-\delta/4} d|P_s|$*
- *(Easily discoverable) $\forall v \in P_s$, $\exists t \leq 160n^{\delta(i+7)}/\alpha$ s.t. $p_{s,t}(v) \geq \alpha/n^{\delta(2i+14)}$.*

2.5.1 Proof overview

The rich literature on local partitioning gives tools to prove variants of the following statement. If there is a vertex s such that short random walks from s do not "spread much", then there exists a low conductance cut "around s" that can be discovered by performing random walks from s. The notion of spreading is measured by the l_2-norm of the random walk distribution.

In Lemma 2.5.1, if we set S to be V, then the lemma is exactly local partitioning. Our aim is to iteratively apply this lemma to partition the entire graph. Suppose, starting with the entire graph, we found low conductances pieces $P_1, P_2, \ldots$. Our aim is to find the next low conductance piece in $S = V \setminus \bigcup_j P_j$. If we could perform random walk restricted to S, then one can simply apply existing local partitioning results. Therein lies the main challenge. For our sublinear application, we need to discover the pieces by doing random walks in the *original graph*. (Let us take a detour to understand why. We are analyzing the scenario where most vertices lie in low strata, which are defined by random walk norms in the original graph. We wish to argue that the graph can be partitioned into small pieces of low conductance, only assuming strata conditions.)

Why is this hard? Note that we need to successfully partition at least $(1-\varepsilon)n$ vertices. Thus, we will have situations where $|S| = \Theta(\varepsilon n)$. In this case, random walks from vertices in S will leave S with high probability. The strata conditions only gives us norm guaratees on such walks (that leave S); yet, we need to locally partition completely within S. There is no clear correspondence between random walks in the original graph and random walks restricted to S.

Our main tool to address this conundrum is the *projected Markov Chain* M_S. Consider the Markov Chain with vertex set S, that contains transitions for all walks in G that start and end at S, but have all internal vertices outside S. Thus, M_S captures all walks in G, but focuses only the vertices in S. Our aim is to perform local partitioning on M_S.

Now for the main technical problem. Lemma 2.5.1 requires us to partition around a vertex s with large $\|\boldsymbol{q}^{(i+1)}_{[S],s}\|_2$ value. We need to get a lower bound for a walk norm, where walks are done in M_S. For any reasonable application of local partitioning in M_S, we need some guarantee on walks (in M_S) of length at least, say, n^δ (or something superconstant). Note

that $\boldsymbol{q}_{[S],s}^{(i+1)}$ is looking at walks of length $2^{i+1}\ell$ that return 2^{i+1} times to S. These walks clearly correspond to walks in M_S. But since i is set to a constant, the lower bound on $\|\boldsymbol{q}_{[S],s}^{(i+1)}\|_2$ only seems to give guarantees on constant length walks in M_S.

We need to argue that, at least on average, walks of length $2^{i+1}\ell$ correspond to sufficiently long walks in M_S. This is precisely what happens in the proof of Lemma 2.5.5. This lemma asserts that for most vertices in S' (as defined in Lemma 2.5.1), random walks in M_S of length n^δ induce a fairly large norm (of the corresponding probability vector). An important tool for proving this lemma is a classic result in random walks, called *Kac's lemma*, that bounds average return times to sets of vertices.

In §2.5.2, we give basic properties of the projected Markov Chain which we build up on to prove Lemma 2.5.5. With Lemma 2.5.5 in hand, we can bring in the machinery of local partitioning to find a low conductance cut. We specifically use the Lovász-Simonovits curve technique to perform local partitioning on M_S [LS90a]. Our analysis closely follows the notation and methods used by Spielman-Teng [ST12].

2.5.2 Projected Markov chain

We define the "projection" of the random walk onto the set S. This uses a construction of [KPS13]. We define a Markov chain M_S over the set S. We retain all transitions from the original random walk on G that are within S, and we denote these by $e_{u,v}^{(1)}$ for every u to v transition in the random walk on G. Additionally, for every $u, v \in S$ and $t \geq 2$, we add a transition $e_{u,v}^{(t)}$. The probability of this transition is equal to the total probability of t-length walks in G from u to v, where all internal vertices in the walk lie outside S.

Since G is irreducible and the stationary mass on S is non-zero, all walks eventually reach S. Thus the outgoing transition probabilities from each v in M_S sum to 1, and hence M_S is a valid Markov chain. Furthermore, by the symmetry of the original random walk, $e_{u,v}^{(t)} = e_{v,u}^{(t)}$. Therefore the transition matrix of M_S remains symmetric, and the stationary distribution is uniform on S.

For a transition $e_{u,v}^{(t)}$ in M_S, we define the length of this transition to be t. For clarity, we use "hops" to denote the length of a walk in M_S, and retain "length" for walks in G. The length of an h hop random walk in M_S is defined to be the sum of the lengths of the transitions it takes. We note that these ideas come from the work of Kale-Peres-Seshadhri to analyze random walks in noisy expanders [KPS13].

We use $\boldsymbol{\tau}_{s,h}$ to denote the distribution of the h-hop walk from s, and $\tau_{s,h}(v)$ to denote the corresponding probability of reaching v. We use $\mathcal{W}_h$ to denote the distribution of h-hop walks starting from the uniform distribution in S.

We state Kac's formula (Corollary 24 in Chapter 2 of [AF02], restated).

Lemma 2.5.2. *(Kac's formula) The expected return time (in G) to S of a random walk starting from S is reciprocal of the fractional stationary mass of S, i.e. $n/|S|$.*

The following is a direct corollary.

Lemma 2.5.3. $\mathbf{E}_{W\sim\mathcal{W}_h}[\textit{length of } W] = hn/|S|$

Proof. Since the walk starts at the stationary distribution, it remains in this distribution at all hops. By linearity of expectation, it suffices to get the expected length for the first hop (and multiply with h). This is precisely expected return time to S, if we performed random walks in G. By Kac's formula above, the expected return time to S equals the reciprocal of the stationary mass of S, which is just $n/|S|$. □

Random walks in M_S do not spread

We begin with an important warmup. Using norm bounds in the premise of Lemma 2.5.1, we show that for every vertex $s \in S'$, there is a large set of destination vertices that are all reached with high probability through random walks of length $2^{i+1}\ell$.

Claim 2.5.4. *For every $s \in S'$, there exists a set $U_s \subseteq S$, $|U_s| \geq n^{\delta(i-2)}/2$, such that $\forall u \in U_s$, $p_{s,2^{i+1}\ell}(u) \geq 1/2n^{\delta i}$.*

Proof. By Prop. 2.3.2, for any $u \in S$, $q^{(i+1)}_{[S],s}(u) = \boldsymbol{q}^{(i)}_{[S],s} \cdot \boldsymbol{q}^{(i)}_{[S],u}$. By the property of S and Cauchy-Schwartz, $q^{(i+1)}_{[S],s}(u) \leq 1/n^{\delta(i-1)}$.

Since $s \in S'$, $\sum_{u\in S} q^{(i+1)}_{[S],s}(u)^2 \geq 1/n^{\delta i}$. Let us simply define U_s to be $\{u | u \in S, q^{(i+1)}_{[S],s}(u) \geq 1/(2n^{\delta i}\})$. Note that $p_{s,2^{i+1}\ell}(u) \geq q^{(i+1)}_{[S],s}(u)$.

$$\begin{aligned}
1/n^{\delta i} \leq \sum_{u\in S} q^{(i+1)}_{[S],s}(u)^2 &= \sum_{u\in U_s} q^{(i+1)}_{[S],s}(u)^2 + \sum_{u\notin U_s} q^{(i+1)}_{[S],s}(u)^2 \\
&\leq |U_s|/n^{2\delta(i-1)} + (1/2n^{\delta i}) \sum_{u\notin U_s} q^{(i+1)}_{[S],s}(u) \\
&\leq |U_s|/n^{2\delta(i-1)} + 1/2n^{\delta i}
\end{aligned}$$

We rearrange to bound the size of U_s. □

The next lemma is an analogue of the previous (Claim 2.5.4) for $\boldsymbol{\tau}$ vectors – (that is for M_S). Recall that $\ell = n^{5\delta}$.

Lemma 2.5.5. *There exists a subset $S'' \subseteq S'$, $|S''| \geq |S'|/2$, such that $\forall s \in S''$, $\|\boldsymbol{\tau}_{s,n^\delta}\|_\infty \geq 1/n^{\delta(i+6)}$.*

Proof. Define event $\mathcal{E}_{s,v,h}$ as follows. The event $\mathcal{E}_{s,v,h}$ occurs when an h-hop random walk from s has length $2^{i+1}\ell$ and ends at v. Observe that $p_{s,2^{i+1}\ell}(v) = \sum_{h\leq 2^{i+1}\ell} \Pr[\mathcal{E}_{s,v,h}]$ (because the number of hops is always at most the length). Since $\boldsymbol{\tau}_{s,h}$ is a random walk vector in a symmetric

Markov Chain, the infinity norm is non-increasing in h. Thus, it suffices to find a subset $S'' \subseteq S'$, $|S''| \geq |S'|/2$ such that $\forall s \in S''$, $\exists v \in S, h \geq n^\delta$, $\Pr[\mathcal{E}_{s,v,h}] \geq 1/n^{\delta(i+6)}$.

We define U_s as given in Claim 2.5.4. For all $v \in U_s$, by Claim 2.5.4, $p_{s,2^{i+1}\ell}(v) \geq 1/2n^{\delta i}$. Therefore, for all $v \in U_s$,

$$\sum_{h \leq 2^{i+1}\ell} \Pr[\mathcal{E}_{s,v,h}] \geq 1/2n^{\delta i} \tag{2.13}$$

We will construct S'' by finding s where for some $v \in U_s$, $\sum_{h \leq n^\delta} \Pr[\mathcal{E}_{s,v,h}]$ is sufficiently small.

For any h,

$$\frac{1}{|S|} \sum_{s \in S'} \sum_{v \in U_s} \Pr[\mathcal{E}_{s,v,h}](2^{i+1}\ell) \leq \mathbf{E}_{W \sim \mathcal{W}_h}[\text{length of } W] = hn/|S|$$

Suppose $h \leq 2^{i+1}\ell/n^{4\delta}$. (This is true for all $h \leq n^\delta$). Then $\sum_{s \in S'} \sum_{v \in U_s} \Pr[\mathcal{E}_{s,v,h}] \leq n^{1-4\delta}$, and $\sum_{h \leq n^\delta} \sum_{s \in S'} \sum_{v \in U_s} \Pr[\mathcal{E}_{s,v,h}] \leq n^{1-3\delta}$.

We rearrange to get

$$\sum_{s \in S'} \sum_{v \in U_s} \sum_{h \leq n^\delta} \Pr[\mathcal{E}_{s,v,h}] \leq n^{1-3\delta}$$

By the Markov bound, there is a set $S'' \subseteq S'$, $|S''| \geq |S'|/2$ such that for all $s \in S''$, $\sum_{v \in U_s} \sum_{h \leq n^\delta} \Pr[\mathcal{E}_{s,v,h}] \leq 2n^{1-3\delta}/|S'|$. By averaging, $\forall s \in S''$, $\exists v \in U_s$, such that $\sum_{h \leq n^\delta} \Pr[\mathcal{E}_{s,v,h}] \leq 2n^{1-3\delta}/(|S'| \cdot |U_s|)$. By the assumptions of Lemma 2.5.1, $|S'| \geq \alpha n \geq n^{1-\delta/2}$. Claim 2.5.4 bounds $|U_s| \geq n^{\delta(i-2)}/2$. Plugging these in,

$$\sum_{h \leq n^\delta} \Pr[\mathcal{E}_{s,v,h}] \leq \frac{2n^{1-3\delta}}{n^{1-\delta/2} n^{\delta(i-2)}/2} \leq \frac{4}{n^{\delta(i+1/2)}}$$

Subtracting this bound from (2.13), $\sum_{h \in [n^\delta, 2^{i+1}\ell]} \Pr[\mathcal{E}_{s,v,h}] \geq 1/4n^{\delta i}$. By averaging, for some $h \in [n^\delta, 2^{i+1}\ell]$, $\Pr[\mathcal{E}_{s,v,h}] \geq 1/(2^{i+3} n^{\delta i} \ell) \geq 1/n^{\delta(i+6)}$. This completes the proof. □

Local Partitioning on M_S: Obtaining the low conductance cut

We perform local partitioning on M_S, starting with arbitrary $s \in S''$. We apply the Lovász-Simonovits curve technique. (The definitions are originally from [LS90a]. Refer to Lecture 7 of Spielman's notes [Spia] as well as Section 2 in Spielman-Teng [ST12].) Before getting into the series of definitions, we give an overview of this technique.

The main idea is to represent the distribution at time t through a one-dimensional function h_t. This function is best thought of as a 2D-plot. Define $h_t(k)$ to be the sum of the k largest probabilities at time t. Note that $h_t(|S|)$ is simply 1 (the sum of all probabilities). Alternately, we sort the probabilities at time t in decreasing order, and consider all prefix sums. These prefix sums are "plotted" by the function h_t. We can linearly interpolate between these values to get a piecewise linear curve, which is the function h_t.

Crucially, this curve is a *concave* function, since we take prefix sums of a non-increasing list. As t increases, the curve h_t flattens out into a straight line from $(0,0)$ to $(|S|,1)$ (which represents the stationary distribution). The rate of flattening is the rate of convergence. The remarkable insight of Lovász-Simonovits is a relation between the flattening rate, the curve structure, and the conductance. They showed that, to upper bound $h_{t+1}(x)$, one can draw a chord between $(x', h_t(x'))$ and $(x'', h_t(x''))$. Here $x' < x < x''$, and these points x', x'' depend on the conductances.

Now for the technical setup.

- Ordering of states at time t: At time t, let us order the vertices in M_S as $v_1^{(t)}, v_2^{(t)}, \ldots$ such that $\tau_{s,t}(v_1^{(t)}) \geq \tau_{s,t}(v_2^{(t)}) \ldots$, breaking ties by vertex id.

- The LS curve h_t: We define a function $h_t : [0, |S|] \to [0,1]$ as follows. For every $k \in [|S|]$, set $h_t(k) = \sum_{j \leq k} [\tau_{s,t}(v_j^{(t)}) - 1/|S|]$. (Set $h_t(0) = 0$.) For every $x \in (k, k+1)$, we linearly interpolate to construct $h_t(x)$. Alternately, $h_t(x) = \max_{\vec{w} \in [0,1]^{|S|}, \|\vec{w}\|_1 = x} \sum_{v \in S} [\tau_{s,t}(v) - 1/|S|] w_i$.

- Level sets: For $k \in [0, |S|]$, we define the (k,t)-level set, $L_{k,t}$ to be

$$\{v_1^{(t)}, v_2^{(t)}, \ldots, v_k^{(t)}\}.$$

  The *minimum probability* of $L_{k,t}$ denotes $\tau_{s,t}(v_k^{(t)})$.

- Conductance: for some $T \subseteq S$ we define the conductance of T in M_S to be

$$\Phi(T) = \frac{\sum_{\substack{u \in T \\ v \in S \setminus T}} \tau_{u,1}(v)}{\min(|T|, |S \setminus T|)}$$

The main lemma of Lovász-Simonovits is the following (Lemma 1.4 of [LS90a]).

Lemma 2.5.6. *For all k and all t,*

$$h_t(k) \leq \frac{1}{2}[h_{t-1}(k - 2\min(k, |S| - k)\Phi(L_{k,t})) + h_{t-1}(k + 2\min(k, |S| - k)\Phi(L_{k,t}))]$$

The typical use of the Lovász-Simonovits technique is to argue about rapid mixing when all conductances (or conductances of sufficiently large sets) are lower bounded. We consider a scenario in which only sets with minimum probability at least (say) p have high conductance. In this case, we can guarantee that the largest probability will converge to p.

Lemma 2.5.7. *Suppose there exists $\phi \in [0,1]$ and $p > 2/|S|$ such that for all $t' \leq t$ and all $k \in [n]$ the following implication is true: if $L_{k,t'}$ has a minimum probability of at least p, then $\Phi(L_{k,t}) \geq \phi$. Then for all $k \in [0, |S|]$, $h_t(k) \leq \sqrt{k}(1 - \phi^2/2)^t + pk$.*

Proof. We will prove by induction over t. If $k \geq 1/p$, then the RHS is at least 1. Thus the bound is trivially true. Let us assume that $k < 1/p < |S|/2$. We split into two cases based on the conductance of $L_{k,t}$.

First let us consider the case where $\Phi(L_{k,t}) \geq \phi$. By Lemma 2.5.6, $h_t(k) \leq (1/2)[h_{t-1}(k - 2\min(k, |S|-k)\Phi(L_{k,t})) + h_{t-1}(k + 2\min(k, |S|-k)\Phi(L_{k,t}))]$. Since $k < |S|/2$, $\min(k, |S|-k) = k$. By concavity of h_k, we can replace $\Phi(L_{k,t})$ in the above inequality by any lower bound. (We can always upper bound by a chord "above" the given chord.) Thus,

$$\begin{aligned} h_t(k) &\leq \frac{1}{2}\Big(h_{t-1}\big(k(1-2\phi)\big) + h_{t-1}\big(k(1+2\phi)\big)\Big) \\ &\overset{(1)}{\leq} \frac{1}{2}\Big(\sqrt{k(1-2\phi)}(1-\phi^2/2)^{t-1} + \sqrt{k(1+2\phi)}(1-\phi^2/2)^{t-1} + 2kp\Big) \\ &\overset{(2)}{=} \frac{1}{2}\Big(\sqrt{k}\left(1-\phi^2/2\right)^{t-1}\left(\sqrt{1-2\phi} + \sqrt{1+2\phi}\right) + 2kp\Big) \\ &\overset{(3)}{\leq} \sqrt{k}\left(1-\phi^2/2\right)^{t} + kp. \end{aligned}$$

Here (**1**) follows by the inductive hypothesis. In the end, for (**3**) we use the bound

$$\left(\sqrt{1+z} + \sqrt{1-z}\right)/2 \leq 1 - z^2/8.$$

Now we deal with the case when $\Phi(L_{k,t}) < \phi$. $L_{k,t}$ must have minimum probability less than p by assumption. Let $k' < k$ be the largest index such that $L_{k',t}$ has minimum probability at least p. Note that $\Phi(L_{k',t}) \geq \phi$. Therefore, as proven in the first case, $h_t(k') \leq \sqrt{k'}\left(1-\phi^2/2\right)^t + k'p$. Every vertex in $L_{k,t} \setminus L_{k',t}$ has a probability at most p. By the concavity of $h_t(x)$,

$$\begin{aligned} h_t(k) &\leq h_t(k') + (k-k')p && (2.14) \\ &\leq \sqrt{k'}\left(1-\phi^2/2\right)^t + k'p + (k-k')p && (2.15) \\ &\leq \sqrt{k'}\left(1-\phi^2/2\right)^t + kp && (2.16) \end{aligned}$$

□

The following lemma is a direct corollary.

Lemma 2.5.8. *Consider a vertex s such that $\|\boldsymbol{\tau}_{s,n^\delta}\|_\infty \geq 1/n^{\delta(i+6)}$. Then, there exists a level set for some $t \leq n^\delta$ with minimum probability at least $1/10n^{\delta(i+6)}$ and conductance $< n^{-\delta/4}$.*

Proof. Suppose not. So, for all $t \leq n^\delta$, if a level set has minimum probability at least $1/10n^{\delta(i+6)}$, it has conductance at least $n^{-\delta/4}$. Since $|S| \geq \alpha n \geq n^{1-\delta/2}$, by choosing δ sufficiently small we get that the minimum probability satisifies $1/10n^{\delta(i+6)} \geq 2/|S|$. Thus, we can apply Lemma 2.5.7. For all $k \in [0, |S|]$, $h_{n^\delta}(k) \leq \sqrt{k}(1-n^{-\delta/2}/2)^{n^\delta} + k/10n^{\delta(i+6)}$. Setting $k = 1$, $h_{n^\delta}(1) \leq \exp(-n^{\delta/2}/2) + 1/10n^{\delta(i+6)} < 1/n^{\delta(i+6)}$. Note that $h_{n^\delta}(1)$ is the largest probability in $\boldsymbol{\tau}_{s,n^\delta}$, which by assumption is at least $1/n^{\delta(i+6)}$. Contradiction.

□

Wrapping up by producing $\widetilde{S}$

Proof. (Of Lemma 2.5.1) Define S'' as given in Lemma 2.5.5. By Lemma 2.5.8, for every $s \in S''$, there exists some level set for $t_s \leq n^\delta$ with minimum probability at least $1/10n^{\delta(i+6)}$ and conductance at most $n^{-\delta/4}$. Let us call this level set P_s. Note that $|P_s| \leq 10n^{i(\delta+6)}$ and according to Lemma 2.5.1, we have $|S| \geq |S'| \geq \alpha n \geq n^{1-\delta/2}$. This implies that $|P_s| < |S|/2$ (for sufficiently small δ and $i \leq 5r^2$). By the construction of M_S, we have,

$$\Phi(P_s) \geq \frac{\sum_{\substack{x \in P_s \\ y \in S \setminus P_s}} \tau_{x,1}(y)}{\min(|P_s|, |S \setminus P_s|)} = \frac{E(P_s, S \setminus P_s)}{2d|P_s|}$$

The first inequality follows because we restrict the numerator to length one transitions in the Markov Chain M_S (which correspond to edges in G). Rearranging, we get $E(P_s, S \setminus P_s) \leq n^{-\delta/4}(2d|P_s|)$.

For all $s \in S''$ and $v \in P_s$, $\tau_{s,n^\delta}(v) \geq 1/10n^{\delta(i+6)}$. Set $L = 160n^{\delta(i+7)}/\alpha$. Let $\widetilde{S}$ be the subset of S'' such that $\forall s \in \widetilde{S}$, P_s is such that $\forall v \in P_s$, $\sum_{l \leq L} p_{s,l}(v) \geq 1/20n^{\delta(i+6)}$. By averaging, $\exists l \leq L$ such that $p_{s,l}(v) \geq \alpha/n^{\delta(2i+14)}$.

We have seen that $\widetilde{S}$ satisfies the two desired properties: for all $s \in \widetilde{S}$ $E(P_s, S \setminus P_s) \leq 2n^{-\delta/4}d|P_s|/\alpha$ and for all $v \in P_s$, $\exists t \leq 160n^{\delta(i+7)}$ such that $p_{s,t}(v) \geq \alpha/n^{\delta(2i+14)}$. It only remains to prove an upper bound on $|S'' \setminus \widetilde{S}|$.

Consider any $s \in S'' \setminus \widetilde{S}$. There exists some $v_s \in P_s$ such that $\tau_{s,n^\delta}(v_s) \geq 1/10n^{\delta(i+6)}$ but $\sum_{l \leq L} p_{s,l}(v_s) < 1/20n^{\delta(i+6)}$. Let us use $\hat{p}_{s,l}(v_s)$ to denote the probability of reaching v_s from s in an l-length walk that makes n^δ hops. Observe that

$$\begin{aligned}
\tau_{s,n^\delta}(v_s) &= \sum_{l \geq n^\delta} \hat{p}_{s,l}(v_s) \\
&= \sum_{l=n^\delta}^{L} \hat{p}_{s,l}(v_s) + \sum_{l>L} \hat{p}_{s,l}(v_s) \\
&\leq \sum_{l=n^\delta}^{L} p_{s,l}(v_s) + \sum_{l>L} \hat{p}_{s,l}(v_s) \\
&< 1/20n^{\delta(i+6)} + \sum_{l>L} \hat{p}_{s,l}(v_s)
\end{aligned}$$

The last inequality follows from the fact that $s \in S'' \setminus \widetilde{S}$, and hence $\sum_{l=n^\delta}^{L} p_{s,l}(v_s) < 1/20n^{\delta(i+6)}$. Since $\tau_{s,n^\delta}(v_s) \geq 1/10n^{\delta(i+6)}$, the above calculation shows that $\sum_{l>L} \hat{p}_{s,l}(v_s) > 1/20n^{\delta(i+6)}$. Thus,

$$\frac{1}{|S|} \sum_{s \in S'' \setminus \widetilde{S}} \sum_{l>L} \hat{p}_{s,l}(v_s) L > \frac{|S'' \setminus \widetilde{S}| \cdot L}{|S| 20n^{\delta(i+6)}} = \frac{160\alpha^{-1}n^{\delta(i+7)} \cdot |S'' \setminus \widetilde{S}|}{20|S|n^{\delta(i+6)}} = \frac{8n^\delta |S'' \setminus \widetilde{S}|}{\alpha|S|}$$

By Lemma 2.5.3,

$$\frac{1}{|S|} \sum_{s \in S'' \setminus \widetilde{S}} \sum_{l>L} \hat{p}_{s,l}(v_s) L \leq \mathbf{E}_{W \sim \mathcal{W}_n^\delta}[\text{length of } W] = \frac{n^{1+\delta}}{|S|}$$

Combining the above, $|S'' \setminus \widetilde{S}| \leq \alpha n/8$. By Lemma 2.5.5, $|S''| \geq |S'|/2 \geq \alpha n/2$, yielding the bound $|\widetilde{S}| \geq \alpha n/4$.

□

2.6 Wrapping it all up: the proof of the main theorem

We have all the tools required to complete the proof of Theorem 2.2.1. Our aim is to show that whenever $\texttt{FindMinor}(G, \varepsilon, H)$ outputs an H-minor with probability $< 2/3$, then G is ε-close to being H-minor-free. Henceforth in this section, we will simply assume the "if" condition.

We produce the decomposition procedure used in the proof from §2.5 again below. We set $\alpha = \varepsilon/(50r^2 \log n)$.

$\texttt{Decompose}(G)$

1. Initialize $S = V$ and $\mathcal{P} = \emptyset$.
2. For $i = 1, \ldots, 5r^2$:
   (a) Assign $S' := \left\{ s \in S : ||\boldsymbol{q}^{(i+1)}_{[S],s}||_2^2 \geq 1/n^{\delta i} \right\}$
   (b) While $|S'| \geq \alpha n$:
      i. Let $S'' = \{s \in S' : \ \|\boldsymbol{\tau}_{s,n^\delta}\|_\infty \ \geq 1/n^{\delta(i+6)}\}$ be as in Lemma 2.5.5.
      ii. Choose arbitrary $s \in S''$, and let P_s be as in Lemma 2.5.1.
      iii. Add P_s to $\mathcal{P}$ and assign $S := S \setminus P_s$
      iv. Assign $S' := \left\{ s \in S : ||\boldsymbol{q}^{(i+1)}_{[S],s}||_2^2 \geq 1/n^{\delta i} \right\}$
   (c) Assign $S := S \setminus S'$
   (d) Assign $X_i := S'$
3. Let $X = \bigcup_i X_i$.
4. Output the partition $\mathcal{P}, X, S$

The procedure `Decompose` repeatedly employs Lemma 2.5.1 for values of $i \leq 5r^2$. In the ith iteration, eventually $|S'|$ becomes too small for Lemma 2.5.1. Then, S' is moved (from S) to an "excess" set X_i, and the next iteration begins. `Decompose` ends with a partition $\mathcal{P}, X, S$ where each set in $\mathcal{P}$ is a low conductance cut, X is fairly small, and `FindBiclique` succeeds with high probability on every vertex in S.

This is formalized in the next lemma.

Lemma 2.6.1. *Assume $\varepsilon > \varepsilon_{\mathsf{CUTOFF}}$. Suppose $\texttt{FindMinor}(G, \varepsilon, H)$ outputs an H-minor with probability $< 2/3$. Then, the output of `Decompose` satisfies the following conditions.*

- $|X| \leq \varepsilon n/10$.
- $|S| \leq \varepsilon n/10$.

- $\forall P_s \in \mathcal{P}, v \in P_s$, $\exists t \leq 160n^{6\delta r^2}/\alpha$ *such that* $p_{s,t}(v) \geq \frac{\alpha}{n^{11\delta r^2}}$.
- *There are at most* $\varepsilon n/10$ *edges that go between different* P_s *sets.*

Proof. Consider the X_i's formed by `Decompose`. Each of these has size at most $\alpha n = \varepsilon n/50r^2 \log n$, and there are at most $5r^2$ of these. Clearly, their union has size at most $\varepsilon n/10$.

The third condition holds directly from Lemma 2.5.1. Consider the number of edges that go between P_s and the rest of S, when P_s was constructed (in `Decompose`). By Lemma 2.5.1 again, the number of these edges is at most

$$2n^{-\delta/4}d|P_s|/\alpha = 100r^2(\log n)\varepsilon^{-1}n^{-\delta/4}d|P_s|$$

Note that $\varepsilon > \varepsilon_{\mathsf{CUTOFF}}$. For sufficiently small constant δ, the number of edges between P_s and $S \setminus P_s$ (at the time of removal) is at most $\varepsilon|P_s|/10$. The total number of such edges is at most $\varepsilon n/10$ (since P_s are all disjoint).

Suppose, for contradiction's sake, that $|S| > \varepsilon n/10$. Consider the stratification process with $R_0 = S$. By construction of S, $\forall s \in S$, $||\boldsymbol{q}_{[S],s}^{(5r^2+1)}|| \leq 1/n^{5\delta r^2}$. Thus, all of these vertices will lie in strata numbered $5r^2$ or above. Since $\varepsilon > \varepsilon_{\mathsf{CUTOFF}}$, by Lemma 2.3.9, at most $\varepsilon n/\log n$ vertices are in strata numbered more than $1/\delta+3$. By Theorem 2.4.1, for at least $\varepsilon n/10 - \varepsilon n/\log n \geq \varepsilon n/20$ vertices, the probability that the paths discovered by `FindBiclique`(s) contain a $K_{r,r}$-minor is at least $n^{-4\delta r^2}$. A $K_{r,r}$-minor contains a K_r-minor (simply contract any perfect matching), and thus, it contains an H-minor. The algorithm succeeds in finding an H-minor with probability at least $n^{-4\delta r^2}$.

All in all, this implies that the probability that a single call to `FindBiclique` finds an H-minor is at least $n^{-4\delta r^2}$. Since `FindMinor` makes $n^{35\delta r^2}$ calls to `FindBiclique`, an H-minor is found with probability at least $5/6$. This is a contradiction, and we conclude that $|S| \leq \varepsilon n/10$. □

And now, we can prove the correctness guarantee of `FindMinor`.

Claim 2.6.2. *Suppose* `FindMinor`(G, ε, H) *outputs an* H*-minor with probability* $< 2/3$*. Then* G *is* ε*-close to being* H*-minor-free.*

Proof. If $\varepsilon \leq \varepsilon_{\mathsf{CUTOFF}}$, then `FindMinor` runs an exact procedure. So the claim is clearly true. Henceforth, assume $\varepsilon > \varepsilon_{\mathsf{CUTOFF}}$. Apply Lemma 2.6.1 to partition V into $\mathcal{P}, X, S$.

Call $s \in V$ bad if there is a corresponding $P_s \in \mathcal{P}$ and P_s induces an H-minor. By Lemma 2.6.1, for all $v \in P_s$, $\exists t \leq 160n^{6\delta r^2}/\alpha$ such that $p_{s,t}(v) \geq \alpha/n^{11\delta r^2}$. Note that $160n^{6\delta r^2}/\alpha \leq n^{7\delta r^2}$ and $\alpha/n^{11\delta r^2} \geq n^{-12\delta r^2}$. Also, $|P_s| \leq 160(n^{6\delta r^2}/\alpha) \times (n^{11\delta r^2}/\alpha) \leq n^{18\delta r^2}$. Note that `LocalSearch`$(s)$ performs walks of all lengths up to $n^{7\delta r^2}$, and performs $n^{30\delta r^2}$ walks of each length. For any $v \in P_s$, the probability that `LocalSearch`(s) does not add v to B (the set of "discovered" vertices in `LocalSearch`(s)) is at most $(1 - n^{-12\delta r^2})^{n^{30\delta r^2}} \leq 1/n^2$. Taking a

union bound over P_s, the probability that P_s is not contained in B is at most $1/n$. Consequently, for bad s, `LocalSearch`(s) outputs an H-minor with probability $> 1 - 1/n$.

Suppose there are more than $n^{1-30\delta r^2}$ bad vertices. The probability that a u.a.r. $s \in V$ is bad is at least $n^{-30\delta r^2}$. Since `FindMinor`(G, ε, H) invokes `LocalSearch` $n^{35\delta r^2}$ times, the probability that `LocalSearch`(s) is invoked for a bad vertex is at least $1 - 1/n$. Thus, `FindMinor`(G, ε, H) outputs an H-minor with probability $> 1 - 2/n$, contradicting the claim assumption.

We conclude that there are at most $n^{1-30\delta r^2}$ bad vertices. Each P_s has at most $n^{18\delta r^2}$ vertices, and $|\bigcup_{s \text{ bad}} P_s| \leq n^{1-12\delta r^4} \leq \varepsilon n/10$.

We can make G H-minor-free by deleting all edges incident to X, all edges incident to S, all edges incident to vertices in any bad P_s sets, and all edges between P_s sets. By Lemma 2.6.1 and the bound given above, the total number of edges deleted is at most $4\varepsilon dn/10 < \varepsilon dn$. □

Finally, we bound the running time.

Claim 2.6.3. *The running time of* `FindMinor`(G, ε, H) *is*

$$dn^{1/2+O(\delta r^2)} + d\varepsilon^{-2\exp(2/\delta)/\delta}$$

Proof. If $\varepsilon < \varepsilon_{\mathsf{CUTOFF}}$, then the running time is simply $O(n^2)$. Since $\varepsilon < n^{-\delta/\exp(2/\delta)}$, this can be expressed as $\varepsilon^{-2\exp(2/\delta)/\delta}$.

Assume $\varepsilon \geq \varepsilon_{\mathsf{CUTOFF}}$. The total number of vertices encountered by all the `LocalSearch` calls is $n^{O(\delta r^2)}$. There is an extra d factor to determine all incident edges through vertex queries. Thus, the total running time is $dn^{O(\delta r^2)}$, because of the quadratic overhead of `KKR`. Consider a single iteration for the main loop of `FindBiclique`. First, `FindBiclique` performs $2r$ random walks of length $2^{i+1}n^{5\delta}$, and then for each of these, `FindPath` performs $n^{\delta i/2+9\delta}$ walks of length $2^i n^{5\delta}$. Hence, the total steps (and thus queries) in all walks performed by a single call to `FindBiclique` is

$$\sum_{i=5r^2}^{1/\delta+3} \left(2r2^{i+1}n^{5\delta} + 2rn^{\delta i/2+9\delta}2^i n^{5\delta}\right) = rn^{1/2+O(\delta)}. \tag{2.17}$$

While this is the total number of vertices encountered, we note that the calls made to `KKR`(F, H) are for much smaller graphs. The output of find path has size $O(2^{1/\delta}n^{5\delta})$, and the subgraph F constructed has at most $O(2^{1/\delta}n^{5\delta})$ vertices. We incur an extra d factor to determine the induced subgraph through vertex queries. Thus, the time for each call to `KKR`(F, H) is $n^{O(\delta)}$. There are $n^{O(\delta r^2)}$ calls to `FindBiclique`, and we can bound the total running time by $dn^{1/2+O(\delta r^2)}$. □

Chapter 3

A polynomial-time two sided tester for hyperfinite properties

This chapter is a generalization of the results originally in [KSS19a] which showed that all minor-closed graph properties are testable in time poly(d/ε) in the bounded-degree graph model. Here we generalize from minor-closed families to hyperfinite families with a polynomial hyperfiniteness function.

Definition 3.0.1. *A bounded-degree graph is (α, k)-hyperfinite if there exists a set of at most αdn edges whose removal leaves G with no connected component of size greater than k. For a real-valued function, $f(\alpha)$, a property of bounded degree graphs, $\mathcal{P}$, is* $(f(\alpha))$-hyperfinite *if $G \in \mathcal{P}$, G is $(\alpha, f(\alpha))$-hyperfinite for all α in the unit interval. We call f the hyperfinitess function for a property P if all graphs in P are $(\varepsilon, f(\varepsilon))$-hyperfinite.*

Testing for minor-closed properties was first addressed by [BSS10], where they showed that it can be accomplished in time independent of graph size. However, the bound they derive has a super exponential dependence on the distance parameter, ε. Hyperfiniteness plays a crucial role in their analysis and [NS13] leverages this to extend their results to all hyperfinite families.

Here we present a tester which is able to test a subset of hyperfinite properties in time polynomial in (d/ε). The analysis does not cover all hyperfinite families however. It requires that the hyperfiniteness function be subexponential. It still remains an open question whether all hyperfinite properties can be tested in polynomial time.

3.1 The algorithm

Theorem 3.1.1. *Let $f(\alpha)$ be a real-valued function defined on the unit interval and bounded from above by $2^{(1/8\alpha)^{1/20}}$. Every monotone $(f(\alpha))$-hyperfinite property of graphs is testable with*

$d \cdot \text{poly}(f(\text{poly}(1/\varepsilon)))$ *queries.*

The symbols used in the algorithm are as follows:

- $\ell = 1/\varepsilon^{40}$
- $k = f(1/4\ell^2)$ where $f(\cdot)$ is a function such that $\mathcal{P}$ is $f(\alpha)$-hyperfinite.

`IsHyperfinite`(G, ε)

1. Pick multiset S of $\varepsilon^{-42}k^2$ uniform random vertices.
2. For every $s \in S$, run `EstClip`(s) and `LocalSearch`(S).
3. If any call to `LocalSearch` returns FOUND, REJECT.
4. If more than $\frac{\varepsilon^{-34}k^2}{2}$ calls to `EstClip` return LOW, REJECT.
5. ACCEPT

`LocalSearch`(s)

1. Perform $k^3\ell^3$ independent random walks of length $25600\ell k$ from s. Add all the vertices encountered to set B_s.
2. Determine $G[B_s]$, the subgraph induced by B_s, and explicitly check if $G[B_s] \in \mathcal{P}$.
3. If $G[B_s] \notin \mathcal{P}$, return FOUND.

`EstClip`(s)

1. Perform $w = k^2$ walks of length $8k$ from s.
2. For every vertex v, let $w_v =$ number of walks that end at v.
3. Let $T = \{v \mid w_v \geq k/8\}$.
4. If $\sum_{v \in T} w_v \geq w/3$, output HIGH, else output LOW.

3.2 Random walks on hyperfinite graphs

We first define the clipped norm.

Definition 3.2.1. *Given $x \in (\mathbb{R}^+)^{|V|}$ and parameter $\xi \in [0, 1)$, the ξ-clipped vector* $\text{cl}(\boldsymbol{x}, \xi)$ *is the lexicographically least vector $\boldsymbol{y}$ optimizing the program:* $\min \|\boldsymbol{y}\|_2$*, subject to* $\|\boldsymbol{x} - \boldsymbol{y}\|_1 \leq \xi$ *and* $\forall v \in V, \boldsymbol{y}(v) \leq \boldsymbol{x}(v)$.

The clipping operation removes "outliers" from a vector, with the intention of minimizing the l_2-norm. For a probability distribution $\mathbf{p}_s^\ell$, a small value of $\|\mathbf{p}_s^\ell\|_2^2$ is a measure of the spread of the walk. But this is a crude lens. There may be one large coordinate in $\mathbf{p}_s^\ell$ that determines the norm, while all other coordinates are (say) uniform. The clipped norm better captures the notion of a random walk spreading.

We state the main result of this section. The constant 3/8 below is just for convenience, and can be replaced by any non-zero constant (with a constant drop in the lower bound).

Lemma 3.2.2. *If G is $(1/4\ell^2, k)$-hyperfinite, then for at least $(1-1/\ell)n$ vertices, $\|\mathrm{cl}(\mathbf{p}_s^\ell, 3/8)\| \geq 1/4k$.*

It is convenient to think of the Markov chain on G in terms of a multigraph on G, with $2d$ edges from each vertex. Each edge has probability exactly $1/2d$, and self-loops consist of many such edges. Note that every edge of the original graph is a single edge in this multigraph. For any subset of vertices $C \subseteq V$, let us define the random walk restricted to C. We remove every cut edge (u, v) (where $u \in C$ and $v \notin C$) and add a self-loop of the same probability at u. This produces a Markov chain on C that is symmetric. Given a subset C and $v \in C$, we use $\mathbf{p'}_{v,t}$ to denote the distribution of endpoints of t-length random walk starting from v and restricted to C. (In our use, C will apparent from context, so we will not carry the dependence on C in the notation.)

The following claim relates the clipped norms of the $\mathbf{p}_v^t$ and $\mathbf{p'}_{v,t}$ vectors.

Claim 3.2.3. *Let $C \subset V$ and $v \in C$. Let η be the probability that a t-length random walk from v (in G) leaves C. For any $\sigma > \eta$, $\|\mathrm{cl}(\mathbf{p}_v^t, \sigma - \eta)\|_2^2 \geq \|\mathrm{cl}(\mathbf{p'}_{v,t}, \sigma)\|_2^2$.*

Proof. The random walk restricted to C is obtained by adding some self-loops that are not in the original Markov chain. Color all these self-loops red. Let $\mathbf{r}_{v,t}(u)$ be the probability of a t-length walk from v to u that contains a red edge. Any path without a red edge is a path in G (with the same probability), so $\mathbf{p'}_{v,t}(u) \leq \mathbf{p}_v^t(u) + \mathbf{r}_{v,t}(u)$.

Note that $\sum_{u \in C} \mathbf{r}_{v,t}(u)$ is the total probability of a random walk from u restricted to C encountering a red self-loop. Red self-loops correspond to cut edges in the original graph, and thus, this is the probability of encountering a cut edge. Hence, $\sum_{u \in C} \mathbf{r}_{v,t}(u) \leq \eta$.

Intuitively, we can obtain a σ-clipping of $\mathbf{p'}_{v,t}$ by first clipping at most η probability mass to get $\mathbf{p}_v^t$, and then performing a $(\sigma - \eta)$-clipping of $\mathbf{p}_v^t$. We formalize this below.

Let $\boldsymbol{q} = \mathrm{cl}(\mathbf{p}_v^t, \sigma - \eta)$, and let us define the $|C|$-dimensional vector $\boldsymbol{w}$ by $\boldsymbol{w}(u) = \min\left(\boldsymbol{q}(u), \mathbf{p'}_{v,t}(u)\right)$. Since $\boldsymbol{w}$ is non-negative and $\boldsymbol{w}(u) \leq \boldsymbol{q}(u)$ for all $u \in C$, it follows that $\|\boldsymbol{w}\|_2^2 \leq \|\boldsymbol{q}\|_2^2 = \|\mathrm{cl}(\mathbf{p}_v^t, \sigma - \eta)\|_2^2$. By construction, for all $u \in C$, $\boldsymbol{w}(u) \leq \mathbf{p'}_{v,t}(u)$. We will prove that $\|\boldsymbol{w} - \mathbf{p'}_{v,t}\|_1 \leq \sigma$, implying that $\|\mathrm{cl}(\mathbf{p'}_{v,t}, \sigma)\|_2^2 \leq \|\boldsymbol{w}\|_2^2$. This will complete the argument.

Let $D \subseteq C$ be the set of coordinates such that $\boldsymbol{q}(u) < \mathbf{p'}_{v,t}(u)$. Since $\boldsymbol{w}(u) = \min(\boldsymbol{q}(u), \mathbf{p'}_{v,t}(u))$, $\|\mathbf{p'}_{v,t} - \boldsymbol{w}\|_1 = \sum_{u \in D}[\mathbf{p'}_{v,t}(u) - \boldsymbol{q}(u)]$. Combining with the previous observations and noting that $\boldsymbol{q} = \mathrm{cl}(\mathbf{p}_v^t, \sigma - \eta)$,

$$\begin{aligned}\|\mathbf{p'}_{v,t} - \boldsymbol{w}\|_1 &\leq \sum_{u \in D}[\mathbf{p}_v^t(u) + \mathbf{r}_{v,t}(u) - \boldsymbol{q}(u)] \leq \|\mathbf{p}_v^t(u) - \boldsymbol{q}\|_1 + \sum_{u \in C} \mathbf{r}_{v,t}(u) \\ &\leq (\sigma - \eta) + \eta = \sigma\end{aligned}$$

□

We now prove the main lemma of this section, Lemma 3.2.2.

Proof. By assumption, there exists a subset R of at most $\frac{dn}{4\ell^2}$ edges whose removal breaks up G into connected components of size at most k. Refer to these as hyperfinite components. Now, consider an ℓ-length walk in G starting from the stationary distribution (which is uniform). The probability that this walk encounters an edge in R at any step is exactly $|R|/2dn$. Let the random variable X_v be the number of edges of R encountered in an ℓ-length walk from v. Note that when $X_v = 0$, then the walk remains in the hyperfinite component containing v. Thus,

$$\begin{aligned}&(1/n)\sum_v \Pr[\text{walk from } v \text{ leaves hyperfinite component}]\\ &\leq \sum_{v\in V} \mathbf{E}[X_v]/n = \ell|R|/2dn \leq 1/8\ell\end{aligned}$$

By the Markov bound, for at least $(1 - 1/\ell)n$ vertices, the probability that an ℓ-length walk starting at v encounters an edge of R and thus leaves the hyperfinite piece containing v is at most $1/8$. Denote the set of these vertices by S.

Consider any $s \in S$. Suppose it is contained in the hyperfinite component C. Note that $\|\mathrm{cl}(\mathbf{p'}_{s,\ell}, 1/2)\|_1 \geq 1/2$. Furthermore, $\mathrm{cl}(\mathbf{p'}_{s,\ell}, 1/2)$ has support at most k. By Jensen's inequality, $\|\mathrm{cl}(\mathbf{p'}_{s,\ell}, 1/2)\|_2^2 \geq (4k)^{-1}$. As argued earlier, the probability that a random walk (in G) from s leaves C is at most $1/8$. Applying Claim 3.2.3 for $\sigma = 1/2$ and $\eta = 1/8$, we conclude that $\|\mathrm{cl}(\mathbf{p}_s^\ell, 1/2 - 1/8)\|_2^2 \geq \frac{1}{4k}$. □

3.3 The existence of a discoverable decomposition

If many vertices have large clipped norms, we prove that G can be partitioned into small low conductance cuts. Furthermore, each cut can be discovered by $\mathrm{poly}(\ell)$ ℓ-length random walks. The analysis follows the structure given in [KSS18].

Lemma 3.3.1. *Let $c > 1$ be a parameter. Suppose there exists $S \subseteq V$ such that $|S| > n/\ell^{1/5}$ and $\forall s \in S, \|\mathrm{cl}(\mathbf{p}_s^\ell, 1/4)\|_2^2 > \ell^{-c}$. Then, there exists $\widetilde{S} \subseteq S$ with $|\widetilde{S}| \geq |S|/4$ such that for each $s \in \widetilde{S}$, there exists a subset $P_s \subseteq S$ where*

- $\forall v \in P_s$, $\sum_{t<16\ell^{c+1}} \mathbf{p}_s^t(v) \geq 1/8\ell^{c+1}$.
- $|E(P_s, S \setminus P_s)| \leq 4d|P_s|\sqrt{c\ell^{-1/5}\log\ell}$.

A straightforward application of this lemma leads to the main decomposition theorem.

Theorem 3.3.2. *Suppose there are at least $(1-1/\ell^{1/5})n$ vertices s such that $\|\mathrm{cl}(\mathbf{p}_s^\ell, 1/4)\|_2^2 > \ell^{-c}$. Then, there is a partition $\{P_1, P_2, \ldots, P_b\}$ of the vertices such that:*

- *For each P_i, there exists $s \in V$ such that: $\forall v \in P_i$, $\sum_{t<10\ell^{c+1}} \mathbf{p}_s^t(v) \geq 1/8\ell^{c+1}$.*
- *The total number of edges crossing the partition is at most $8dn\sqrt{c\ell^{-1/5}\log\ell}$.*

Proof. We simply iterate over Lemma 3.3.1. Let $T = \{s \mid \|\mathrm{cl}(\mathbf{p}_s^\ell, 1/4)\|_2^2 \le \ell^{-c}\}$. By assumption, $|T| \le n/\ell^{1/5}$. We will maintain a partition of the vertices $\{T, Q_1, Q_2, \ldots, Q_a, S\}$ with the following properties. (1) Each Q_i satisfies the first condition of the theorem. (2) The total number of edges crossing the partition is at most $4d\sqrt{c\ell^{-1/5}\log\ell}\sum_{i\le a}|Q_i| + d|T|$. We initialize with the trivial partition $\{T, S = V \setminus T\}$.

As long as $|S| > n/\ell^{1/5}$, we invoke Lemma 3.3.1. We get a new set $Q \subseteq S$ satisfying the first condition of the theorem, and the number of edges from Q to $S \setminus Q$ is at most $4d\sqrt{c\ell^{1/5}\log\ell}|Q|$. We add Q to our partition, reset $S = S \setminus Q$, and iterate.

When this process terminates, $|S| \le n/\ell^{1/5}$. We get the final partition by removing all edges incident to $S \cup T$. Alternately, every single vertex in $S \cup T$ becomes a separate set. Note that a single vertex trivially satisfies the first condition of theorem, since for all s, $p_{s,s}(1) \ge 1/2$. The total number of edges crossing the partition is at most $4dn\sqrt{c\ell^{-1/5}\log\ell} + 2dn\ell^{-1/5} \le 8dn\sqrt{c\ell^{-1/5}\log\ell}$. □

3.4 Proof of main result

Claim 3.4.1. *Consider* `EstClip`(s)*, and recall that T is the set of all vertices such that $w_v \ge k/8$. With probability at least $1 - \exp(-k/128)$ over the randomness in* `EstClip`(s)*: all v such that $\mathbf{p}_s^\ell(v) \ge 1/4k$ are in T, and no v such that $\mathbf{p}_s^\ell(v) \le 1/64k$ is in T.*

Proof. Consider v such that $\mathbf{p}_s^\ell(v) \ge 1/4k$. Recall that the total number of walks is $w = k^2$. The expected value of w_v is at least $w/4k = k/4$. Note that w_v is a sum of Bernoulli random variables. By a multiplicative Chernoff bound (Theorem 1.1 of [DP09]), $\Pr[w_v \le k/8] \le \Pr[w_v \le \mathbf{E}[w_v]/2] \le \exp(-k/16)$. There are at most $4k$ such vertices. By a union bound over all of them, the probability that some such v is not in T is at most $4k \cdot \exp(-k/16) \le \exp(-k/32)$.

For the second part, consider v such that $\mathbf{p}_s^\ell(v) \le 1/64k$. We split into two cases.

Case 1, $\mathbf{p}_s^\ell(v) \ge \exp(-k/32)$. The expectation of w_v is at most $k^2 \cdot \frac{1}{64k} = k/64$. Therefore, $2e\mathbf{E}[w_v] \le k/8$, and by a Chernoff bound (third part, Theorem 1.1 of [DP09]), $\Pr[w_v \ge k/2] \le 2^{-k/8}$. There are at most $\exp(k/32)$ such vertices v. Taking a union bound over all of them, the probability that any such vertex appears in T is at most $\exp(k/32)2^{-k/8} \le \exp(k/32)\exp(-k/16) \le \exp(-k/32)$.

Case 2, $\mathbf{p}_s^\ell(v) < \exp(-k/32)$. For convenience, set $p = \mathbf{p}_s^\ell(v)$. The probability that $w_v \le 1$ is:

$$(1-p)^w + wp(1-p)^{w-1} \ge (1-wp) + wp(1-p(w-1)) = 1 - p^2w(w-1) \ge 1 - p^2w^2 \quad (3.1)$$

(We use the inequality $(1-x)^r \ge 1 - xr$, for $|x| \le 1, r \in \mathbb{N}$.) Thus, the probability that $w_v > 1$ is at most p^2w^2. Note that $k/8$ (the threshold to be placed in T) is at least 2.

Let us take a union bound over all such vertices. We note that $w = k^2$. The probability that any such v is placed in T is at most

$$\sum_{v:\mathbf{p}_s^\ell(v)<\exp(-k/32)} \mathbf{p}_s^\ell(v)^2 w^2 \leq w^2 \exp(-k/32) \sum_v \mathbf{p}_s^\ell(v) \leq \exp(-k/64) \tag{3.2}$$

where the last inequality holds for sufficiently large k (or equivalently, sufficiently small ε).

Union bounding over both errors completes the proof. □

We can now argue about the main guarantee of `EstClip`.

Claim 3.4.2. *For all vertices s, with probability at least $1 - e^{-k/64}$ over the randomness of* `EstClip`(s)*:*

- *If $\|\mathrm{cl}(\mathbf{p}_s^\ell, 1/4)\|_2^2 < \frac{1}{2560k}$, then* `EstClip`$(s)$ *outputs LOW.*
- *If $\|\mathrm{cl}(\mathbf{p}_s^\ell, 3/8)\|_2^2 > \frac{1}{4k}$, then* `EstClip`$(s)$ *outputs HIGH.*

Proof. Consider the first case. Let $H = \{v \mid \mathbf{p}_s^\ell(v) \geq 1/64k\}$. We first argue that $\sum_{v\in H} \mathbf{p}_s^\ell(v) \leq 1/4 + 1/20$. Suppose not. Then, any clipping of $1/4$ of the probability mass of $\mathbf{p}_s^\ell$ leaves at least $1/20$ probability mass on H. The size of H is at most $64k$. By Jensen's inequality, $\|\mathrm{cl}(\mathbf{p}_s^\ell, 1/4)\|_2^2 \geq 1/2560k$, contradicting the case condition.

Thus, $\sum_{v\in H} \mathbf{p}_s^\ell(v) \leq 1/4 + 1/20$. The expected value of $\sum_{v\in H} w_v \leq w(1/4 + 1/20)$. By an additive Chernoff bound (first part, Theorem 1.1 of [DP09]), $\Pr[\sum_{v\in H} w_v \geq w/3] \leq \exp(-2(1/3 - 1/4 - 1/20)^2 w) \leq \exp(-w/10000)$. By Claim 3.4.1, with probability at least $1 - 2\exp(-k/64)$, every vertex in T is such that $\mathbf{p}_s^\ell(v) > 1/64k$, and hence $T \subseteq H$. By a union bound, with probability at least $1 - 3\exp(-k/64) \geq 1 - \exp(-k/128)$, $\sum_{v\in T} w_v \leq \sum_{v\in H} w_v < w/3$, and the output is LOW.

Now for the second case. Let $H' = \{v \mid \mathbf{p}_s^\ell(v) \geq k^{-1}/4\}$. We first argue that $\sum_{v\in H} \mathbf{p}_s^\ell(v) \geq 3/8$. Suppose not. We can clip away all the probability mass of $\mathbf{p}_s^\ell$ that is on H', which is at most $3/8$. All remaining probability/entries of the clipped vector are at most $1/4k$. Thus, the squared l_2-norm is at most $1/4k$, implying $\|\mathrm{cl}(\mathbf{p}_s^\ell, 3/8)\|_2^2 \leq 1/4k$ (contradiction).

Thus, $\sum_{v\in H'} \mathbf{p}_s^\ell(v) \geq 3/8$. By an additive Chernoff bound (first part, Theorem 1.1 of [DP09]), $\Pr[\sum_{v\in H'} w_v < w/3] \leq \exp(-2(3/8 - 1/3)^2 w) \leq \exp(-w/10000)$. By Claim 3.4.1, with probability at least $1 - 2\exp(-k/64)$, $H' \subseteq T$. By a union bound, with probability at least $1 - e^{-k/128}$, $\sum_{v\in T} w_v \geq \sum_{v\in H'} w_v \geq w/3$, and the output is HIGH. □

We break the proof of Theorem 3.1.1 into the following three main claims.

Claim 3.4.3. *With probability at least 2/3,* `IsHyperfinite`(G, ε) *rejects any G which is ε-far from $\mathcal{P}$.*

Proof. We split into two cases.

Case 1: There are less than $(1 - 1/\ell^{1/5})n$ vertices such that $\|\mathrm{cl}(\mathbf{p}_s^\ell, 1/4)\|_2^2 > 1/2560k$.

Then, there are at least $n/\ell^{1/5}$ vertices such that $\|\mathrm{cl}(\mathbf{p}_s^\ell, 1/4)\|_2^2 \leq 1/2560k$. The expected number of such vertices (with repetition) in the multiset S (of Step 1) is at least $|S|\frac{n}{\ell^{1/5}} = \varepsilon^{-34}k^2$. By a multiplicative Chernoff bound, there are at least $\frac{\varepsilon^{-34}k^2}{2}$ such vertices in S with probability at least $1 - \exp(\frac{\varepsilon^{-34}k^2}{8})$. For each such vertex s, the probability that $\texttt{EstClip}(s)$ outputs LOW is at least $1 - \exp(-k/128)$ (Claim 3.4.2). By a union bound over all vertices in S, with probability $> (1 - \exp(-\frac{\varepsilon^{-34}k^2}{8}))(1 - |S|\exp(-k/128)) > 5/6$, there are at least $\frac{\varepsilon^{-34}k^2}{2}$ calls to $\texttt{EstClip}(s)$ that return LOW. So the tester rejects.

Case 2: There are at least $(1 - 1/\ell^{1/5})n$ vertices such that $\|\mathrm{cl}(\mathbf{p}_s^\ell, 1/4)\|_2^2 > 1/2560k$. We apply the decomposition of Theorem 3.3.2 with $c = \frac{\log(2560k)}{\log(\ell)}$, and hence $\ell^{-c} = 1/2560k$, and so the theorem applies. There is a partition $\{P_1, P_2, \ldots, P_b\}$ of the vertices such that:

- For each P_i, there exists $s \in V$ such that: $\forall v \in P_i$, $\sum_{t<25600\ell k} \mathbf{p}_s^t(v) \geq 1/20480\ell k$. Call s the *anchor* for P_i, noting that multiple sets may have the same anchor.
- The total number of edges crossing the partition is at most $8dn\sqrt{\log(2560k)\ell^{-1/5}}$.

Among the sets in the partitbion, let $\{Q_1, Q_2, \ldots, Q_a\}$ be the sets of vertices that induce a graph not in $\mathcal{P}$. Note that one can remove

$$d\sum_{i\leq a}|Q_i| + 8dn\sqrt{\log(2560k)\ell^{-1/5}} \tag{3.3}$$

edges to make G have $\mathcal{P}$ since $\mathcal{P}$ is monotone. Since $\ell = \varepsilon^{-40}$ and $k \leq 2^{\ell^{1/10}}$, the number of edges crossing the partition (the second term of (3.3)), is at most $8dn\sqrt{\log(2560k)\ell^{-1/5}} \leq 1000dn\ell^{-1/20} \leq 1000\varepsilon^2 dn \leq \varepsilon dn/2$ (for sufficiently small ε). Note that this is where we require the upper bound on the hyperfiniteness function, $f(\alpha)$. If $k = f(1/8\ell^2)$ were too large, we would not be able to bound the number of edges crossing the partition. Since G is ε-far from being H-minor free, we deduce from the above that $\sum_{i\leq a}|Q_i| \geq \varepsilon n/2$.

Let $Z = \{s \mid s \text{ is anchor for some } Q_i\}$. Let us lower bound $|Z|$. For every Q_i, there is some $s \in Z$ such that $\forall v \in Q_i$, $\sum_{t<25600\ell k} \mathbf{p}_s^t(v) \geq 1/20480\ell k$. Thus, for every Q_i, there is some $s \in Z$ such that $\sum_{v\in Q_i}\sum_{t<25600\ell k} \mathbf{p}_s^t(v) \geq |Q_i|/20480\ell k$. Let us sum over all $s \in Z$ (and note that $\sum_{v\in V} \mathbf{p}_s^t(v) = 1$).

$$\sum_{i\leq a}|Q_i|/20480\ell k \leq \sum_{s\in Z}\sum_{v\in V}\sum_{t<25600\ell k} \mathbf{p}_s^t(v) \leq \sum_{t<25600\ell k}\sum_{s\in Z}\sum_{v\in V} \mathbf{p}_s^t(v) \leq 25600\ell k|Z| \tag{3.4}$$

Since $\sum_{i\leq a}|Q_i| \geq \varepsilon n/2$, $|Z| \geq c_1\frac{\varepsilon n}{\ell^2k^2}$ where c_1 is some small absolute constant.

Focus on the multiset S in Step 1 of $\texttt{IsHyperfinite}$. Note that S contains an element of Z with probability $\geq 1 - (\frac{c_1\varepsilon}{\ell^2k^2})^{|S|} \geq 9/10$ (recall that $|S| = \varepsilon^{-42}k^2$). Let us condition on this event, and let $s \in S \cap Z$. There exists some Q_i such that $\forall v \in Q_i$, $\sum_{t<25600\ell k} \mathbf{p}_s^t(v) \geq 1/20480\ell k$. By averaging over walk length, $\forall v \in Q_i$, $\exists t < 25600\ell k$ such that $\mathbf{p}_s^t(v) \geq c_2/\ell^2k^2$ where c_2 is a small absolute constant.

Now, consider the call to $\texttt{LocalSearch}(s)$. The set B_s in Step 1 of $\texttt{LocalSearch}$ is constructed by performing $k^3\ell^3$ random walks of length $25600\ell k$. For any $v \in Q_i$, the probability

that v is in B_s is at least $1-(1-\frac{c_2}{\ell^2k^2})^{k^3\ell^3} \geq 1-\exp(-c_2k\ell)$. Taking a union bound over all $v \in Q_i$, the probability that $Q_i \subseteq B_s$ is at least $1-\frac{\ell^2k^2}{c_2}\exp(-c_2k\ell) \geq 9/10$ for sufficiently small ε. When $Q_i \subseteq B_s$, then $G[B_s] \notin \mathcal{P}$ and the tester rejects. The probability of this happening is at least $(9/10)^2 > 2/3$.

In either case, the graph is rejected with probability at least $2/3$, and hence the tester rejects any graph that is ε-far from $\mathcal{P}$. □

Claim 3.4.4. *With probability at least* $2/3$, `IsHyperfinite` (G,ε) *accepts any* $G \in \mathcal{P}$.

Proof. Let us turn our attention to graphs which are in $\mathcal{P}$. Note that since $\mathcal{P}$ is monotone, calls to `LocalSearch` can never return FOUND, so rejection can only happen because of the output of calls to `EstClip`.

By Lemma 3.2.2, since G is $(1/8\ell^2,k)$-hyperfinite, there are at least $(1-1/\ell)n$ vertices such that $\|\mathrm{cl}(\mathbf{p}_s^\ell, 3/8)\|_2^2 \geq 1/4k$. Call these vertices *heavy*. The expected number of light vertices in the multiset S chosen in Step 1 of `IsHyperfinite` is at most $1/\ell \times |S| = \varepsilon^{-2}\ell k^2$. By a multiplicative Chernoff bound (Theorem 1 of [DP09]), the number of light vertices in S is strictly less than $\frac{\varepsilon^{-34}k^2}{2}$ with probability at least $1-\exp(-c_2\ell^{13/5}k^2) > 9/10$ for a large constant c_2. Let us condition on this event. The probability that any call to `EstClip`(s) returns HIGH for a heavy $s \in S$ is at least $1-\exp(-k/128)$ by Claim 3.4.2. By a union bound over the at most $\varepsilon^{-42}k^2$ heavy vertices in S, all calls to `EstClip`(s) for heavy $s \in S$ return HIGH with probability at least $1-(\varepsilon^{-42}k^2)\exp(-k/128) > 9/10$.

We now remove the conditioning. With probability $> (9/10)^2 > 2/3$, there are strictly less than $\frac{\varepsilon^{-34}k^2}{2}$ calls (for the light vertices) that return LOW. When this happens, `IsHyperfinite` accepts. □

Claim 3.4.5. *There exist fixed polynomials,* p *and* q*, such that the query complexity of* `IsHyperfinite` *is* $O\left(p\left(df\left(q\left(1/\varepsilon\right)\right)\right)\right)$.

Proof. Note that $k = f(q(1/\varepsilon))$ for a polynomial, q. `LocalSearch` performs $\mathrm{poly}(k)$ walks of length $\mathrm{poly}(k)$, and in order to determine the subgraph $G[B_s]$, it requires d queries for every vertex encountered. Therefore, each call to `LocalSearch` requires $d\mathrm{poly}(k)$ queries. Each call of `EstClip` performs $8k^3$ queries. Both `LocalSearch` and `EstClip` are called $\mathrm{poly}(k)$ times, and hence the overall query complexity is $d\mathrm{poly}(k)$. □

Chapter 4

An efficient partition oracle

The algorithmic study of planar graphs is a fundamental direction in theoretical computer science and graph theory. Classic results like the Kuratowski-Wagner characterization [Kur30, Wag37], linear time planarity algorithms [HT74], and the Lipton-Tarjan separator theorem underscore the significance of planar graphs [LT80]. The celebrated theory of Robertson-Seymour give a grand generalization of planar graphs through minor-closed families [RS95a, RS95b, RS04]. This has led to many deep results in graph algorithms, and an important toolkit is provided by separator theorems and associated decompositions [AST94].

Over the past decade, there have been many advances in *sublinear* algorithms for planar graphs and minor-closed families. We focus on the model of random access to bounded degree adjacency lists, introduced by Goldreich-Ron [GR02]. Let $G = (V, E)$ be a graph with vertex set $V = [n]$ and degree bound d. The graph is accessed through *neighbor queries*: there is an oracle that on input $v \in V$ and $i \in [d]$, returns the ith neighbor of v. (If none exist, it returns $\perp$.)

One of the key properties of bounded-degree graphs in minor-closed families is that they exhibit hyperfinite decompositions. A graph G is hyperfinite if $\forall\, 0 < \varepsilon < 1$, one can remove εdn edges from G and obtain connected components of size independent of n (we refer to these as pieces). For minor-closed families, one can remove εdn edges and get pieces of size $O(\varepsilon^{-2})$.

The seminal result of Hassidim-Kelner-Nguyen-Onak (HKNO) [HKNO09] introduced the notion of *partition oracles*. This is a local procedure that provides "constant-time" access to a hyperfinite decomposition. The oracle takes a query vertex v and outputs the piece containing v. Each piece is of size independent of n, and at most εdn edges go between pieces. Furthermore, all the answers are consistent with a single hyperfinite decomposition, despite there being no preprocessing or explicit coordination. (All queries uses the same random seed, to ensure consistency.) Partition oracles are extremely powerful as they allow a constant time procedure to directly access a hyperfinite decomposition. As observed in previous work, partition oracles

lead to a plethora of property testing results and sublinear time approximation algorithms for minor-closed graph families [HKNO09, NS13]. In some sense, one can think of partition oracles as a moral analogue of Szémeredi's regularity lemma for dense graph property testing: it is a decomposition tool that immediately yields a litany of constant time (or constant query) algorithms.

We give a formal definition of partition oracles. (We deviate somewhat from the definition in Chap. 9.5 of Goldreich's book [Gol17] by including the running time as a parameter, instead of the set size.)

Definition 4.0.1. *Let $\mathcal{P}$ be a family of graphs with degree bound d and $T : (0,1) \to \mathbb{N}$ be a function. A procedure $\boldsymbol{A}$ is an* $(\varepsilon, T(\varepsilon))$-partition oracle *for $\mathcal{P}$ if it satisfies the following properties. The deterministic procedure takes as input random access to $G = (V, E)$ in $\mathcal{P}$, random access to a random seed r (of length polynomial in graph size), a proximity parameter $\varepsilon > 0$, and a vertex v of G. (We will think of fixing G, r, ε, so we use the notation $\boldsymbol{A}_{G,r,\varepsilon}$. All probabilities are with respect to r.) The procedure $\boldsymbol{A}_{G,r,\varepsilon}(v)$ outputs a set of vertices and satisfies the following properties.*

1. *(Consistency) The sets $\{\boldsymbol{A}_{G,r,\varepsilon}(v)\}$, over all v, form a partition of V. Also, these sets $\boldsymbol{A}_{G,r,\varepsilon}(v)$ induce connected graphs for all $v \in V$.*
2. *(Cut bound) With probability (over r) at least $2/3$, the number of edges between the sets $\boldsymbol{A}_{G,r,\varepsilon}(v)$ is at most εdn.*
3. *(Running time) For every v, $\boldsymbol{A}_{G,r,\varepsilon}(v)$ runs in time $T(\varepsilon)$.*

We stress that there is no explicit "coordination" or sharing of state between calls to $\boldsymbol{A}_{G,r,\varepsilon}(v)$ and $\boldsymbol{A}_{G,r,\varepsilon}(v')$ (for $v \neq v'$). There is no global preprocessing step once the random seed is fixed. The consistency guarantee holds with probability 1. Note that the running time $T(\varepsilon)$ is clearly an upper bound on the size of the sets $\boldsymbol{A}_{G,r,\varepsilon}(v)$. For minor-closed families, one can convert any partition oracle to one that output sets of size $O(\varepsilon^{-2})$ with a constant factor increase in the cut bound. (refer to the end of Sec. 9.5 in [Gol17]).

The challenge in partition oracles is to bound the running time $T(\varepsilon)$. HKNO gave a partition oracle with running time $(d\varepsilon^{-1})^{\mathrm{poly}(d\varepsilon^{-1})}$. Levi-Ron [LR15] built on the ideas from HKNO and dramatically improved the bound to $(d\varepsilon^{-1})^{\log(d\varepsilon^{-1})}$. Yet, for all minor-closed families, one can (in linear time) remove εdn edges to get connected components of size $O(\varepsilon^{-2})$. HKNO raise the natural open question as to whether $(\varepsilon, \mathrm{poly}(d\varepsilon^{-1}))$-partition oracles exist.

In this paper, we resolve this open problem.

Theorem 4.0.2. *Let $\mathcal{P}$ be the set of d-bounded degree graphs in a minor-closed family. There is an $(\varepsilon, \mathrm{poly}(d\varepsilon^{-1}))$-partition oracle for $\mathcal{P}$.*

4.0.1 Consequences

As observed by HKNO and Newman-Sohler [NS13], partition oracles have many consequences for property testing and sublinear algorithms.

Recall the definition of property testers. Let $\mathcal{Q}$ be a property of graphs with degree bound d. The distance of G to $\mathcal{Q}$ is the minimum number of edge additions/removals required to make G have $\mathcal{Q}$, divided by dn. A property tester for $\mathcal{P}$ is a randomized procedure that takes query access to an input graph G and a proximity parameter, $\varepsilon > 0$. If $G \in \mathcal{P}$, the tester accepts with probability at least $2/3$. If the distance of G to $\mathcal{Q}$ is at least ε, the tester rejects with probability at least $2/3$. We often measure the query complexity as well as time complexity of the tester.

A direct consequence of Theorem 4.0.2 is an "efficient" analogue (for monotone and additive properties) of a theorem of Newman-Sohler stating that all properties of hyperfinite graphs are testable. A graph property closed under vertex/edge removals is called *monotone*. A graph property closed under disjoint union of graphs is called *additive*.

Theorem 4.0.3. *Let $\mathcal{Q}$ be any monotone and additive property of bounded degree graphs of a minor-closed family. There exists a* $\mathrm{poly}(d\varepsilon^{-1})$*-query tester for $\mathcal{Q}$.*

If membership in $\mathcal{Q}$ can be determined exactly in polynomial (in input size) time, then $\mathcal{Q}$ has $\mathrm{poly}(d\varepsilon^{-1})$*-time testers.*

An appealing consequence of Theorem 4.0.3 is that the property of bipartite planar graphs can be tested in $\mathrm{poly}(d\varepsilon^{-1})$ time. For any fixed subgraph H, the property of H-free planar graphs can be tested in the same time. And all of these bounds hold for any minor-closed family.

As observed by Newman-Sohler, partition oracles give sublinear query algorithms for any additive graph parameter that is "robust" to edge changes. Again, Theorem 4.0.2 implies an efficient version for minor-closed families.

Theorem 4.0.4. *Let f be a real-valued function on graphs that changes by $O(1)$ on edge addition/removals, and has the property that $f(G_1 \cup G_2) = f(G_1) + f(G_2)$ for graphs G_1, G_2 that are not connected to each other.*

For any minor-closed family $\mathcal{P}$, there is a randomized algorithm that, given $\varepsilon > 0$ and $G \in \mathcal{P}$, outputs an additive εn-approximation to $f(G)$ and makes $\mathrm{poly}(d\varepsilon^{-1})$ *queries. If f can be computed exactly in polynomial time, then the above algorithm runs in* $\mathrm{poly}(d\varepsilon^{-1})$ *time.*

The functions captured by Theorem 4.0.4 are quite general. Functions such as maximum matching, minimum vertex cover, maximum independent set, minimum dominating set, maxcut, etc. all have the robustness property. As a compelling application of Theorem 4.0.4,

we can get $(1+\varepsilon)$-approximations[1] for the maximum matching in planar (or any minor-closed family) graphs in $\text{poly}(d\varepsilon^{-1})$ time.

These theorems are easy consequences of Theorem 4.0.2. Using the partition oracle, an algorithm can essentially assume that the input is a collection of connected components of size $\text{poly}(d\varepsilon^{-1})$, and run an exact algorithm on a collection of randomly sampled components. We sketch the proofs in §4.7.

4.0.2 Related work

The subject of property testing and sublinear algorithms in bounded degree graphs is a vast topic. We refer the reader to Chapters 9 and 10 of Goldreich's textbook [Gol17]. We focus on the literature relevant to sublinear algorithms for minor-closed families.

The first step towards a characterization of testable properties in the bounded-degree model was given by Czumaj-Sohler-Shapira, who showed hereditary properties in non-expanding graphs are testable [CSS09]. This was an indication that notions like hyperfiniteness are connected to property testing. Benjamini-Schramm-Shapira achieved a breakthrough by showing that all minor-closed properties are testable, in time triply-exponential in $d\varepsilon^{-1}$ [BSS10]. Hassidim-Kelner-Nguyen-Onak introduced partition oracles, and designed one running in time $\exp(d\varepsilon^{-1})$. Levi-Ron improved this bound to quasipolynomial in $d\varepsilon^{-1}$, using a clever analysis inspired by algorithms for minimum spanning trees [LR15]. Newman-Sohler built on partition oracles for minor-close families to show that all properties of hyperfinite graphs are testable [NS13]. Fichtenberger-Peng-Sohler showed any testable property contains a hyperfinite property [**?**].

There are two dominant combinatorial ideas in this line of work. The first is using subgraph frequencies in neighborhood of radius $\text{poly}(\varepsilon^{-1})$ to characterize properties. This naturally leads to exponential dependencies in $\text{poly}(\varepsilon^{-1})$. The second idea is to use random edge contractions to reduce the graph size. Recursive applications lead to hyperfinite decompositions, and the partition oracles of HKNO and Levi-Ron simulate this recursive procedure. This is extremely non-trivial, and leads to a recursive local procedure with a depth dependent of ε. Levi-Ron do a careful simulation, ensuring that the recursion depth is at most $\log(d\varepsilon^{-1})$, but this simulation requires looking at neighborhoods of radius $\log(d\varepsilon^{-1})$. Following this approach, there is little hope of getting a recursion depth independent of ε, which is required for a $\text{poly}(d\varepsilon^{-1})$-time procedure.

Much of the driving force behind this work was the quest for a $\text{poly}(d\varepsilon^{-1})$-time tester for planarity. This question was resolved recently using a different approach from spectral graph theory, which was itself developed for sublinear time algorithms for finding

[1] The maximum matching is $\Omega(n/d)$ for a connected bounded degree graph. One simply sets $\varepsilon \ll 1/d$ in Theorem 4.0.4.

minors [KSS18, KSS19b]. A major inspiration is the random walk based one-sided bipartiteness tester of Goldreich-Ron [GR99]. This paper is a continuation of that line of work, and is a further demonstration of the power of spectral techniques for sublinear algorithms. The tools build on local graph partitioning techniques pioneered by Spielman-Teng [ST12], which is itself based on classic mixing time results of Lovász-Simonovits [LS90b]. In this paper, we develop new diffusion-based local partitioning tools that form the core of partition oracles.

We also mention other key results in the context of sublinear algorithms for minor-closed families, notably the Czumaj et al [CGR+14] upper bound of $O(\sqrt{n})$ for testing cycle minor-freeness, the Fichtenberger et al [FLVW17] upper bound of $O(n^{2/3})$ for testing $K_{2,r}$-minor-freeness, and poly$(d\varepsilon^{-1})$ testers for outerplanarity and bounded treewidth graphs [YI15, EHNO11].

4.1 Main Ideas

The starting point for this work are the spectral methods used in [KSS18, KSS19b]. These methods discover cut properties within a neighborhood of radius poly$(d\varepsilon^{-1})$, without explicitly constructing the entire neighborhood.

One of the key tools used in these results in a local partitioning algorithm, based on techniques of Spielman-Teng [ST12]. The algorithm takes a seed vertex s, performs a diffusion from s (equivalently, performs many random walks) of length poly$(d\varepsilon^{-1})$, and tracks the diffusion vector to detect a low conductance cut around s in poly$(d\varepsilon^{-1})$ time. We will use the term *diffusions*, instead of random walks, because we prefer the deterministic picture of a unit of "ink" spreading through the graph. A key lemma in previous results states that, for graphs in minor-closed families, this procedure succeeds from more than $(1-\varepsilon)n$ seed vertices. This yields a global algorithm to construct a hyperfinite decomposition with components of poly$(d\varepsilon^{-1})$ size. Pick a vertex s at random, run the local partitioning procedure to get a low conductance cut, remove and recurse. Can there be a local implementation of this algorithm?

Let us introduce some setup. We will think of a global algorithm that processes seed vertices in some order. Given each seed vertex s, a local partitioning algorithm generates a low conductance set $C(s)$ containing s (this is called a cluster). The final output is the collection of these clusters. For any vertex v, let the *anchor* of v be the vertex s such that $v \in C(s)$. A local implementation boils down to finding the anchor of query vertex v.

Observe that at any point of the global procedure, some vertices have been clustered, while the remaining are still *free*. The global procedure described above seems hopeless for a local implementation. The cluster $C(s)$ is generated by diffusion in some subgraph G' of G, which was the set of free vertices when seed s was processed. Consider a local procedure trying to discover the anchor of v. It would need to figure out the free set corresponding to every

potential anchor s, so that it can faithfully simulate the diffusion used to cluster v. From an implementation standpoint, it seems that the natural local algorithm is to use diffusions from v in G to discover the anchor. But diffusion in a subgraph G' is markedly different from G and difficult to simulate locally. Our first goal is to design a partitioning method using diffusions directly in G.

Finding low conductance cuts in subsets, by diffusion in supersets: Let us now modify the global algorithm with this constraint in mind. At some stage of the global algorithm, there is a set F of free vertices. We need to find a low conductance cut contained in F, while running random walks in G. Note that we must be able to deal with F as small as $O(\varepsilon n)$. Thus, random walks (even starting from F) will leave F quite often; so how can these walks/diffusions find cuts in F?

One of our main insights is that these challenges can be dealt with, even for diffusions of $\text{poly}(d\varepsilon^{-1})$ length. We show that, for a uniform random vertex $s \in F$, a spectral partitioning algorithm that performs diffusion from s in G can detect low conductance cuts contained in F. Diffusion in the superset (all of V) provides information about the subset F. This is a technical and non-trivial result, and crucially uses the spectral properties of minor-closed families. Note that diffusions from F can spread very rapidly in short random walks, even in planar graphs. Consider a graph G, where F is a path on εn vertices, and there is a tree of size $1/\varepsilon$ rooted at every vertex of F. Diffusions from any vertex in F will initially be dominated by the trees, and one has to diffuse for at least $1/\varepsilon$ timesteps before structure within F can be detected. Thus, the proof of our theorem has to look at average behavior over a sufficiently large time horizon before low conductance cuts in F are "visible". Remarkably, it suffices to look at $\text{poly}(d\varepsilon^{-1})$ timesteps to find structure in F, because of the behavior of diffusions in minor-closed families.

The main technical tool used is the Lovász-Simonovits curve technique [LS90b], whose use was pioneered by Spielman-Teng [ST12]. We also use the truncated probability vector technique from Spielman-Teng to give cleaner implementations and proofs. A benefit of using diffusion (instead of random walks) on truncated vectors is that the clustering becomes deterministic.

The problem of ordering the seeds: With one technical hurdle out of the way, we end up at another gnarly problem. The above procedure only succeeds if the seed is in F. Quite naturally, one does not expect to get any cuts in F by diffusing from a random vertex in G. From the perspective of the global algorithm, this means that we need some careful ordering of the seeds, so that low conductance cuts are discovered. Unfortunately, we also need local implementations of this ordering. The authors struggled with carrying out this approach, but to no avail.

To rid ourselves of the ordering problem, let us consider the following, almost naive global algorithm. First, order the vertices according to a uniform random permutation. At

any stage, there is a free set F. We process the next seed vertex s by running some spectral partitioning procedure, to get a low conductance cut $C(s)$. Simply output $C(s) \cap F$ (instead of $C(s)$) as the new cluster, and update F to $F \setminus C(s)$. It is easy to locally implement this procedure. To find the anchor of v, perform a diffusion of $\text{poly}(\varepsilon^{-1})$ timesteps from v. For every vertex s with high enough value in the diffusion vector, determine if $C(s) \ni v$. The vertex s that is lowest according to the random ordering is the anchor of v. Unfortunately, there is little hope of bounding the number of edges cut by the clustering. When s is processed, it may be that $s \notin F$, and there is no guarantee of $C(s) \cap F$. Can we modify the procedure to bound the number of cut edges, but still maintain its ease of local implementability?

The amortization argument: Consider the scenario when $F = \Theta(\varepsilon n)$. Most of the subsequent seeds processed are not in F and there is no guarantee on the cluster conductance. But every $\Theta(1/\varepsilon)$ seeds (in expectation), we will get a "good" seed s contained in F, such that $C(s) \cap F$ is a low conductance set. (This is promised by the diffusion algorithm that we develop in this paper, as discussed earlier.) Our aim is to perform some amortization, to argue that $|C(s) \cap F|$ is so large, that we can "charge" away the edges cut by the previous $\Theta(1/\varepsilon)$ seeds.

This amortization is possible because our spectral tools give us much flexibility in the (low) conductances obtained. Put differently, we essentially prove that existence of many cuts of extremely low conductance, and show that it is "easy" for a diffusion-based algorithm to find such cuts. (This is connected to the spectral behavior of minor-closed families.) As a consequence, we can actually pre-specify the size of the low conductance cuts obtained. We show that as long as $|F| = \Omega(\varepsilon n)$, we can find a *size threshold* $k = \text{poly}(\varepsilon^{-1})$ such that for at least $\Omega(\varepsilon^2 n)$ vertices $s \in F$, a spectral partitioning procedure seeded at s can find a cut of size $\Theta(k)$ and conductance at most ε^c. Moreover, this cut is guaranteed to contain at least $\varepsilon^{c'} k$ vertices in F, despite the procedure being oblivious to F. The parameter c can be easily tuned, so we can increase c arbitrarily while keeping c' fixed, at the cost of polynomial increases in running time. This tunability is crucial to our amortization argument. We also show that given query access to F, a size threshold k can be computed in $\text{poly}(d\varepsilon^{-1})$ time.

So when the global algorithm processes seed s, it runs the above spectral procedure to try to obtain a set of size $\Theta(k)$ with conductance at most ε^c. (If the procedure fails, the global algorithm simply set $C(s) = \{s\}$.) Thus, we cut $O(\varepsilon^c kd)$ edges for each seed processed. But after every $O(1/\varepsilon)$ seeds, we choose a "good" seed such that $|C(s) \cap F| > \varepsilon^{c'} k$. The total number of edges cut is $O(\varepsilon^c kd \times \varepsilon^{-1}) = O(\varepsilon^{c-1} kd)$. The total number of new vertices clustered is at least $\varepsilon^{c'} k$. Because we can tune parameters with much flexibility, we can set $c \gg c'$. So the total number of edges cut is $O(\varepsilon^{c-c'-1} d)$ times the number of vertices clustered, where $c - c' - 1 > 1$. Overall, we will cut only $O(\varepsilon nd)$ edges.

Making it work through phases: Unfortunately, as the process described above continues, F shrinks. Thus, the original choice of k might not work, and the guarantees on

$|C(s) \cap F|$ for good seeds no longer hold. So we need to periodically recompute the value of k. In a careful analysis, we show that this recomputation is only required $\text{poly}(\varepsilon^{-1})$ times. Formally, we implement the recomputation through *phases*. Each vertex is independently assigned to one of $\text{poly}(\varepsilon^{-1})$ phases. (Technically, we choose the phase of a vertex by sampling an independent geometric random variable. We heavily use the memoryless property of the geometric distribution.)

For each phase, the value of k is fixed. The local partition oracle will compute these size thresholds for all phases, as a $\text{poly}(d\varepsilon^{-1})$ time preprocessing step. The oracle (for v) runs a diffusion from v to get a collection of candidate anchors. For each candidate s, the oracle determines its phase, runs the spectral partitioning algorithm with correct phase parameters, and determines if the candidate's low conductance cut contains v. The anchor is simply such a candidate of minimum phase, with ties broken by vertex id.

4.1.1 Outline of sections

The algorithm description and proof has many moving parts, encapsulated by different sections. §4.2 begins by discussing the truncated diffusion process, the main algorithmic tool for partitioning. We then describe the global partitioning algorithm `globalPartition` (modulo a preprocessing step called `findr`), which is far more convenient to analyze. It will be readily apparent that this global procedure outputs a partition of G into connected components; the main challenge is to bound the number of edges cut.

Within §4.2, we discuss how to implement `globalPartition` by a local procedure. By ensuring that the output of the local procedure is identical to `globalPartition`, we prove the consistency property of Def 4.0.1. We then perform a fairly straightforward running time analysis, which proves the running time property of Def 4.0.1.

The real heavy lifting begins in §4.3, where we describe the procedure `findr` that computes the size thresholds. This section is devoted to proving salient properties of the size thresholds output by `findr`. The analysis hinges on the diffusion and cut properties stated in Theorem 4.3.1, which is the main tool connecting minor-freeness, diffusions, and local partitioning. §4.4 uses all these tools to prove the cut bound of `globalPartition`. At this stage, the complete description and guarantees of the partition oracle are complete, modulo the proof of Theorem 4.3.1.

The proof of Theorem 4.3.1 is split into sections. In §4.5, we use the hyperfiniteness of minor-closed families to prove properties of truncated diffusions on minor-free families. §4.6 has the key spectral calculations, where the Lovász-Simonovits curve technique is used to find low conductance cuts. This section has the crucial insights that allow for partitioning in the free set, using diffusions in the overall graph.

§4.7 has short proofs of the applications Theorem 4.0.3 and Theorem 4.0.4. These are

provided for completeness, since identical calculations appear in the proof of Theorem 9.28 in [Gol17].

4.2 Global partitioning and its local implementation

There are a number of parameters that are used in the algorithm. We list them out here for reference. It is convenient to fix the value of ε in advance, so that all the values of the following parameters are fixed. Note that all these parameters are polynomial is ε. We will express all running times as polynomials in these parameters, ensuring all running time are $\text{poly}(\varepsilon^{-1})$.

- $\rho = d^{-60}\varepsilon^{3000}$: Minimum probability for truncation.
- $\ell = d^6\varepsilon^{-30}$: Maximum random walk length.
- $\beta = \varepsilon/10$: Unclustered fraction cutoff.
- $\delta = d^{-70}\varepsilon^{3100}$: Phase probability.
- $\alpha = \frac{\varepsilon^{4/3}}{300{,}000}$: Heavy bucket parameter.
- $\phi = \varepsilon^{10}$: Conductance parameter.

4.2.1 Truncated diffusion

The main process used to find sets of the partition is a *truncated diffusion*. We assume that the input graph G is connected, has n vertices, and degree bound d. Define the symmetric random walk matrix M as follows. For every edge (u, v), $M_{u,v} = M_{v,u} = 1/2d$. For every vertex v, $M_{v,v} = 1 - d(v)/2d$, where $d(v)$ is the degree of v. The matrix M is doubly stochastic, symmetric, and the (unique) stationary distribution is the uniform distribution.

Given a vector $\vec{x} \in (\mathbb{R}^+)^n$, diffusion is the evolution $M^t\vec{x}$. We define a truncated version, where after every step, small values are removed. For any vector $\vec{x}$, let $\text{supp}(\vec{x})$ denote the support of the vector.

Definition 4.2.1. *Define the operator $\widehat{M}\colon (\mathbb{R}^+)^n \to (\mathbb{R}^+)^n$ as follows. For $\vec{x} \in (\mathbb{R}^+)^n$, the vector $\widehat{M}\vec{x}$ is obtained by zeroing out all coordinates in $M\vec{x}$ whose value is at most ρ.*

For $t > 1$, the operator $\widehat{M}^t$ is the t-step truncated diffusion, and is recursively defined as $\widehat{M}(\widehat{M}^{t-1}\vec{x})$.

Define $\widehat{p}_{v,t}(w)$ to be the coordinate corresponding to vertex w in the t-step truncated diffusion starting from vertex v.

We stress that the t-step truncated diffusion is obtained from a standard diffusion by truncating low values at *every* step of the diffusion. Note that as the truncated diffusion progresses, the l_1-norm of the vector may decrease at each step. Importantly, for any distribution vector $\vec{x}$, $\mathrm{supp}(\widehat{M}^t\vec{x})$ has size at most ρ^{-1}. We heavily use this property in our running time analysis.

We define *level sets*, a standard concept in spectral partitioning algorithms. Somewhat abusing notation, for vertex $v \in V$, we use $\vec{v}$ to denote the unit vector in $(\mathbb{R}^+)^n$ corresponding to the vertex v. (We never use to vector notation for any other kind of vectors.)

Definition 4.2.2. *For vertex $v \in V$, length t, and threshold k, let $L_{v,t,k}$ be the set of vertices corresponding to the k largest coordinates in $\widehat{M}^t\vec{v}$ (ties are broken by vertex id).*

For any set S of vertices, the conductance of S is

$$\Phi(S) := E(S, \overline{S})/[2\min(|S|, |\overline{S}|)d]$$

where $E(S, \overline{S})$ denotes the number of edges between S and its complement.

We describe the key subroutine that finds low conductance cuts. It performs a sweep cut over the truncated diffusion vector.

`cluster`(v, t, k)

1. Determine $\widehat{M}^t\vec{v}$
2. For all $k' \in [k, 2k]$ calculate $\Phi(L_{v,t,k'})$.
3. Find the largest $k' \in [k, 2k]$ (if any) with the following properties: $\Phi(L_{v,t,k'} \cup \{v\}) \leq \phi$ and $L_{v,t,k'} \in \mathrm{supp}(\widehat{M}^t\vec{v})$.
4. If such a k' exists, set $C := L_{v,t,k'} \cup \{v\}$, else $C := \{v\}$.
5. Return C.

Claim 4.2.3. *The procedure `cluster`(v, t, k) runs in time $O(\rho^{-1}td\log(\rho^{-1}td) + kd\log k)$. The output set C has the following properties. (i) $v \in C$. (ii) If C is not a singleton, then $|C| \in [k, 2k]$, $\Phi(C) \leq \phi$, and $C \subseteq \mathrm{supp}(\widehat{M}^t\vec{v})$.*

Proof. The latter properties are apparent from the description of `cluster`.

We analyze the running time. The convenience of the truncated diffusion is that it can computed exactly by a deterministic process. First, for any $b \geq 1$, we show that the running time to compute $\widehat{M}^b\vec{v}$ is $O(\rho^{-1}bd)$. Note that for any t, $\mathrm{supp}(\widehat{M}^b\vec{v})$ has size at most ρ^{-1}, since M is a stochastic matrix and all non-zero entries in $\widehat{M}^b\vec{v}$ have value at least ρ. Given the vector $\widehat{M}^b\vec{v}$, the vector $\widehat{M}^{b+1}\vec{v}$ can be computed by determining $M\widehat{M}^b\vec{v}$ and then zeroing out coordinates that are less than ρ. This process can be done in $O(d|\mathrm{supp}(\widehat{M}^b\vec{v})|) = O(\rho^{-1}d)$. By summing this running time over all timesteps, we get that the total time is $O(\rho^{-1}bd)$.

Thus, $\widehat{M}^t\vec{v}$ can be computed exactly in $O(\rho^{-1}td)$ time. To compute the level sets, one can sort the coordinates of this vector (breaking ties by id), and process them in decreasing

order. One can iteratively store $L_{v,t,k}$ in a dictionary data structure. Given $\Phi(L_{v,t,k})$, one can compute $\Phi(L_{v,t,k+1})$ by $O(d)$ lookups into the dictionary. The total running time of this step is $O(kd \log k)$. □

4.2.2 The global partitioning procedure

The global partitioning procedure `globalPartition` will output a partition of the vertices satisfying the conditions in Def 4.0.1. This global procedure will run in linear time. In the next subsection, we show how the output of the global procedure can be generated locally in $\text{poly}(\varepsilon^{-1})$ time, thereby giving us the desired partition oracle. It will be significantly easier to understand and analyze the partition properties of the global procedure.

The key ingredient in `globalPartition` that allows for a local implementation is a *preprocessing* step. The preprocessing allows for the "coordination" required for consistency of various local partitioning steps. All the randomness is used in the preprocessing, after which the actual partitioning is deterministic. The job of the preprocessing is to find the following sets of values, which are used for two goals: (i) ordering vertices, (ii) setting parameters for calls to `cluster`.

The preprocessing generates, for all vertices v, the following values.

- h_v: The *phase* of v.
- k_v: The size threshold of v.
- t_v: The walk length of v.

Before giving the procedure description, we explain how these values are generated.

Phases: For each v, h_v is set to $\max(X, \overline{h})$, where X is independently sampled from $Geo(\delta)$, the geometric distribution with parameter δ. Moreover $\overline{h} := 2\delta^{-1}\log(\delta^{-1})$, so the maximum phase value is capped.

Size thresholds: The computation of these thresholds is the most complex part of our algorithm (and analysis), and is the "magic ingredient" that makes the partition oracle possible. We first run a procedure `findr` that runs in $\text{poly}(\varepsilon^{-1})$ time and outputs a set of *phase size thresholds* $k_1, k_2, \ldots, k_{\overline{h}}$. All the thresholds have value at most ρ^{-1} and $k_{\overline{h}}$ will be zero. The (involved) description of `findr` and its properties are in §4.3. For now, it suffices to say that its running time is $\text{poly}(\varepsilon^{-1})$, and that it outputs phase size thresholds. The size threshold for a vertex v is simply k_{h_v}, corresponding to the phase it belongs to.

Walk lengths: These are simply chosen independently and uniformly in $[1, \ell]$.

The analysis is more transparent when we assume that all the randomness used by the algorithm is in a random seed $\boldsymbol{R}$, of $O(n \cdot \text{poly}(\varepsilon^{-1}))$ length. The seed $\boldsymbol{R}$ is passed as an argument to the partitioning procedure, which uses $\boldsymbol{R}$ to generate all the values described above. (For convenience, we will assume random access to the adjacency list of G, without passing the

graph as a parameter.)

It is convenient to define an ordering on the vertices, given these values. For cleaner notation, we drop the dependence on $\boldsymbol{R}$.

Definition 4.2.4. *For vertex $u, v \in V$, we say that $u \prec v$ if: $h_u < h_v$ or if $h_u = h_v$, the id of v is less than that of v.*

> `globalPartition`($\boldsymbol{R}$)
>
> Preprocessing:
>
> 1. For every $v \in V$:
>   (a) Use $\boldsymbol{R}$ to set $h_v := \max(X, \overline{h})$ ($X \sim Geo(\delta)$).
>   (b) Use $\boldsymbol{R}$ to set t_v uniform random in $[1, \ell]$.
> 2. Call `findr`($\boldsymbol{R}$) to generate values $k_1, k_2, \ldots, k_{\overline{h}}$. For every $v \in V$, set $k_v = k_{h_v}$.
>
> Partitioning:
>
> 1. Initialize the partition $\boldsymbol{P}$ as an empty collection. Initialize the free set $F := V$.
> 2. For all vertices in V in increasing order of $\prec$:
>   (a) Compute $C =$ `cluster`(v, t_v, k_v).
>   (b) Add the connected components of $C \cap F$ to the partition $\boldsymbol{P}$.
>   (c) Reset $F = F \setminus C$.
> 3. Output $\boldsymbol{P}$.

Since all of our subsequent discussions are about `globalPartition`, we abuse notation assuming that the preprocessing is fixed. We refer to `cluster`(v) to denote `cluster`(v, t_v, k_v). These are the only calls to cluster that are ever discussed, so it is convenient to just parametrize by the vertex argument. Furthermore, for ease of notation, we sometimes refer to the output of the procedure as `cluster`(v).

We observe that the output $\boldsymbol{P}$ is indeed a partition of V into connected components. At any intermediate step, the free set F is precisely the set of vertices that have not been assigned to a cluster. Note that `cluster`(v) always contains v (Claim 4.2.3), so all vertices eventually enter (the sets of) $\boldsymbol{P}$.

We note that v might not be in F when `cluster`(v) is called. This may lead to new components in $\boldsymbol{P}$ that do not involve v, which may actually not be low conductance cuts. This may seem like an oversight: why initiate diffusion clusters from vertices that are already partitioned? Many challenges in our analysis arise from such clusters. On the other hand, such an "oblivious" partitioning scheme leads to a simple local implementation.

4.2.3 The local implementation

A useful definition in the local implementation is that of *anchors* of vertices. As mentioned earlier, we fix the output of the preprocessing (which is equivalent to fixing $\boldsymbol{R}$).

Definition 4.2.5. *Consider the running of* `globalPartition`($\boldsymbol{R}$). *The* anchor *of w is the (unique) vertex w such that the component in $\boldsymbol{P}$ containing v was created by the call to* `cluster`(v).

Suppose we label every vertex by its anchor. We can easily determine the sets of $\boldsymbol{P}$ locally.

Claim 4.2.6. *The sets of $\boldsymbol{P}$ are exactly the maximal connected components of vertices with the same anchor.*

Proof. We prove by induction over the $\prec$ ordering of vertices. The base case is vacuously true. Suppose, just before v is considered, all current sets in $\boldsymbol{P}$ are maximal connected components with the same anchor, which cannot be v. No vertex in F can have an anchor yet; otherwise, it would be clustered and part of (a set in) $\boldsymbol{P}$. All the new vertices clustered have v as anchor. Moreover, the sets added to $\boldsymbol{P}$ are precisely the maximal connected components with v as anchor. □

We come to a critical definition that allows for searching for anchors. We define the "inverse ball" of a vertex: this is the set of all vertices that reach v through truncated diffusions. We note that reachability is not symmetric, because the diffusion is truncated at every step.

Definition 4.2.7. *For $v \in V$, let $IB(v) = \{w \mid \exists t \in [0, \ell], v \in \mathrm{supp}(\widehat{M}^t \vec{w})\}$.*

Claim 4.2.8. $|IB(v)| \leq \ell\rho^{-1}$.

Proof. All vertices $w \in IB(v)$ have the property that (for some $t \leq \ell$) $\widehat{p}_{w,t}(v) \neq 0$. That implies that $p_{w,t}(v) \geq \rho$. By the symmetry of the random walk, $p_{v,t}(w) \geq \rho$. For any fixed t, there are at most ρ^{-1} such vertices w. Overall, there can be at most $\ell\rho^{-1}$ vertices in $IB(v)$. □

Now we have a simple characterization of the anchor that allows for local implementations.

Lemma 4.2.9. *The anchor of v is the smallest vertex (according to $\prec$) in the set $\{s | s \in IB(v)$ and $v \in$ `cluster`$(s)\}$.*

Proof. Let the anchor of v be the vertex u. We first argue that u in the given set. Clearly, $v \in$ `cluster`(u). If $u = v$, then $u = v \in IB(v)$ and we are done. Suppsoe $u \neq v$. Then `cluster`(u) is not a singleton (since it contains v). By Claim 4.2.3, `cluster`(u) is contained in the support of $\widehat{M}^{t_v}\vec{u}$, implying that $v \in \mathrm{supp}(\widehat{M}^{t_v}\vec{u})$. Thus, $u \in IB(v)$ and the anchor u is present in the given set.

It remains to argue that u is the smallest such vertex. Suppose there exists $u' \prec u$ in this set. In `globalPartition`, `cluster`(u') is called before `cluster`(u). At the end of this call, v is partitioned and would have u' as its anchor. Contradiction. □

We are set for the local implementation. For a vertex v, we compute $IB(v)$ and run $\texttt{cluster}(u)$ for all $u \in IB(v)$. By Lemma 4.2.9, we can compute the anchor of v, and by Claim 4.2.6, we can perform a BFS to find all connected vertices with the same anchor.

We begin by a procedure that computes $IB(v)$. Since the truncated diffusion is not symmetric, this requires a little care. We use $N(u)$ to denote the neighborhood of vertex u.

$\texttt{findIB}(v)$

1. Initialize $S = \{v\}$.
2. For every $t = 1, \ldots, \ell$:
   (a) For every $w \in S \cup N(S)$, compute $\widehat{M}^t \vec{w}$. If $v \in \mathrm{supp}(\widehat{M}^t \vec{w})$, add v to S.
3. Return S.

Claim 4.2.10. *The output of* $\texttt{findIB}(v)$ *is* $IB(v)$*. The running time is* $O(d^2\ell^3\rho^{-2})$.

Proof. We prove by induction on t, that after t iterations of the loop, S is the set $\{w \mid \exists t' \in [0,t], v \in \mathrm{supp}(\widehat{M}^{t'}\vec{w})\}$. The base case $t = 0$ holds because S is initialized to $\{v\}$. Now for the induction. Consider some w such that $v \in \mathrm{supp}(\widehat{M}^{t+1}\vec{w})$. This means that $(1 - d(w)/2d)\widehat{p}_{w,t}(v) + (1/2d)\sum_{w' \in N(w)} \widehat{p}_{w',t}(v) \geq \rho$. Since the LHS is an average, for some $w' \in N(w) \cap \{w\}$, $\widehat{p}_{w',t}(v) \geq \rho$. Hence, $v \in \mathrm{supp}(\widehat{M}^t \vec{w'})$, and by induction $w' \in S$ at the beginning of the $(t+1)$th iteration. The inner loop will consider w (as it is either w' or a neighbor of w'), correctly determine that $v \in \mathrm{supp}(\widehat{M}^{t+1}\vec{w})$, and add it to S. By construction, every (new) vertex w added to S has the property that $v \in \mathrm{supp}(\widehat{M}^{t+1}\vec{w})$. This completes the induction and the output property.

For the running time, observe that for all iterations, $S \subseteq IB(v)$. By Claim 4.2.8, $|S| \leq \ell\rho^{-1}$. Hence, $|S \cup N(S)|$ has size $O(d\ell\rho^{-1})$. The computation of each $\widehat{M}^t\vec{w}$ can be done in $O(d\ell\rho^{-1})$ time, since the distribution vector after each step has support size at most ρ^{-1}. The total running time of each iteration is $O(d^2\ell^2\rho^{-2})$. There are at most ℓ iterations, leading to a total running time of $O(d^2\ell^3\rho^{-2})$.

□

We can now describe the local partitioning oracle (modulo the description of $\texttt{findr}$).

$\texttt{findAnchor}(v, \boldsymbol{R})$

1. Run $\texttt{findr}(\boldsymbol{R})$ to get the set $K = \{k_1, k_2, \ldots, k_{\overline{h}}\}$.
2. Run $\texttt{findIB}(v)$ to compute $IB(v)$.
3. Initialize $A = \emptyset$.
4. For every $s \in IB(v)$:
   (a) Using $\boldsymbol{R}$ determine h_s, t_s. Using K, determine k_s.
   (b) Compute $C = \texttt{cluster}(s, t_s, k_s)$.
   (c) If $C \ni v$, then add s to A.
5. Output the smallest vertex according to $\prec$ in A.

`findPartition`$(v, \boldsymbol{R})$

1. Call `findAnchor`$(v, \boldsymbol{R})$ to get the anchor s.
2. Perform BFS from v. For every vertex w encountered, first call `findAnchor`$(w, \boldsymbol{R})$. If the anchor is s, add w to the BFS queue (else, ignore w).
3. Output the set of vertices that entered the BFS queue.

The following claim is a direct consequence of Lemma 4.2.9 and Claim 4.2.10.

Claim 4.2.11. *The procedure* `findAnchor`$(v, \boldsymbol{R})$ *outputs the anchor of* v *and runs in time* $O((d\ell\rho^{-1})^3)$ *plus the running time of* `findr`.

Proof. Observe that `findAnchor`$(v, \boldsymbol{R})$ finds $IB(v)$, computes `cluster`(s) for each $s \in IB(v)$, and outputs the smallest (by $\prec$) s such that $v \in$ `cluster`(s). By Lemma 4.2.9, the output is the anchor of v.

By Claim 4.2.10, the running time of `findIB`(v) is $O(d^2\ell^3\rho^{-2})$. The number of calls to `cluster` is $|IB(v)|$, which is at most $\ell\rho^{-1}$ (Claim 4.2.8). Each call to `cluster` runs in time $O(d\ell\rho^{-2})$, by Claim 4.2.3 and the fact that $k_s \leq \rho^{-1}$). Ignoring the call to `findr`, the total running time is $O(d^2\ell^3\rho^{-3})$. □

Theorem 4.2.12. *The output of* `findPartition`$(v, \boldsymbol{R})$ *is precisely the set in* $\boldsymbol{P}$ *containing* v, *where* $\boldsymbol{P}$ *is the partition output by* `globalPartition`$(\boldsymbol{R})$. *The running time of* `findPartition`$(v, \boldsymbol{R})$ *is* $O((d\ell\rho^{-1})^4)$ *plus the running time of* `findr`.

Proof. By Claim 4.2.11, `findAnchor` correctly outputs the anchor. By Claim 4.2.6, the set S in $\boldsymbol{P}$ containing v is exactly the maximal connected component of vertices sharing the same anchor (as v). The set S in $\boldsymbol{P}$ is generated in `globalPartition`$(\boldsymbol{R})$ by a call to `cluster`, whose output is a set of size at most ρ^{-1}. The total number of calls to `findAnchor` made by `findPartition`$(v, \boldsymbol{R})$ is at most $d\rho^{-1}$, since a call is made to either a vertex in the set S or a neighbor of S. Overall, the total running time is $O((d\ell\rho^{-1})^5)$ plus the running time of `findr`. (Instead of calling `findr` in each call to `findAnchor`, one can simply store its output.) □

4.3 Coordination through the size thresholds: the procedure `findr`

We now come to the heart of our algorithm; coordination through `findr`. This section gives the crucial ingredient in arguing that the partitioning scheme does not cut too many edges. The ordering of vertices (to form clusters) is chosen independent of the graph structure. It is highly likely that, as the partitioning proceeds, newer `cluster`(v) sets overlap heavily with the existing partition. Such clusters may cut many new edges, without clustering enough vertices.

Note that `cluster`(v) is a low conductance cut only in the original graph; it might have high conductance restricted to F (the current free set).

To deal with such "bad" clusters, we need to prove that every so often, `cluster`(v) will successfully partition enough new vertices. Such "good" clusters allow the partitioning scheme to suffer many bad clusters. This argument is finally carried about by a careful charging argument. First, we need to argue that such good clusters exist. The key tool is given by the following theorem, which is proved using spectral graph theoretic methods. We state the theorem as an independent statement.

Theorem 4.3.1. *Let G be a bounded degree graph in a minor-closed family. Let F be an arbitrary set of vertices of size at least βn. There exists a size threshold $k \leq \rho^{-1}$ such that the following holds. For at least $(\beta^2/\log^2 \beta^{-1})n$ vertices $s \in F$, there are at least $(\beta/\log^2 \beta^{-1})\ell$ timesteps $t \leq \ell$ such that: there exists $k' \in [k, 2k]$ such that (i) $L_{s,t,k'} \subseteq \mathrm{supp}(\widehat{M}^t \vec{s})$, (ii) $\Phi(L_{s,t,k'} \cup \{s\}) < \phi$, and (iii) $|L_{s,t,k'} \cap F| \geq \beta^3 k$.*

The proof of this theorem is deferred to §4.6. In this section, we apply this theorem to complete the description of the partition oracle and prove its guarantees.

We discuss the significance of this theorem. The diffusion used to define $L_{s,t,k'}$ occurs in G, but we are promised a low conductance cut with non-trivial intersection with F (since $\phi \ll \beta^3$). Moreover, such cuts are obtained for a non-trivial fraction of timesteps, so we can choice one uar. Given oracle access to membership in F, it is fairly easy to find such a size threshold by random sampling.

The importance of phases: Recall the global partitioning procedure `globalPartition`. We can think of the partitioning process as divided into phases, where the hth phase involves calling `cluster`(v, t_v, k_v) for all vertices v whose phase value is h. Consider the free set at the beginning of a phase h, denoting it F_h. We apply Theorem 4.3.1 to determine the size threshold k_h. Since all k_v values in this phases are precisely k_h, this size threshold "coordinates" all clusters in this phase. As the phase proceeds, the free set shrinks, and the size threshold k_h stops satisfying the properties of Theorem 4.3.1. Roughly speaking, at this point, we start a new phase $h+1$, and recompute the size threshold. The frequency of recomputation is chosen carefully to ensure that the total running time remains poly(ε^{-1}).

We now discuss the randomness involved in selecting phases and why geometric random variables are used. Recall that h_v is independently (for all v) set to be $\min(X, \overline{h})$, where $X \sim Geo(\delta)$. We first introduce some notation regarding phases.

Definition 4.3.2. *The* phase h seeds*, denoted V_h, are the vertices whose phase value is h. Formally, $V_h = \{v \mid h_v = h\}$. We use $V_{<h}$ to denote $\bigcup_{h'<h} V_h$. (We analogously define $V_{\leq h}, V_{\geq h}$.)*

The free set at phase h*, denoted F_h, is the free set F in* `globalPartition`*, just before*

the first phase h vertex is processed. Formally,

$$F_h = V \setminus \bigcup_{v \in V_{<h}} \texttt{cluster}(v)$$

One can think of the V_hs being generated iteratively. Assume that we have fixed the vertices in $V_1, \ldots, V_{h-1}$. All other vertices are in $V_{\geq h}$, implying that $h_v \geq h$ for such vertices. By the properties of the geometric random variables, $\Pr[h_v = h+1 | h_v > h] = \delta$. Thus, we can imagine that V_{h+1} is generated by independently sampling each element in $V_{\geq h}$ with δ probability. We restate this observation as Claim 4.3.4. Claim 4.3.5 is a simple Chernoff bound argument.

Before proceeding, we state some standard Chernoff bounds (Theorem 1.1 of [DP09]).

Theorem 4.3.3. *Let $X_1, X_2, \ldots, X_r$ be independent variables in $[0,1]$. Let $\mu := \mathbf{E}[\sum_i X_i]$.*

- $\Pr[X \geq 3\mu/2] \leq \exp(-\mu/12)$.
- $\Pr[X \leq \mu/2] \leq \exp(-\mu/8)$.
- *For $t \geq 6\mu$, $\Pr[X \geq t] \leq 2^{-t}$.*

Claim 4.3.4. *For all $v \in V$ and $1 < h < \overline{h}$, $\Pr[v \in V_h \mid v \in V_{\geq h}] = \delta$.*

Claim 4.3.5. *Let $h < \overline{h}$. Condition on the randomness used to specify $V_1, V_2, \ldots, V_{h-1}$. Let S be an arbitrary subset of $V_{\geq h}$. With probability at least $1 - 2\exp(-\delta|S|/12)$ over the choice of V_h, $|S \cap V_h| \in [\delta|S|/2, 2\delta|S|]$.*

Proof. For every $s \in S$, let X_s be the indicator random variable for $s \in V_h$. By Claim 4.3.4 and independent phase choices for each vertex, the X_s are independent Bernoullis with δ probability. By the Chernoff lower tail of Theorem 4.3.3, $\Pr[\sum_{s \in S} X_s \leq \delta|S|/2] \leq \exp(-\delta|S|/8)$ and $\Pr[\sum_{s \in S} X_s \geq 2\delta|S|] \leq \exp(\delta|S|/12)$. A union bound completes the proof. □

Claim 4.3.6. *With probability at least $1 - 2^{-\delta n}$, $|V_{\overline{h}}| \leq \delta n$.*

Proof. Recall that $\overline{h}$ is the last phase and $\overline{h} = 2\delta^{-1}\log(\delta^{-1})$. The probability that $X \sim Geo(\delta)$ is at least $2\delta^{-1}\log(\delta^{-1})$ is $(1-\delta)^{2\delta^{-1}\log(\delta^{-1})-1} < \delta/6$. Hence, the probability that any vertex lies in $V_{\overline{h}}$ is at most $\delta/6$ and the expectation of $V_{\overline{h}}$ is at most $\delta n/6$. . By the Chernoff bound of Theorem 4.3.3, $\Pr[|V_{\overline{h}}| \geq \delta n] \leq 2^{-\delta n}$. □

With this preamble, we proceed to the description of `findr` and the main properties of its output.

4.3.1 The procedure `findr`

It is convenient to assume that for all v, h_v and t_v have been chosen. These quantities are chosen independently for each vertex using simple distributions, so we will not carry as

arguments the randomness used to decide these quantities. Recall that the output of `findr` is the set of size thresholds $\{k_1, k_2, \ldots, k_{\overline{h}}\}$. It is convenient to use K_h to denote $\{k_1, k_2, \ldots, k_h\}$. Before describing `findr`, we define a procedure that is a membership oracle for F_h.

`IsFree`(u, h, K_{h-1})

1. If $h = 1$, output YES.
2. Run `findIB`(u) to determine $IB(u)$. Let C be $IB(u) \cap V_{<h}$.
3. Using K_{h-1}, determine k_v for all $v \in C$.
4. For all $v \in C$, compute `cluster`(v, t_v, k_v). If the union contains u, output NO. Else, output YES.

Claim 4.3.7. *Assume that K_{h-1} is provided correctly. Then* `IsFree`(v, h, K_{h-1}) *outputs YES iff $v \in F_h$. The running time is $O((d\ell\rho^{-1})^3)$.*

Proof. If $h = 1$, then all vertices are free (this is the free set before `globalPartition` begins any partitioning). Assume $h > 1$. So $F_h = V \setminus \bigcup_{v \in V_{<h}}$ `cluster`(v).

If $u \notin F_h$, then there exists $v \in V_{<h}$ such that $u \in$ `cluster`(v). By construction `cluster`(v) is contained in $\mathrm{supp}(\widehat{M}^t \vec{v})$ for some $t \leq \ell$. Thus, $v \in IB(u)$ and $h_v < h$. Hence, v will be considered in Step 4 and the union will contain u. The output is NO. For the converse, observe that if the output is NO, then there is a $v \in V_{<h}$ such that $u \in$ `cluster`(v). Hence, $u \notin F_h$.

Now for the running time analysis. The running time of `findIB`(v) is $O(d^2\ell^3\rho^{-2})$ (Claim 4.2.10) and $|C| \leq \ell\rho^{-1}$ (Claim 4.2.8). Each call to `cluster` takes $O(d\ell\rho^{-2})$ (Claim 4.2.3). The total running time is $O((d\ell\rho^{-1})^3)$.

□

We have the necessary tools to define the procedure `findr`. We will need the following definition in our description and analysis of `findr`.

Definition 4.3.8. *Assume $F_h \geq \beta n$. A vertex $s \in V_{\geq h}$ is called* (h, k)-viable *if $C :=$ `cluster`(s, t_s, k) is not a singleton and $|C \cap F_h| \geq \beta^3 k$. (If $F_h < \beta n$, no vertex is (h, k)-viable.)*

Let us motivate this definition. When $C :=$ `cluster`(s, t_s, k) is not a singleton, it is a low conductance cut of $\Theta(k)$ vertices. The vertex s is (h, k)-viable if C contains a non-trivial fraction of free vertices available in the hth phase. The viable vertices are those from which clustering will make significant "progress" in the hth phase. For each h, the procedure `findr` searches for values of k that lead to many (h, k)-viable vertices. In the next section, we prove that having sufficiently many clusters come from viable vertices ensures the cut bound of Def 4.0.1.

`findr`($\boldsymbol{R}$)

1. For $h = 1$ to $\overline{h}$:
   (a) Sample β^{-10} uar vertices independently. Let S_h be the multiset of sampled vertices that are in phase $\geq h$.
   (b) If $|S_h| \leq \beta^{-9}/2$, set $k_h = 0$ and continue for loop. Else, reset S_h to the multiset of the first β^{-8} vertices sampled.
   (c) For $k \in [\rho^{-1}]$ and for every $s \in S_h$:
      i. Compute $C := \texttt{cluster}(s, t_s, k)$.
      ii. For all $u \in C$, call $\texttt{IsFree}(u, h, K_{h-1})$ to determine if $u \in F_{h-1}$.
      iii. If C is not a singleton and $|C \cap F_{h-1}| \geq \beta^3 k$, mark s as being (h,k)-viable.
   (d) If there exists some k such that there are at least $12\beta^4|S_h|$ (h,k)-viable vertices, assign an arbitrary such k as k_h. Else, assign $k_h := 0$.
2. Output $K_{\overline{h}} = \{k_1, k_2, \ldots, k_{\overline{h}}\}$.

Claim 4.3.9. *The running time of* `findr` *is* $O((d\ell\delta^{-1}\rho^{-1})^5)$.

Proof. There are $\overline{h} = 2\delta^{-1}\log(\delta^{-1})$ iterations. We compute the running time of each iteration. There are at most $\rho^{-1}\beta^{-8}$ calls to `cluster`, each of which takes $O(d\ell\rho^{-2})$ time by Claim 4.2.3. For each call to `cluster`, there are at most ρ^{-1} calls to `IsFree`. Each call to `IsFree` takes $O((d\ell\rho^{-1})^3)$ time (Claim 4.3.7). The running time of each iteration is $O(\beta^{-10} + d\ell\rho^{-3}\beta^{-8} + d^3\ell^3\rho^{-5}\beta^{-8})$. By the parameter settings, since $\ell^2 \geq \varepsilon^{2\cdot 30} \geq (\varepsilon/10)^{-8} = \beta^{-8}$, the running time of each iteration $O((d\ell\rho^{-1})^5)$. The total running time is $O((d\ell\delta^{-1}\rho^{-1})^5)$. □

The following theorem gives the main guarantee of `findr`. The proof is a fairly straightforward Chernoff bound on top of an application of Theorem 4.3.1. Quite simply, the proof just says the following. Theorem 4.3.1 shows the existence of (h,k) pairs for which many vertices are viable. The `findr` procedure finds such pairs by random sampling.

Theorem 4.3.10. *The following property of the values* $K_{\overline{h}}$ *of* `findr`($\boldsymbol{R}$) *and the preprocessing choices holds with probability at least* $1 - \exp(-1/\varepsilon)$ *over all the randomness in* $\boldsymbol{R}$. *For all* $h \leq \overline{h}$, *if* $|F_h| \geq \beta n$, *at least* $\beta^5\delta n$ *vertices in* V_h *are* (h, k_h)*-viable.*

Proof. The proof has two parts. In the first part, we argue that whp, if $|F_h| \geq \beta n$, then a non-zero k_h is output. This part is an application of Theorem 4.3.1. In the second part, we prove that (whp), if a non-zero k_h is output, then it satisfies the desired properties. This part is proven using a simple Chernoff bound argument.

Fix an h. Condition on any choice of $V_1, V_2, \ldots, V_{h-1}$ such that $|F_h| \geq \beta n$. Note that $V_{\geq h} \supseteq F_h$, since all vertices in $V_{<h}$ are necessarily clustered by the hth phase. (Recall that `cluster`(v) always contains v.) Hence, $|V_{\geq h}| \geq \beta n$. There will be numerous low probability

"bad" events that we need to track. We will describe these bad events, and refer to their probabilities as "Error 1", "Error 2", etc.

Error 1, $\exp(-\beta^{-8})$. The probability that a uar vertex is in $V_{\geq h}$ is at least β, and the expected size of S_h is at least $\beta \times \beta^{-10} = \beta^{-9}$. By the Chernoff bound of Theorem 4.3.3, $\Pr[|S_h| \leq \beta^{-9}/2] \leq \exp(-\beta^{-9}/12) \leq \exp(-\beta^{-8})$. Thus, with probability at least $1 - \exp(-\beta^{-8})$, Step 1c is reached and S_h is a multiset of iid uar β^{-8} elements in $V_{\geq h}$.

Let us assume that S_h is such a multiset, and prove that a non-zero k_h is output whp. We bring out the main tool, Theorem 4.3.1. Since $|F_h| \geq \beta n$, there exists a size threshold $k \leq \rho$ such that the following holds. For at least $(\beta^2/\log^2 \beta^{-1})n$ vertices $s \in F_h$, there are at least $(\beta/\log^2 \beta^{-1})\ell$ timesteps t such that: there exists $k' \in [k, 2k]$ such that (i) $L_{s,t,k'} \subseteq \mathrm{supp}(\widehat{M}^t \vec{s})$, (ii) $\Phi(L_{s,t,k'} \cup \{s\}) < \phi$, and (iii) $|L_{s,t,k'} \cap F| \geq \beta^3 k$. For any such (s, t, k) triple, consider a call to `cluster`(s, t, k). Observe that the call will output the largest level set of size in $[k, 2k]$ satisfying (i) and (ii). Hence, it will output (non-singleton) $L_{s,t,k''}$ such that $k' \leq k'' \leq 2k$ and (i) and (ii) hold. Note that $L_{s,t,k''} \supseteq L_{s,t,k'}$, so the third item will also hold. Thus, *if t_s is set to one of these $(\beta/\log^2 \beta^{-1})\ell$ timesteps t*, then s will be (h, k)-viable.

Error 2, $\exp(-\beta^{-1})$. Let us fix a size threshold k promised by Theorem 4.3.1. The probability that a uar element on $V_{\geq h}$ is marked as (k, h)-viable is at least the product of probability of choosing an appropriate s with the probability that t_s is chosen appropriately. Thus, the probability of find an (h, k)-viable vertex is at least $(\beta^2/\log^2 \beta^{-1}) \times (\beta/\log^2 \beta^{-1}) = \beta^3/\log^4 \beta^{-1}$. This probability is independent for all vertices in $V_{\geq h}$. By the Chernoff bound in Theorem 4.3.3, with probability at least $1 - \exp(-\beta^4|S_h|/12)$, at least

$$\beta^3|S_h|/2\log^4 \beta^{-1} \geq 12\beta^4|S_h|$$

(h, k)-viable vertices are discovered in `findr`. In this case, in Step 1d, k_h is set to a non-zero value. The probability of this event happening is at least $1 - \exp(-\beta^{-8}) - \exp(-\beta^4|S_h|/8) \geq 1 - \exp(\beta^{-1})$. (Recall that whp S_h is a multiset of iid uar β^{-8} vertices. In the union bound above, the first "bad event" is S_h *not* having β^{-8} vertices and the second "bad event" is discovering too few viable vertices.) We have concluded that whp, if $|F_h| \geq \beta n$, then k_h is non-zero.

We move to the second part of the proof, which asserts that (with high probability), an output non-zero k_h has the desired properties. Condition on any choice of the preprocessing. Note that the randomness is only over the choice of S_h. Fix any $k \leq \rho^{-1}$. Suppose that the number of (h, k)-viable vertices in $V_{\geq h}$ is at most $2\beta^5 n$. Then, the expected number of such vertices in S_h is at most $2\beta^5 n/|V_{\geq h}| \times |S_h| \leq 2\beta^4|S_h|$. (We use the lower bound $|V_{\geq h}| \geq |F_h| \geq \beta n$.)

Error 3, $2^{-12\beta^{-4}}$. Let X_k denote the random variable of the number of (h, k)-viable vertices in S_h. Since X_k is distributed as a binomial, by the Chernoff bound of Theorem 4.3.3, $\Pr[X_k > 12\beta^4|S_h|] \leq 2^{-12\beta^4|S_h|}$. Note than when $X_k < 12\beta^4|S_h|$, then k_h cannot be k. All in all, for any h, any choice of the t_vs, and any choice of k, if Step 1c is reached and the number of

(h,k)-viable vertices in $V_{\geq h}$ is at most $2\beta^5 n$, then $k_h \neq k$ with probability at least $1 - 2^{-12\beta^{-4}}$. Taking the contrapositive, if $k_h \neq 0$ (Step 1c must have been reached), then the number of (h,k_h)-viable vertices in $V_{\geq h}$ is at least $2\beta^5 n$.

Error 4, $2\exp(-\delta\beta^5 n/12)$. Suppose the number of (h,k_h)-viable vertices in $V_{\geq h}$ is at least $2\beta^5 n$. By Claim 4.3.5 applied on the set of (h,k_h)-viable vertices in $V_{\geq h}$, with probability at least $1 - 2\exp(-\delta\beta^5 n/12)$, the number of such viable vertices in V_h is at least $\delta\beta^5 n$.

We take a union bound over the $2\delta^{-1}\log(\delta^{-1})$ values of h, the ρ^{-1} values of k, and all errors encountered thus far. The total error probability is at most $2\delta^{-1}\log(\delta^{-1}) \cdot \rho^{-1}(\exp(-\beta^{-8}) + \exp(\beta^{-1}) + 2^{-12\beta^{-4}} + 2\exp(-\delta\beta^5 n/12))$. Note that $2\delta^{-1}\log(\delta^{-1}), \beta, \rho^{-1}$ are $\text{poly}(\varepsilon^{-1})$, and thus the total error probability is at most $\exp(-\varepsilon^{-1})$. With the remaining probability, the following holds. For all phases h, if $|F_h| \geq \beta n$, a non-zero k_h is output. If a non-zero k_h is output, the number of (h,k_h)-viable vertices in V_h is at least $\delta\beta^5 n$. □

4.4 Proving the cut bound: the amortization argument

We come to the final piece of proving the guarantees of Theorem 4.0.2. We need to prove that the number of edges cut by the partition of `globalPartition` is at most εnd. This requires an amortization argument explained below. For the sake of exposition, we will ignore constant factors in this high-level description. One of the important takeaways is how various parameters are chosen to prove the cut bound.

Consider phase h where $|F_h| \geq \beta n$. Let us upper bound the number of edges cut by the clustering done on this phase. Roughly speaking, $|V_h| = \delta n$, so there are δn clusters created in this phase. Each cluster in this phase has at most $2k_h$ vertices. The number of edges cut by each such cluster is at most $2\phi k_h d$ (since `cluster` outputs a low conductance cut; ignore singleton outputs). So the total number of edges cut is at most $2\phi\delta k_h nd$.

Let us now lower bound the number of new vertices that are partitioned in phase h; this is the set $F_{h+1} \setminus F_h$. For each (h,k_h)-viable v in V_h, `cluster`(v) contains at least $\beta^3 k_h$ vertices in F_h. These will be newly partitioned vertices. Here comes the primary difficulty: the clusters for the different such v might not be disjoint. We need to lower bound the union of the clustered vertices in F_h. An alternate description of the challenge is as follows. We are only guaranteed that clusters from viable vertices v contains many vertices in F_h, the free set at the *beginning* of phase h. What we really need is for the cluster from v to contain many free vertices *at the time* that v is processed. Phases were introduced to solve this problem. By reducing δ, we can limit the size of V_h, thereby limiting the intersection between the clusters produced in this phase.

We now explain the math behind this argument. Consider some $w \in F_h$ and let c_w be the number of vertices in $V_{\geq h}$ that cluster v (call these seeds). Thus, $c_w = |\{s \mid s \in V_{\geq h}, v \in$

$\texttt{cluster}(s)\}$. The vertex w is clustered in phase h iff one of these c_w seeds is selected in V_h. By Claim 4.3.4, each such seed is independently selected in V_h with probability δ. The probability that w is clustered in this phases is precisely $1-(1-\delta)^{c_w}$. Crucially, $c_w \leq |IB(w)| \leq \ell\rho^{-1}$. We chose $\delta \ll \ell\rho^{-1}$, so $1-(1-\delta)^{c_w} \approx \delta c_w$.

Thus, the expected number of newly clustered vertices is at least $\sum_{w\in F_h} \delta c_w$. By rearranging summations, $\sum_{w\in F_h} c_w = \sum_{v\in V_{\geq h}} |\texttt{cluster}(v)\cap F_h|$. For every (h,k_h)-viable vertex v in $V_{\geq h}$, $|\texttt{cluster}(v)\cap F_h| \geq \beta^3 k_h$. The arguments in the proof of Theorem 4.3.10 shows that there are $\beta^5 n$ such vertices in $V_{\geq h}$ whp. Hence, we can lower bound (in expectation) the new number of newly clustered vertices as follows:

$$\sum_{w\in F_h} \delta c_w \geq \delta\cdot(\beta^5 n)\cdot(\beta^3 k_h) = \delta\beta^8 k_h n$$

We upper bounded the number of edges cut by $2\phi\delta k_h nd$. The ratio of edges cut to vertices clustered is $8\phi\beta^{-8}d$. The parameters are set to ensure that $8\phi\beta^{-8} \ll \varepsilon$, so the total number of edges cut is εnd.

The formal analysis requires some care to deal with conditional probabilities and dependencies between various phases. Also, Theorem 4.3.10 talks about V_h and not $V_{\geq h}$, which necessitates some changes. But the essence of the argument is the same.

Our main theorem is a cut bound for `globalPartition`.

Theorem 4.4.1. *The expected number of edges cut by the partitioning of* $\texttt{globalPartition}(\boldsymbol{R})$ *is at most* εnd.

We will break up the proof into two technical claims. Somewhat abusing notation, we say a vertex in $V_{\geq h}$ is h-viable if it is (h,k_h)-viable.

Claim 4.4.2.

$$\mathbf{E}[\#\text{ edges cut by }\texttt{globalPartition}(\boldsymbol{R})] \leq 32\phi\beta^{-8}d^2\Big(\sum_{h<\overline{h}}\mathbf{E}[\sum_{v\in V_h}|\texttt{cluster}(v)\cap F_h|)]\Big) + 2\beta nd$$

Proof. The proof goes phase by phase. We call a phase significant if $|F_h| \geq \beta n$. Edges cut in a significant phase are also called significant. Observe that the total number of edges cut is at most the number of significant edges cut plus βnd. (This contributes to the extra additive term in the claim statement.) Below, we will bound the total number of significant edges cut.

By Claim 4.3.6, with probability at least $1-2^{-\delta n}$, $|V_{\overline{h}}| \leq \delta n$. Note that $|F_h| \leq V_{\geq\overline{h}} = |V_{\overline{h}}|$. (The equality is because this is the last phase.) Since $\delta n < \beta n$, the expected number of significant edges cut in the last phase is at most $2^{-\delta n}nd < 1$.

Now assume that $h < \overline{h}$. Consider the edges cut in the hth phase. Consider any choice of $V_1, V_2, \ldots, V_{h-1}$ and $k_1, k_2, \ldots, k_h$. If $|F_h| < \beta n$, no significant edges are cut. Let us assume that $|F_h| \geq \beta n$. Each set $\texttt{cluster}(v)$ output in this phase is either a singleton or a set of size

at most $2k_h$ and conductance at most ϕ. In either case, the number of edges cut by removing $\texttt{cluster}(v) \cap F$ (in `globalPartition`) is at most $2\phi k_h d + d$. Note that $2\phi k_h d \geq 1$ (otherwise, by the connectedness of G, there can never be a set of size at most $2k_h$ of conductance $\leq \phi$). Hence, the number of significant edges cut by a single cluster is at most $2\phi k_h(d+d^2) \leq 4\phi k_h d^2$.

Note that $|V_{\geq h}| \geq |F_h| \geq \beta n$ and $|V_{\geq h}|$ is obviously at most n. By Claim 4.3.5 with $S = V_{\geq h}$, with probability at least $1 - 2\exp(-\delta\beta n/12)$ over the choice of V_h, $|V_h| \leq 2\delta n$. Hence, the total number of significant edges cut is at most $4\phi k_h d^2 \times 2\delta n = 8\phi\delta k_h d^2 n$.

By Theorem 4.3.10, with probability at least $1 - \exp(\varepsilon^{-1})$, if $|F_h| \geq \beta n$, at least $\beta^5\delta n$ vertices in V_h are h-viable. Call this event $\mathcal{E}$. For every h-viable vertex in V_h, $|\texttt{cluster}(v) \cap F_h| \geq \beta^3 k_h$. For convenience, let $X_h := \sum_{v \in V_h} |\texttt{cluster}(v) \cap F_h|)$. Conditioned on $\mathcal{E}$, $X_h \geq \beta^8(\delta k_h n)$. Recall that with probability at least $1 - 2\exp(-\delta\beta n/12)$, the number of significant edges cut in this phase is at most $8\phi d^2(\delta k_h n)$. If $\mathcal{E}$ occurs, we can apply the bound $\beta^{-8}X_h \geq \delta k_h n$ and upper bound the number of significant edges cut in this phase by $8\phi\beta^{-8}d^2 X_h$,

Thus, with probability at least $1 - \exp(\varepsilon^{-1}) - 2\exp(-\delta\beta n/12)$, the number of significant edges cut in phase h is at most $(8\phi\beta^{-8}d^2)X_h$. In other words, there is an event $\mathcal{F}_h$ conditioned on which the above bound happens, and $\Pr[\mathcal{F}_h] \geq 1 - \exp(\varepsilon^{-1}) - 2\exp(-\delta\beta n/12)$. In the calculation below, we break into conditional expectations and use the fact that $\delta = \text{poly}(\varepsilon)$, $\beta = \Theta(\varepsilon)$, and that the number of phases is at most $2\delta^{-1}\log(\delta^{-1})$. We also use the fact that X_h is non-negative.

$$\sum_h \mathbf{E}[\#\text{ significant edges cut in phase } h] \leq \sum_h (\Pr[\mathcal{F}]\mathbf{E}[X_h|\mathcal{F}] + \Pr[\overline{\mathcal{F}}]nd) \tag{4.1}$$

$$\leq \sum_h \mathbf{E}[X_h] + 2\delta^{-1}\log(\delta^{-1})(\exp(\varepsilon^{-1}) + 2\exp(-\delta\beta n/12))nd \leq \sum_h \mathbf{E}[X_h] + \beta nd/2 \tag{4.2}$$

To this bound, we add the expected number of edges cut in the last phase (at most 1) and the number of non-significant edges cut (at most βn). This completes the proof. □

Claim 4.4.3.

$$\sum_{h<\overline{h}} \mathbf{E}[\sum_{v \in V_h} |\texttt{cluster}(v) \cap F_h|)] \leq 4n$$

Proof. We will apply the following charging argument. When a vertex v is processed in `globalPartition`($\boldsymbol{R}$), we will add one unit of charge to every vertex in $\texttt{cluster}(v) \cap F_h$. Note that the total amount of charge is exactly the quantity we wish to bound. Crucially, note that any vertex w receives charge in at most one phase; the phase where it leaves the free set.

We will prove that the expected charge that any vertex receives is at most 4 units, which will prove the claim. Fix a vertex w. Let χ be the random variable denoting the charge that w receives, and $\mathcal{E}_h$ be the event that w receives charge in phase h. Since w receives charge

in exactly one phase, $\mathbf{E}[\chi] = \sum_h \mathbf{E}[\chi|\mathcal{E}_h]\Pr[\mathcal{E}_h]$. We will prove that, for all h, $\mathbf{E}[\chi|\mathcal{E}_h] \leq 4$, which implies that $\mathbf{E}[\chi] \leq 4$ as desired.

To analyze $\mathbf{E}[\chi|\mathcal{E}_h]$, first condition on a setting of $V_1, V_2, \ldots, V_{h-1}$ (such that $w \in F_h$) and all other preprocessing for all vertices. We refer to this setting as the event $\mathcal{C}$. The randomness for specifying V_h has not been set. The event $\mathcal{E}_h$ occurs if there is a $v \in V_h$ such that $w \in \texttt{cluster}(v)$. The charge χ is the number of vertices $v \in V_h$ such that $w \in \texttt{cluster}(v)$. Let c be the number of such vertices in $V_{\geq h}$. Note that $v \in IB(w)$, and by Claim 4.2.8, $c \leq \ell\rho^{-1}$.

By Claim 4.3.4, every vertex in $V_{\geq h}$ is in V_h with probability δ. Hence, $\Pr[\mathcal{E}_h|\mathcal{C}] = 1-(1-\delta)^c$. Note that $\delta c \leq \delta\ell\rho^{-1} = (d^{-70+6+60}\varepsilon^{3100} \cdot \varepsilon^{30} \cdot \varepsilon^{-3000} < 1/2$. Hence $(1-\delta)^c \leq 1-\delta c+(\delta c)^2 \leq 1-\delta c/2$ and $\Pr[\mathcal{E}_h|\mathcal{C}] \geq \delta c/2$. Note that $\mathbf{E}[\chi|\mathcal{C}] = \sum_{b>0}\binom{c}{b}\delta^b \leq \sum_{b>0}(\delta c)^b \leq 2\delta c$. Observe that $\mathbf{E}[(\chi|\mathcal{E}_h)|\mathcal{C}] \leq (2\delta c)/(\delta c/2) = 4$.

Note that the event $\mathcal{E}_h$ can be partitioned according to the different $\mathcal{C}$ events. Hence $\mathbf{E}[\chi|\mathcal{E}_h] = \sum_{\mathcal{C}} \mathbf{E}[(\chi|\mathcal{E}_h)|\mathcal{C}]\Pr[\mathcal{C}] \leq 4$. Thus, the proof is completed. □

Theorem 4.4.1 follows by a direct application of these claims and plugging in the parameter values.

Proof. (of Theorem 4.4.1) By Claim 4.4.2 and Claim 4.4.3, the expected number of edges cut by $\texttt{globalPartition}(\boldsymbol{R})$ is at most $128\phi\beta^{-8}d \cdot nd + 2\beta nd$. Plugging in the parameters $\phi = d^{-1}\varepsilon^{10}$, $\beta = \varepsilon/10$, and noting that ε is sufficiently small, the expectation is at most εnd. □

We can now wrap up the proof of Theorem 4.0.2, showing the existence of $(\varepsilon, \text{poly}(d\varepsilon^{-1}))$-partition oracles for minor-closed families.

Proof. (of Theorem 4.0.2) The procedure for the partition oracle is $\texttt{findPartition}(v, \boldsymbol{R})$. Let us prove each property of Def 4.0.1.

Consistency: By Theorem 4.2.12, the partition created by calls to $\texttt{findPartition}(v, \boldsymbol{R})$ is precisely the same as the partition created by $\texttt{globalPartition}(\boldsymbol{R})$.

Cut bound: By Theorem 4.4.1, the expected number of edges cut is at most εnd.

Running time: The running time of $\texttt{findPartition}(v, \boldsymbol{R})$ is $O((d\ell\rho^{-1})^5)$ plus the running time of `findr`. The running time of `findr` is $O((d\ell\delta^{-1}\rho^{-1})^5)$, by Claim 4.3.9. By the parameter settings, $\ell, \delta^{-1}, \rho^{-1}$ are all $\text{poly}(d\varepsilon^{-1})$. Hence, the total running time of $\texttt{findPartition}(v, \boldsymbol{R})$ is also $\text{poly}(d\varepsilon^{-1})$. □

4.5 Diffusion Behavior on Minor-Free Families

In this section, we state and prove the main theorem about diffusions on minor-free graph classes. This is the (only) part of the paper where the property minor-freeness makes an appearance. Theorem 4.5.1 is used in the proof of §4.6. For convenience, we recall the parameters involved.

- $\rho = d^{-60}\varepsilon^{3000}$: Minimum probability for truncation.
- $\ell = d^6\varepsilon^{-30}$: Maximum random walk length.
- $\beta = \varepsilon/10$: Unclustered fraction cutoff.
- $\delta = d^{-70}\varepsilon^{3100}$: Phase probability.
- $\alpha = \frac{\varepsilon^{4/3}}{300{,}000}$: Heavy bucket parameter.
- $\phi = \varepsilon^{10}$: Conductance parameter.

Theorem 4.5.1. *Let G be a bounded degree graph in minor-closed family. Let F be an arbitrary subset of at least βn vertices. There are at least $\beta^2 n/8$ vertices $s \in F$ such that: for at least $\beta\ell/8$ timesteps $t \in [\ell]$, $\widehat{M}^t\vec{s}(F) \geq \beta/16$.*

We note that this theorem holds for all graphs, if we replace the truncated walk $\widehat{M}$ by the standard random walk M. The main insight is that, for G in a minor-closed family, "polynomial" truncation of the walk distribution does not significantly affect the behavior.

The main property of bounded degree minor-free graphs we require is hyperfiniteness, as expressed by Proposition 4.1 of [AST90] (also used as Lemma 3.3 of [KSS19b]).

Theorem 4.5.2. *There is an absolute constant γ such that the following holds. Let H be a graph on r vertices. Suppose G is an H-minor-free graph. Then, for all $b \in \mathbb{N}$, there exists a set of at most $\gamma r^{3/2} n/\sqrt{b}$ vertices whose removal leaves G with all connected components of size at most k.*

The key stepping stone to proving Theorem 4.5.1 is Lemma 4.5.4, which shows that truncation does not affect walk distributions from many vertices. Let us first state a simple fact on l_1-norms.

Fact 4.5.3. *Let $\vec{x}$ and $\vec{y}$ be vectors with non-negative entries, such that for all coordinates i, $\vec{x}(i) \geq \vec{y}(i)$. Then $\|\vec{x} - \vec{y}\|_1 = \|\vec{x}\|_1 - \|\vec{y}\|_1$.*

Proof. $\|\vec{x} - \vec{y}\|_1 \geq \sum_i |\vec{x}(i) - \vec{y}(i)| = \sum_i (\vec{x}(i) - \vec{y}(i)) = \|\vec{x}\|_1 - \|\vec{y}\|_1$. □

This fact bears relevance for us, since truncations of walk distribution vectors only reduce coordinates.

Lemma 4.5.4. *For at least $(1 - \rho^{1/8})n$ vertices v, the following holds. For every $t \leq \ell$, $\|M^t\vec{v} - \widehat{M}^t\vec{v}\|_1 \leq \ell\rho^{1/9}$.*

Proof. Let H be an arbitrary forbidden minor for the minor-closed family of interest. We first apply Theorem 4.5.2 with $k = \lceil 1/\sqrt{\rho} \rceil$. There exists a set C of at most $\gamma r^{3/2}\rho^{1/4} dn$ edges who removal leads to connected components of size at most $\lceil 1/\sqrt{\rho} \rceil \leq 2/\sqrt{\rho}$. For convenience, set the constant $\gamma' := \gamma r^{3/2}$. We will need the following claim.

Claim 4.5.5. *For at least $(1-\rho^{1/8})n$ vertices v, the probability that an ℓ-length random walk encounters an edge of R is at most $\gamma'\ell\rho^{1/8}$.*

Proof. The proof is a Markov bound argument. Suppose not; so there exist strictly more than $\rho^{1/8}n$ vertices v such that an ℓ-length random walk encounters an edge of C with at least $\gamma'\ell\rho^{1/8}$ probability. Consider an ℓ-length random walk that starts from the uniform (also stationary) distribution. The above assumption implies that the expected number of C edges encountered is $> \rho^{1/8}\cdot\gamma'\ell\rho^{1/8} = \gamma'\ell\rho^{1/4}$. On the other hand, since the walk remains in the stationary distribution, for all $t \leq \ell$, the probability of encountering an edge in C at the tth step is precisely $|C|/2dn$. (Recall that the lazy random walk has $1/2dn$ of taking any edge.) By linearity of expectation, the expected number of C edges encountered is $\ell|C|/2dn$. By the bound of Theorem 4.5.2, $\ell|C|/2dn \leq \gamma'\ell\rho^{1/4}$ contradicting the bound obtained from the assumption. □

Consider such a vertex v, as promised by the previous paragraph. Let S be the connected component over vertices that contains v, after removing the edge cut C. Let q_t be the probability that the walk from v leaves S at the tth step; by the property of the previous parameter, $\sum_{t\leq\ell} p_t \leq \gamma'\ell\rho^{1/8}$. Let M_S be the transition matrix of the random walk M *restricted to* S. Note that M_S is not necessarily stochastic. We will use the truncated walk $\widehat{M}_S$. Observe that $\|\widehat{M}^t\vec{v}\|_1 \geq \|\widehat{M}_S^t\vec{v}\|_1$.

Since all coordinates of $\widehat{M}^t\vec{v}$ are at most those of $M^t\vec{v}$, by Fact 4.5.3, $\|M^t\vec{v} - \widehat{M}^t\vec{v}\|_1 = \|M^t\vec{v}\|_1 - \|\widehat{M}^t\vec{v}\|_1$. Since $\|M^t\vec{v}\|_1 = 1 = \|\vec{v}\|_1$ and $\|\widehat{M}^t\vec{v}\|_1 \geq \|\widehat{M}_S^t\vec{v}\|_1$, we can upper bound as follows by a telescoping sum.

$$\begin{aligned}\|M^t\vec{v} - \widehat{M}^t\vec{v}\|_1 &\leq \sum_{l=1}^{t}\left(\|\widehat{M}_S^{l-1}\vec{v}\|_1 - \|\widehat{M}_S^{l}\vec{v}\|_1\right) &(4.3)\\ &= \sum_{l=1}^{t}\left(\|\widehat{M}_S^{l-1}\vec{v}\|_1 - \|M_S\widehat{M}_S^{l-1}\vec{v}\|_1 + \|M_S\widehat{M}_S^{l-1}\vec{v}\|_1 - \|\widehat{M}_S^{l}\vec{v}\|_1\right) &(4.4)\end{aligned}$$

The quantity $\|\widehat{M}_S^{l-1}\vec{v}\|_1 - \|M_S\widehat{M}_S^{l-1}\vec{v}\|_1$ is exactly the probability that a single step (according to M) from $\widehat{M}_S^{l-1}\vec{v}$ leaves S. Since all coordinates in $\widehat{M}_S^{l-1}\vec{v}$ are at most those of $M^{l-1}\vec{v}$, this probability is at most q_l. The quantity $\|M_S\widehat{M}_S^{l-1}\vec{v}\|_1 - \|\widehat{M}_S^{l}\vec{v}\|_1$ is the probability mass lost by truncation of $M_S\widehat{M}_S^{l-1}\vec{v}$. We apply the trivial bound $\rho|S|$. This is where the hyperfiniteness plays a role; since $|S| \leq 2/\sqrt{\rho}$, $\|\widehat{M}_S\vec{x}_{l-1} - M_S\vec{x}_{l-1}\|_1 \leq \rho\cdot/2\sqrt{\rho} = 2\sqrt{\rho}$.

We sum all the these bounds over $l \leq t$, and plug into (4.4). We bound $\|M^t\vec{v} - \widehat{M}^t\vec{v}\|_1 \leq \sum_{l\leq t} p_l + 2t\sqrt{\rho}$. By the properties of v, this is at most $\gamma'\ell\rho^{1/8} + 2\ell\sqrt{\rho} \leq \ell\rho^{1/9}$ (for sufficiently small ρ). □

We are now ready to prove Theorem 4.5.1. We will need the following simple "reverse Markov" inequality for bounded random variables.

Fact 4.5.6. *Let X be a random variable taking values in $[0,1]$ such that $\mathbf{E}[X] \geq \delta$. Then $\Pr[X \geq \delta/2] \geq \delta/2$.*

Proof. Let p be the probability that $\Pr[X \geq \delta/2]$.

$$\begin{aligned}\delta \leq \mathbf{E}[X] &= \Pr[X \geq \delta/2]\mathbf{E}[X|X \geq \delta/2] + \Pr[X < \delta/2]\mathbf{E}[X|X < \delta/2] \\ &\leq p + (1-p)(\delta/2) \leq p + \delta/2\end{aligned}$$

□

Proof. (of Theorem 4.5.1) Define $\theta_{s,t}$ as follows. For $s \in F$ and $t \in [\ell]$: if t is odd, $\theta_{s,t} = 0$. If t is even, then $\theta_{s,t}$ is the probability that the t-length random walk starting from s ends in F.

Let us pick a uar source vertex in $s \in F$, pick a uar length $t \in [\ell]$. We use the fact that M is a symmetric matrix. We use $\mathbf{1}_F$ to denote the all 1s vector on F.

$$\mathbf{E}_{s,t}[\theta_{s,t}] = \mathbf{1}_F^T \sum_{i=1}^{\ell/2} (M^{2i}/\ell)(\mathbf{1}_F/|F|) = (\ell|F|)^{-1} \sum_{i \leq \ell/2} \mathbf{1}_F^T M^{2i} \mathbf{1}_F = (\ell|F|)^{-1} \sum_{i \leq \ell/2} \|M^i \mathbf{1}_F\|_2^2 \quad (4.5)$$

Note that $\|M^i \mathbf{1}_F\|_1 = |F|$, so by Jensen's inequality, $\|M^i \mathbf{1}_F\|_2^2 \geq |F|^2/n$. Plugging in (4.5), $\mathbf{E}_{s,t}[\theta_{s,t}] \geq \ell^{-1} \times (\ell/2)|F|/n \geq \beta/2$. For any s, $\mathbf{E}_t[\theta_{s,t}] \leq 1$. By Fact 4.5.6, there are at least $\beta|F|/4$ vertices $s \in F$ such that $\mathbf{E}_t[\theta_{s,t}] \geq \beta/4$. Again applying Fact 4.5.6, for at least $\beta|F|/4$ vertices $s \in F$, there are at least $\beta\ell/8$ timesteps $t \in [\ell]$ such that $\theta_{s,t} \geq \beta/8$, implying that $M^t \vec{s}(F) \geq \beta/8$.

By Lemma 4.5.4, there are at least $(1-\rho^{1/8})n$ vertices s such that for all $t \leq \ell$, $\|M^t \vec{s} - \widehat{M}^t \vec{s}\|_1 \leq \ell\rho^{1/9} = d^{6-60/9}\varepsilon^{-30+3000/9} \leq \beta/16$. By the parameters settings, $\rho^{1/8} < \varepsilon^{3000/8} \leq \beta|F|/8$. Invoking the bound from the previous paragraph, there are at least $\beta|F|/8$ satisfying the property of Lemma 4.5.4 and the condition at the end of the previous paragraph. For all such vertices s, for all $t \leq \ell$, $\widehat{M}^t \vec{s}(F) \geq M^t \vec{s}(F) - \beta/16$. Thus, for all such s, there are at least $\beta\ell/8$ timesteps $t \in [\ell]$ such that $\widehat{M}^t \vec{s}(F) \geq \beta/16$.

□

4.6 The proof of Theorem 4.3.10: local partitioning within F

We repeat the parameter values for convenience.

- $\rho = d^{-60}\varepsilon^{3000}$: Minimum probability for truncation.
- $\ell = d^6\varepsilon^{-30}$: Maximum random walk length.
- $\beta = \varepsilon/10$: Unclustered fraction cutoff.

- $\delta = d^{-70}\varepsilon^{3100}$: Phase probability.
- $\alpha = \frac{\varepsilon^{4/3}}{300{,}000}$: Heavy bucket parameter.
- $\phi = \varepsilon^{10}$: Conductance parameter.

Recall that Theorem 4.3.1 shows that there are many $s \in F$ from which (level sets of) diffusions in G discover low conductances cuts in F. We use the Lovász-Simonovits curve to represent the truncated diffusion vector, and keep track of the vertices of F wrt to the curve. This is done via a careful adaptation of Lovász-Simonovits method, as presented in Lemma 4.6.4.

The main technical tool which we will use in our analysis is the Lovász-Simonovits method, defined in [LS90b], whose use for clustering was pioneered by [ST12].

Definition 4.6.1. *For a non-negative vector* $\mathbf{p}$ *over* V*, the function* $I : \mathbb{R}^n \times [n] \to [0,1]$ *is defined as*

$$I(\mathbf{p}, x) = \max_{\substack{\mathbf{w} \in [0,1]^n \\ \sum \mathbf{w}(u) = x}} \sum_{u \in V} \mathbf{p}(u)\mathbf{w}(u)$$

This is equivalent to summing over the x *heaviest elements of* $\mathbf{p}$ *when* x *is an integer, and linearly interpolating between these points otherwise.*

For notational convenience, we define:

$$I_{s,t}(x) = I(\widehat{M}^t \vec{s}, x).$$

Note that $I_{s,t}$ is a concave curve.

4.6.1 The Lovász-Simonovits lemma

The fundamental lemma of Lovász-Simonovits is the following (Lemma 1.4 of [LS90b], also refer to Theorem 7.3.3 of Lecture 7 of [Spia]).

Lemma 4.6.2. *Let* $\overline{x} = \min(x, n-x)$*. Consider any non-negative vector* $\vec{p}$*, and let* S_x *denote the level set of* $M\vec{p}$ *with* x *vertices.*

$$I(M\vec{p}, x) \leq (1/2)(I(\vec{p}, x - 2\overline{x}\Phi(S_x)) + I(\vec{p}, x - 2\overline{x}\Phi(S_x)))$$

The concavity of the curves implies monotonicity, $I(M\vec{p}) \leq I(\vec{p})$. The application of this lemma to our setting leads to the following statement.

Lemma 4.6.3. *For all* $t \leq \ell$ *and* $x \leq 1/\rho$*,*

$$I_{s,t}(x) \leq (1/2)(I_{s,t-1}(x(1 - \Phi(L_{s,t,x}))) + I_{s,t-1}(x(1 + \Phi(L_{s,t,x}))))$$

Let $f_{t,w,y}$ be the straight line between the points $(w, I_{s,t}(w))$ and $(y, I_{s,t}(y))$.

Lemma 4.6.4. *Let $t_0 < t_1 < \ldots < t_h$ be time steps. Suppose $\forall i \leq h$ and $x \in [w, y]$: $L_{s,t_i,x} \subseteq \text{supp}(\widehat{M}^t \vec{s}) \implies \Phi(L_{s,t_i,x}) \geq \psi$. Then, $\forall i \leq h, \forall x \in [w, y]$*

$$I_{s,t_i}(x) \leq f_{t_0-1,w,y}(x) + \sqrt{\min(x-w, y-x)}(1-\psi^2/128)^i$$

Proof. For convenience, let $\Delta_x = \min(x-w, y-x)$. We prove by induction over i.

For showing the base case take $i = 0$. Now consider the following cases.

- Suppose $x = w$ or $x = y$. By monotonicity, $I_{s,t_0}(x) \leq I_{s,t_0-1}(x)$. Since $x \in \{w, y\}$, the latter is exactly $f_{t_0,w,y}(x)$.
- Suppose $x \in [w+1, y-1]$. Then $\Delta_x \geq 1$ and $I_{s,t_0}(x) \leq 1 \leq \sqrt{\Delta_x}$.
- Suppose $x \in (w, w+1)$. Note that $\Delta_x = w - x < 1$. By the definition of the LS curve, $I_{s,t_0}(x) = I_{s,t_0}(w) + (w-x)(I_{s,t_0}(w+1) - I_{s,t_0}(w)) \leq I_{s,t_0-1}(w) + \sqrt{w-x} \leq f_{t_0-1,w,y}(x) + \sqrt{\Delta_x}$.
- Suppose $x \in (y-1, y)$. An identical argument to the above holds.

Now for the induction. Suppose the premise holds at step t_i. Namely for $x \in [w, y]$, for all level sets $L_{s,t_i,x}$ contained inside $\text{supp}(\widehat{M}^t \vec{s})$, $\Phi(L_{s,t_i,x}) \geq \psi \geq \psi$. We would like to upperbound $I_{s,t_i}(x)$. To this end, let us consider some $x \in [w, y]$. By Lemma 4.6.3,

$$\begin{aligned} I_{s,t_i}(x) &\leq (1/2)[I_{s,t_i-1}(x(1-\Phi(L_{s,t_i,x}))) + I_{s,t_i-1}(x(1+\Phi(L_{s,t_i,x})))] && (4.6) \\ &\leq (1/2)[I_{s,t_{i-1}}(x(1-\Phi(L_{s,t_i,x}))) + I_{s,t_{i-1}}(x(1+\Phi(L_{s,t_i,x})))] && (4.7) \end{aligned}$$

The second inequality follows by monotonicity, since $t_{i-1} \leq t_i - 1$. Note that $\Delta_x = \min(x-w, y-x) \leq x$ for all $x \in [w, y]$. Claim 4.6.5 (which we prove after the current lemma) shows the following.

Claim 4.6.5. *For all $1 \leq i \leq h$, for all $x \in [w, y]$, the following holds*

$$I_{s,t_i}(x) \leq (1/2)[I_{s,t_{i-1}}(x - \Delta_x\psi/4)) + I_{s,t_{i-1}}(x + \Delta_x\psi/4))] \quad (4.8)$$

Now, let $x_L = x - \Delta_x\psi/4$ and $x_R = x + \Delta_x\psi/4$. Using Claim 4.6.5 we get

$$\begin{aligned} I_{s,t_i}(x) &\leq (1/2)[f_{t_0-1,w,y}(x_L) + \sqrt{\Delta_{x_L}}(1-\psi^2/128)^{i-1} \\ &\quad + f_{t_0-1,w,y}(x_R) + \sqrt{\Delta_{x_R}}(1-\psi^2/128)^{i-1}] && (4.9) \\ &= (1/2)[f_{t_0-1,w,y}(x_L) + f_{t_0-1,w,y}(x_R)] \\ &\quad + (1/2)[\sqrt{\Delta_{x_L}})(1-\psi^2/8)^{i-1} + \sqrt{\Delta_{x_R}}(1-\psi^2/128)^{i-1}] && (4.10) \end{aligned}$$

Here, (4.10) follows from the induction hypothesis. Since $f_{t_0-1,w,y}$ is a linear function, the first term is exactly $f_{t_0-1,w,y}(x)$. We analyze the second term.

We first assume that $\Delta_x = x - w$ (instead of $y - x$).

$$\begin{aligned} \Delta_{x_L} &= \min(x - \psi\Delta_x/4 - w, y - x + \psi\Delta_x/4) && (4.11) \\ &= \min((1-\psi/4)\Delta_x, y - x + \psi/4\Delta_x) \leq (1-\psi/4)\Delta_x && (4.12) \end{aligned}$$

Analogously,

$$\Delta_{x_R} = \min(x + \psi\Delta_x/4 - w, y - x - \psi\Delta_x/4) \tag{4.13}$$
$$= \min((1+\psi/4)\Delta_x, y - x - \psi\Delta_x/4) \leq (1+\psi/4)\Delta_x \tag{4.14}$$

Thus, the second term of (4.10) is at most $(1/2)(1-\psi^2/128)^{i-1}\sqrt{\Delta_x}(\sqrt{1-\psi/4}+\sqrt{1+\psi/4})$.

Now, we consider $\Delta_x = y - x$.

$$\Delta_{x_L} = \min(x - \psi\Delta_x/4 - w, y - x + \psi\Delta_x/4) \tag{4.15}$$
$$= \min(x - \psi\Delta_x/4 - w, (1+\psi/4)\Delta_x) \leq (1+\psi/4)\Delta_x \tag{4.16}$$

Analogously,

$$\Delta_{x_R} = \min(x + \psi\Delta_x/4 - w, y - x - \psi\Delta_x/4) \tag{4.17}$$
$$= \min(x + \psi\Delta_x/4 - w, (1-\psi/4)\Delta_x) \leq (1-\psi/4)\Delta_x \tag{4.18}$$

In this case as well, the second term of (4.10) is at most $(1/2)(1-\psi^2/128)^{i-1}\sqrt{\Delta_x}(\sqrt{1-\psi/4}+\sqrt{1+\psi/4})$.

In both cases, we can upper bound (4.10) as follows. (We use the inequality $\frac{\sqrt{1-z}+\sqrt{1+z}}{2} \leq 1 - z^2/8$.

$$I_{s,t_i}(x) \leq f_{t_0-1,w,y}(x) + (1-\psi^2/128)^{i-1}\sqrt{\Delta_x}\frac{\sqrt{1-\psi/4}+\sqrt{1+\psi/4}}{2}$$
$$\leq f_{t_0-1,w,y}(x) + (1-\psi^2/128)^{i}\sqrt{\Delta_x}$$

□

Now, we establish Claim 4.6.5, the missing piece in the above proof.

Proof. (of Claim 4.6.5) Suppose $x_{max} \in [w,y]$ is the maximum value of $x \in [w,y]$ for which $L_{s,t_i,x}$ is still inside the support of the truncated diffusion at the t_i-th step. We split into three cases: $x \leq x_{max}$, $x \in (x_{max}, x_{max} + \Delta_{x_{max}}\psi/2]$, $x > x_{max} + \Delta_{x_{max}}\psi/2$. Note that in the latter two cases, $L_{s,t_i,x}$ is not contained in $\mathrm{supp}(\widehat{M}^{t_i}\vec{s})$.

Case 1, $x \leq x_{max}$**:** Note that (4.8) holds by concavity of the Lovász-Simonovits curve when $L_{s,t_i,x} \subseteq \mathrm{supp}(\widehat{M}^{t_i}\vec{s})$ (because then this level set has conductance at least ψ).

Case 2, $x \in (x_{max}, x_{max} + \Delta_{x_{max}}\psi/2]$**:** Let $S = L_{s,t_i,x_{max}}$ and let $T = L_{s,t_i,x}$. Observe that

$$\Phi(T) = \frac{|E(T,\overline{T})|}{d|T|} \overset{(1)}{\geq} \frac{|E(S,\overline{S})| - \psi/2 \cdot d|S|}{d|S| + \psi/2 \cdot d|S|} \overset{(2)}{\geq} \frac{\psi d|S|/2}{2d|S|} \geq \frac{\psi}{4} \tag{4.19}$$

Here, (**1**) follows because T could contain at most $\psi|S|/2$ neighbors of S which could cost us at most $\psi d|S|/2$ edges in the cut $(S,\overline{S})$. (**2**) follows by upperbounding ψ by 1. Again

the claim in (4.8) follows by concavity of the Lovázs-Simonovits curve.

Case 3, $x > x_{max} + \Delta_{x_{max}}\psi/2$**:** Now let $x_r = x_{max} + \Delta_{x_{max}}\psi/2$. Write $x = x_{max} + \Delta_{x_{max}}\psi/2 + s$. Recall $\Delta_x = \min(x - w, y - x)$. We claim that $x - \Delta_x\psi/4 \geq x_{max}$. First let us see how to establish (4.8) assuming this claim holds. Assuming this claim, we have

$$I_{s,t_i}(x - \Delta_x\psi/4) = I_{s,t_i}(x_{max}) = I_{s,t_i}(x + \Delta_x\psi/4) = \|\widehat{M}^{t_i}\vec{s}\|_1.$$

And therefore,

$$\begin{aligned} I_{s,t_i}(x) &= \frac{1}{2} \cdot [I_{s,t_i}(x - \Delta_x\psi/4) + I_{s,t_i}(x + \Delta_x\psi/4)] \\ &\leq \frac{1}{2} \cdot \left[I_{s,t_{i-1}}(x - \Delta_x\psi/4) + I_{s,t_{i-1}}(x + \Delta_x\psi/4)\right] \end{aligned}$$

Now, all that remains to establish (4.8) is to show $x - \Delta_x\psi/4 \geq x_{max}$. For simplicity, write $\Delta_m = \Delta_{x_{max}}$. Now consider two cases depending on the value of Δ_m

1. **Case 1** $\Delta_m = x_{max} - w$.

   In this case note that

   $$\begin{aligned} x - \Delta_x\psi/4 &= x_{max} + \Delta_m\psi/2 + s - (x - w)\psi/4 \\ &\geq x_{max} + \Delta_m\psi/2 + s - (x_{max} + \Delta_m\psi/2 + s - w)\psi/4 \\ &\geq x_{max} + \Delta_m\psi/4 - \Delta_m\psi^2/8 + s - s\psi/4 \\ &\geq x_{max} + \Delta_m\psi/8 + s(1 - \psi/4) \geq x_{max} \end{aligned}$$

   which establishes the claim above as desired.

2. **Case 2** $\Delta_m = y - x_{max}$.

   In this case note that

   $$\begin{aligned} x - \Delta_x\psi/4 &= x_{max} + \Delta_m\psi/2 + s - (y - x)\psi/4 \\ &\geq x_{max} + \Delta_m\psi/2 + s - (y - x_{max} - \Delta_m\psi/2 - s)\psi/4 \\ &\geq x_{max} + \Delta_m\psi/4 + \Delta_m\psi^2/8 + s + s\psi/4 \\ &\geq x_{max} \end{aligned}$$

Thus, in both cases, the claim from above holds. This means that (4.8) holds as long as the premise holds for the t_i-th step. □

4.6.2 From leaking timesteps to the dropping of the LS curve

We fix a source vertex s, and consider the evolution of $\widehat{M}^t\vec{s}$. Therefore, we drop the dependence of s from much of the notation.

We use $\widehat{p}_t$ to denote $\widehat{M}^t\vec{s}$. We begin with a few definitions.

Definition 4.6.6. *A timestep t is called* leaking for source s *if, for all $k \le \rho^{-1}$: if $L_{s,t,k} \subseteq \mathrm{supp}(\widehat{M}^t\vec{s})$ and $|L_{s,t,k} \cap F| \ge \alpha^2 k/400$, then $\Phi(L_{s,t,k}) \ge 1/d\ell^{1/3}$.*

*If timestep t is not leaking for s, there exists $k \le \rho^{-1}$ such that $L_{s,t,k} \subseteq \mathrm{supp}(\widehat{M}^t\vec{s})$, $|L_{s,t,k} \cap F| \ge \alpha^2 k/400$, and $\phi(L_{s,t,k}) < 1/d\ell^{1/3}$. Such a k is denoted as an (s,t)-*certificate of non-leakiness.

We set $\alpha = \varepsilon^{4/3}/300,000$.

Following the construction of the LS curve $I_{s,t}$, we will order each vector $\widehat{p}_t$ in decreasing order, breaking ties by id. The *rank* of a vertex is its position in (the sorted version of) $\widehat{p}_t$.

Definition 4.6.7. *Let the* bucket $B_{t,r}$ *denote the set of vertices whose rank in $\widehat{p}_t$ is in the range $[2^r, 2^{r+1})$.*

A bucket $B_{t,r}$ is called heavy *if $\sum_{v \in B_{t,r} \cap F} \widehat{p}_t(v) \ge \alpha$. (The bucket restricted to F has large probability.)*

The following lemma says that if there are many leaking timesteps, then the LS curve drops at heavy buckets.

Lemma 4.6.8. *Fix $r \ge 0$. Suppose for some $s \in F$, there exist $\ell' \ge \beta^3\ell/8$ leaking timesteps $t_0 < t_1 < \ldots < t_{\ell'}$ such that for all $0 \le i \le \ell'$, $B_{t_i,r}$ is heavy. Then, $I_{s,t_{\ell'}}(2^{r+1}) < I_{s,t_0}(2^{r+1}) - \alpha/4$.*

The main tool used in our proof is our adaptation of Lovász-Simonovits lemma done in Lemma 4.6.4. We first make a definition.

Definition 4.6.9. *Fix $r \ge 0$, a source s and a timestep t. A vertex $w \in [2^r, 2^{r+1}]$ is called a balanced split for t if $|L_{t,w} \cap F| \ge \alpha 2^r/3$ and $\sum_{v \in B_{t,r} \setminus L_{t,w}} \widehat{p}_t(v) \ge \alpha/3$.*

We will first prove the following claim which essentially follows by averaging arguments.

Claim 4.6.10. *Fix $r \ge 0$ and suppose for some source vertex $s \in F$, there exist ℓ' leaking timesteps $t_0 < t_1 < \ldots < t_{\ell'}$ such that for all $0 \le i \le \ell'$, $B_{t_i,r}$ is heavy. Then, there exists a vertex w that is a balanced split for at least an $\alpha/3$-fraction of timesteps in $T = \{t_0, t_1, \ldots t_{\ell'}\}$.*

Proof. Since $B_{t_0,r}$ is heavy, $I_{s,t_0}(2^r) < 1$. Since the support of $\widehat{p}_t$ is at most ρ^{-1}, this implies that $2^r < \rho^{-1}$ and $r \le -\lg \rho$ (and this holds by the choice of parameters).

For all $v \in B_{t,r}$, $\widehat{p}_t(v) \le 1/2^r$. Since $\sum_{v \in B_{t,r} \cap F} \widehat{p}_t(v) \ge \alpha$, $|B_{t,r} \cap F| \ge \alpha 2^r$.

For convenince, let $T = \{t_0, t_1, \ldots t_{\ell'}\}$. Pick w uar in $[2^r, 2^{r+1})$. Let X_i be the indicator for w being a balanced split for t_i. Recall that $|B_{t_i,r} \cap F| \ge \alpha 2^r$. Sort the vertices of $B_{t_i,r} \cap F$

by increasing rank and consider the vertices in positions $\alpha 2^r/3$ and $2\alpha 2^r/3]$. Let the rank corresponding to these vertices by u_1 and u_2. We first argue that any rank $w \in [u_1, u_2]$ is a balanced split. We have $|L_{t,w} \cap F| \geq \alpha 2^r/3$ because $w \geq u_1$. For all $v \in B_{t_i,r}$, $\widehat{p}_{t_i}(v) \leq 1/2^r$. Thus, $\sum_{v \in L_{t_i,u_2} \cap B_{t_i,r}} \widehat{p}_{t_i}(v) \leq (1/2^r)(2\alpha 2^r/3) = 2\alpha/3$. Note that $\sum_{v \in B_{t_i,r}} \widehat{p}_t(v) \geq \alpha$, since the bucket is heavy Hence, for any $w \leq u_2$, $\sum_{v \in B_{t,r} \setminus L_{t,w}} \widehat{p}_t(v) \geq \alpha - 2\alpha/3 = \alpha/3$.

As a consequence, for any t_i, there are at least $\alpha 2^r/3$ values of w that are balanced splits. In other words, $\mathbf{E}[X_i] \geq \alpha/3$. By linearity of expectation, $\mathbf{E}[\sum_{i \leq \ell'} X_i] \geq \alpha \ell'/3$. Thus, there must exist some $w \in [2^r, 2^{r+1})$ that is a balanced split for at least $\alpha \ell'/3$ timesteps. □

Next, we show the following claim which essentially uses leakiness of a timestep $t \in T$ and the balanced split vertex w promised by Claim 4.6.10 to spell out a set with enough free vertices with large conductance.

Claim 4.6.11. *Fix $r \geq 0$ and let $w \in [2^r, 2^{r+1})$ be a split vertex as promised by Claim 4.6.10 and let $t_{i_1} < t_{i_2} < \ldots < t_{i_{\alpha\ell'/3}}$ denote the timesteps for which w is a balanced split. Let $y = \min(2^{r+6+\lceil \lg(1/\alpha) \rceil}, \rho^{-1})$. Then, for all $x \in [w, y]$ and for all $t \in \{t_{i_1}, t_{i_2}, \cdots, t_{i_{\alpha\ell'/3}}\}$, whenever $L_{t,x} \subseteq \mathrm{supp}(\widehat{M}^t \vec{s})$, then $\Phi(L_{t,x}) \geq 1/d\ell^{1/3}$.*

Proof. Take $x \in [w, y]$ and a leaking timestep $t \in \{t_{i_1}, t_{i_2}, \cdots, t_{i_{\alpha\ell'/3}}\}$. Note that $x \leq y \leq \rho^{-1}$ clearly holds. Now, to establish the lower bound on conductance claimed, we first unpack what it means for t to be a leaking timestep Def 4.6.6. It says: If $L_{t,x} \subseteq \mathrm{supp}(\widehat{M}^t \vec{s})$ and $|L_{t,x} \cap F| \geq \alpha^2 k/400$, then it better hold that $\phi(L_{t,x}) \geq 1/d\ell^{1/3}$.

Note that $y \leq 2^{r+6+\lceil \lg(1/\alpha) \rceil} \in [2^r(64/\alpha), 2^{r+1}(64/\alpha)]$. Since $r \leq -\lg \rho$, $y \leq 128(\rho\alpha)^{-1}$.

Note that for all $t \in \{t_{i_1}, t_{i_2}, \cdots, t_{i_{\alpha\ell'/3}}\}$ and $x \in [w, y]$, $L_{t,x}$ contains at least $\alpha 2^r/3$ vertices of F. Thus, at least a $(\alpha 2^r/3)/(2^{r+1} \cdot 64/\alpha) \geq \alpha^2/400$-fraction of $L_{t,x}$ is in F. Now note that since t is leaking, we see that one of the following will hold. Either

- $L_{t,x} \subseteq \mathrm{supp}(\widehat{M}^t \vec{s})$ and $\Phi(L_{t,x}) \geq 1/d\ell^{1/3}$, Or
- $L_{t,x} \not\subseteq \mathrm{supp}(\widehat{M}^t \vec{s})$.

And this establishes the claim. □

Now, we have all the ingredients to prove Lemma 4.6.8. The key step which remains is an application of Lemma 4.6.4.

Proof. (Of Lemma 4.6.8) Suppose $w \in [2^r, 2^{r+1})$ is a balanced split at $\alpha\ell'/3$ timesteps as promised by Claim 4.6.10. Let $y = \min(2^{r+6+\lceil \lg(1/\alpha) \rceil}, \rho^{-1})$ and as observed in Claim 4.6.11, note that for $x \in [w, y]$ if $L_{t,x} \subseteq \mathrm{supp}(\widehat{M}^t \vec{s})$, it holds that $\phi(L_{t,x}) \geq 1/d\ell^{1/3}$. Now, we apply Lemma 4.6.4. For all $x \in [w, y]$, we have $I_{s,t_{\ell'}}(x) \leq I_{s,t_{i_{\alpha\ell'/3}}}(x) \leq f_{t_{i_1-1},w,y}(x) + \sqrt{x}(1 - 1/128d^2\ell^{2/3})^{\alpha\ell'/3}$. By the premise, $\ell' \geq \beta^3\ell/8$ and therefore we have

$$(1-1/128d^2\ell^{2/3})^{\alpha\ell'/3} \leq (1-1/128d^2\ell^{2/3})^{\alpha\beta^3\ell/3} = (1-1/128d^2\ell^{2/3})^{128d^2\ell^{2/3} \cdot \frac{\alpha\beta^3\ell^{1/3}}{3 \cdot 128d^2}} \leq \exp(-1/\alpha)$$

which holds because, for sufficiently small $\varepsilon > 0$, we have

$$\ell^{1/3} = \frac{d^2}{\varepsilon^{10}} \geq \frac{d^2 \cdot 10^{20}}{\varepsilon^7} \geq \frac{d^2}{\alpha^3 \beta^3}.$$

Further, by the monotonicity of LS curves, $I_{s,t_{\ell'}}(x) \leq f_{t_{i_1-1},w,y}(x) + \exp(-1/\alpha) \leq f_{t_{i_0},w,y}(x) + \exp(-1/\alpha)$. Specifically, we get

$$I_{s,t_{\ell'}}(2^{r+1}) \leq f_{t_{i_0},w,y}(2^{r+1}) + \exp(-1/\alpha). \tag{4.20}$$

Since w is a good split, $I_{s,t_{i_0}}(2^{r+1}) \geq I_{s,t_{i_0}}(w) + \alpha/3$. Note that

$$\begin{aligned} f_{t_{i_0},w,y}(2^{r+1}) &= I_{s,t_{i_0}}(w) + (2^{r+1} - w)\left(\frac{I_{s,t_{i_0}}(y) - I_{s,t_{i_0}}(w)}{y - w}\right) \\ &\leq I_{s,t_{i_0}}(w) + 2^{r+1}/(y/2) \qquad (4.21) \\ &\leq I_{s,t_{i_0}}(w) + 2^{r+1} \times \left(\frac{2\alpha}{2^r \cdot 64}\right) = I_{s,t_{i_0}}(w) + \alpha/16 \qquad (4.22) \end{aligned}$$

The first inequality above follows by upper bounding $I_{s,t_{i_0}}(y) - I_{s,t_{i_0}}(w)$ by 1, dropping the negative term and noting that $y - w \geq y/2$ for a sufficiently small α. Together with (4.20), we get

$$\begin{aligned} I_{s,t_{\ell'}}(2^{r+1}) \leq f_{t_{i_0},w,y}(2^{r+1}) + \exp(-1/\alpha) &\leq I_{s,t_{i_0}}(w) + \alpha/16 + \exp(-1/\alpha) \\ &\leq I_{s,t_{i_0}}(2^{r+1}) - \alpha/3 + \alpha/16 + \exp(-1/\alpha) \qquad (4.23) \end{aligned}$$

By monotonicity of the LS curve, $I_{s,t_{\ell'}}(2^{r+1}) < I_{s,t_0}(2^{r+1}) - \alpha/4$.

□

Now, we state a key lemma. It says that a fixed bucket (parameterized by r) satisfies the following at most timesteps: (i) either it does not contain enough free vertices, or (ii) if it contains many free vertices at a particular timestep, then most of the corresponding timesteps are not leaky.

Lemma 4.6.12. *Fix $r \geq 0$ and take any $s \in F$. There are at most $\beta^3\ell/\alpha$ leaking timesteps t (with respect to s) where $B_{t,r}$ is heavy.*

Proof. We prove by contradiction. Suppose there are more than $\beta^3\ell/\alpha$ leaking timesteps t where $B_{t,r}$ is heavy. We break these up into $4/\alpha$ contiguous blocks of $\beta^3\ell/4$ leaking timesteps. By Lemma 4.6.8, after every such block of timesteps, $I_{s,t}(2^{r+1})$ reduces by more than $\alpha/4$. Note that $I_{s,0}(2^{r+1}) \leq 1$, and thus, after $4/\alpha$ blocks, $I_{s,t}(2^{r+1})$ becomes negative. Contradiction to the non-negativity of $I_{s,t}(2^{r+1})$. □

4.6.3 Proof of Theorem 4.3.1

We finally prove Theorem 4.3.1. In particular, recall that this theorem claims that for an arbitrary set $F \subseteq V$ with $|F| \geq \beta n$, there exists a size threshold k such that one can find enough source vertices $s \in F$ such that ℓ-step diffusions from s contain enough non-leaky timesteps. Moreover, these non-leaky timesteps can be used to obtain a low conductance cut restricted to F. We begin by showing that indeed many sources $s \in F$ have the desired behavior.

Lemma 4.6.13. *There are at least $\beta^2 n/8$ vertices $s \in F$, such that: there are at least $\beta\ell/16$ timesteps t in $[\ell]$ that are not leaking for s.*

Proof. We fix any vertex s satisfying the conditions of Theorem 4.5.1. Let us recall what this means. This means that for at least $\beta\ell/8$ timesteps t, it holds that $\widehat{M}^t\vec{s}(F) \geq \beta/16$. We will show that conclusion in Lemma 4.6.13 above holds for s which will establish the lemma. We prove by contradiction.

To this end, let us suppose for any vertex s satisfying the conditions of Theorem 4.5.1, there are at most $\beta\ell/16$ non-leaky timesteps. There are at least $\beta\ell/8 - \beta\ell/16 = \beta\ell/16$ timesteps t that are leaking for s, such that $\widehat{M}^t\vec{s}(F) \geq \beta/16$. Fix any such timestep t and consider the buckets $B_{t,r}$. There are at most $-\lg\rho$ buckets with non-zero probability mass, and by averaging, there exists $r \leq -\lg\rho$ such that

$$\sum_{v \in F \cap B_{t,r}} \widehat{p}_t(v) \geq \beta/(-16\lg\rho) = \frac{\varepsilon}{160 \cdot 3000 \lg(1/\varepsilon)} \geq \frac{\varepsilon^{4/3}}{300,000} = \alpha$$

where the last step holds for sufficiently small ε and therefore, $B_{t,r}$ is heavy.

Thus, for each of the $\beta\ell/16$ leaking timesteps t above, there exists some $r \leq -\lg\rho$ such that $B_{t,r}$ is heavy. By averaging, there exists some $r \leq -\lg\rho$ such that for $\beta\ell/(-16\lg\rho)$ leaking timesteps t, $B_{t,r}$ is heavy. However, for sufficiently small ε ($\varepsilon < 2^{-30}$), we have

$$\frac{\beta\ell}{-16\lg\rho} = \frac{\varepsilon \cdot \ell}{160 \cdot 3000 \log(1/\varepsilon)} \geq 1000\varepsilon^{3-4/3}\ell \geq \frac{\beta^3\ell}{\alpha}$$

which contradicts Lemma 4.6.12. □

Lemma 4.6.14. *Let $|F| \geq \beta n$. There exists a $r \leq \lg(1/\rho)$ such that for $\geq \beta^2 n/(8\lg^2(\rho^{-1}))$ vertices $s \in F$, the following holds. For at least $\beta\ell/(\lg^2(\rho^{-1}))$ timesteps t, there exists $k \in [2^r, 2^{r+1}]$ that is an (s,t)-certificate of non-leakiness.*

Proof. This is an averaging argument. Apply Lemma 4.6.13. For each of the $\beta^2 n/8$ vertices $s \in F$, there are at least $\beta\ell/16$ timesteps t that are not leaking for s. Thus, for every such (s,t) pair, there exists $k_{s,t} \leq \rho^{-1}$ that is an (s,t)-certificate of non-leakiness. We basically bin the logarithm of the certificates. Thus, to every pair (s,t) (of the above form), we associate $r_{s,t} = \lfloor \lg k_{s,t} \rfloor$. By averaging, for each relevant s, there is a value r_s such that for at least

$\beta\ell/(16\lg(\rho^{-1}))$ timesteps t, there is an (s,t)-certificate in $[2^{r_s}, 2^{r_s+1}]$. Again, by averaging there exists $r \le \lg(\rho-1)$ such that there are at least $\beta^2 n/(8\lg(\rho^{-1})) \ge \beta^2 n/(\lg^2(\rho^{-1}))$ vertices $s \in F$ for which there exist at least $\beta\ell/(16\lg(\rho^{-1})) \ge \beta\ell/\lg^2(\rho^{-1})$ timesteps t, such that there is an (s,t)-certificate for non-leakiness in $[2^r, 2^{r+1}]$. □

Theorem 4.3.1 follows as a corollary of Lemma 4.6.14. We now present the proof.

Proof. (Of Theorem 4.3.1) As seen from Lemma 4.6.14, there exists some $r \le -\lg(\rho)$ such that there are at least $\Omega(\beta^2/\lg(\beta^{-1})) \cdot n$ vertices $s \in F$ each of which in turn has (s,t)-certificates of non-leakiness for at least $\Omega(\beta/16\lg^2(\beta^{-1})) \cdot \ell$ different values of t. We simply choose $k = 2^r$.

Let $S \subseteq F$ denote the collection of these relevant sources. And for $s \in S$, define

$$C_s = \{t \le \ell : \text{ there exists a } (s,t) - \text{ certificate of non-leakiness}\}.$$

Take $s \in S$, $t \in C_s$. We will show that there exists $k' = k'(s,t) \in [k, 2k]$ such that the level set $L_{s,t,k'}$ satisfies the following.

- $L_{s,t,k} \subseteq \text{supp}(\widehat{M}^t \vec{s})$.
- $\phi(L_{s,t,k'} \cup \{s\}) \le 1/\ell^{1/3}$.
- $|L_{s,t,k'} \cap F| \ge \alpha^2 k'/400 \ge \beta^3 k$.

The first item above follows from the conclusion of Lemma 4.6.14, Def 4.6.6 and taking contrapositive in Lemma 4.6.4. Unpacking, this means that since $t \in C_s$ is a non-leaking timestep for s, it follows that there exists $k' = k'(s,t) \in [k, 2k]$ for which $L_{s,t,k'} \subseteq \text{supp}(\widehat{M}^t \vec{s})$. The last item above holds for this choice of k' from the conclusion of Lemma 4.6.14. For item 2 above, again note that our choice of k' and Lemma 4.6.14 imply that

$$\phi(L_{s,t,k'}) \le 1/d\ell^{1/3} = 1/d \cdot \frac{\varepsilon^{10}}{d^2} = \varepsilon^{10}/d^3 = \phi/d^3$$

and therefore $\phi(L_{s,t,k'} \cup \{s\}) \le \phi$ also follows as by (possibly) including a single vertex in the set, the number of cut-edges can only increase by d. □

4.7 Proofs of applications

The proofs here are quite straightforward and appear (in some form) in previous work. We sketch the proofs, and do not give out the specifics of the Chernoff bound calculations. Specifically, we mention Theorem 9.28 and its proof in [Gol17], which contains these calculations.

Proof. (of Theorem 4.0.3) Given input graph G, we set up the partition oracle with proximity parameter $\varepsilon/8$. Therefore, with probability at least $2/3$ over the random seed $\boldsymbol{R}$, the number of cut edges is at most $\varepsilon dn/8$. The tester repeats the following $O(1)$ times. For a random $\boldsymbol{R}$, we first estimate the number of edges cut by random sampling. The tester samples $\Theta(1/\varepsilon)$ uar vertices u, picks a uar neighbor v of u, and calls the partition oracle on u and v. If these lie in

different components, the edge (u, v) is cut. If more that an $\varepsilon/4$ fraction of edges are cut, then repeat with a new $\boldsymbol{R}$. Otherwise, we fix the seed $\boldsymbol{R}$ and proceed to the second phase of the tester. (If no such $\boldsymbol{R}$ is found, the tester rejects.)

In the second phase, we sample a multiset $S \subseteq V$ of $O(\varepsilon^{-1})$ uar vertices, and query the subgraph induced by the component $C(v)$ (of the partition given by the oracle) that each $v \in S$ belongs to. For each (poly(ε^{-1})-sized) component $C(v)$, we directly determine if it belongs to $\mathcal{Q}$. (If there is an efficient algorithm, we can run that algorithm.) If any of these components does not belong to $\mathcal{Q}$, the tester rejects, otherwise it accepts.

Now, let us argue that this is a bonafide tester for $\mathcal{Q}$. Recall $\mathcal{Q}$ is both monotone and additive. Suppose $G \in \mathcal{Q}$. Since $\mathcal{Q}$ is a subproperty of a minor-closed property, the first phase of setting the partition oracle succeeds with high probability. Since $\mathcal{Q}$ is monotone and additive, all the subgraphs induced on the connected components $C(v)$ also satisfy $\mathcal{Q}$. So the tester accepts whp. Suppose G is ε-far from $\mathcal{Q}$. If the first phase does not succeed, then the tester rejects. So assume that the first phase succeeds. Whp, by a Chernoff bound, the number of cut edges (of the partition) is at most $\varepsilon dn/2$. Since $\mathcal{Q}$ is monotone, the graph obtained by removing these cut edges is at least $\varepsilon/2$-far from $\mathcal{Q}$. Since $\mathcal{Q}$ is additive, at least $\Omega(\varepsilon n)$ vertices participate in connected components that not in $\mathcal{Q}$. Hence, by a Chernoff bound, the second phase rejects whp.

The query complexity has at most an $O(d\varepsilon^{-1})$ multiplicative overhead of the time complexity of the partition oracle, which is poly$(d\varepsilon^{-1})$. If $\mathcal{Q}$ can be decided in polynomial time, then the second phase also runs in poly$(d\varepsilon^{-1})$ time. □

Proof. (of Theorem 4.0.4) As with the previous proof, we set up the partition oracle with proximity parameter $\varepsilon dn/c$, where c is the largest amount by which an edge addition/deletion changes f. As before, there is a first phase to determine an appropriate setting of $\boldsymbol{R}$ for the partition oracle. We sample poly$(d\varepsilon^{-1})$ uar vertices and determine the component that each vertex belongs to. For each component, we compute f exactly. We take the sum of f-values, and rescale appropriately to get an additive εnd estimate for f. □

Chapter 5

The impossibility of low rank representations for triangle-rich complex networks

Complex networks (or graphs) are a fundamental object of study in modern science, across domains as diverse as the social sciences, biology, physics, computer science, and engineering [WF94, New03, EK10a]. Designing good models for these networks is a crucial area of research, and also affects society at large, given the role of online social networks in modern human interaction [BA99, WS98, CF06]. Complex networks are massive, high-dimensional, discrete objects, and are challenging to work with in a modeling context. A common method of dealing with this challenge is to construct a low-dimensional Euclidean embedding that tries to capture the structure of the network (see [HYL17b] for a recent survey). Formally, we think of the n vertices as vectors $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n \in \mathbb{R}^d$, where d is typically constant (or very slowly growing in n). The likelihood of an edge (i, j) is proportional to (usually a non-negative monotone function in) $\vec{v}_i \cdot \vec{v}_j$ [ASN+13, CLX16]. This gives a graph distribution that the observed network is assumed to be generated from.

The most important method to get such embeddings is the Singular Value Decomposition (SVD) or other matrix factorizations of the adjacency matrix [ASN+13]. Recently, there has also been an explosion of interest in using methods from deep neural networks to learn such graph embeddings [PARS14, TQW+15a, CLX16, GL16b] (refer to [HYL17b] for more references). Regardless of the specific method, a key goal in building an embedding is to keep the dimension d small — while trying to preserve the network structure — as the embeddings are used in a variety of downstream modeling tasks such as graph clustering, nearest neighbor search, and link prediction [Twi18]. Yet a fundamental question remains unanswered: to

what extent do such low dimensional embeddings actually capture the structure of a complex network?

These models are often justified by treating the (few) dimensions as "interests" of individuals, and using similarity of interests (dot product) to form edges. Contrary to the dominant view, we argue that low-dimensional embeddings are *not* good representations of complex networks. We demonstrate mathematically and empirically that they lose local structure, one of the hallmarks of complex networks. This runs counter to the ubiquitous use of SVD in data analysis. The weaknesses of SVD have been empirically observed in recommendation tasks [BCG10, GGL+13, KUK17], and our result provides a mathematical validation of these findings.

Let us define the setting formally. Consider a set of vectors $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n \in \mathbb{R}^d$ (denoted by the $d \times n$ matrix V) used to represent the n vertices in a network. Let $\mathcal{G}_V$ denote the following distribution of graphs over the vertex set $[n]$. For each index pair i, j, independently insert (undirected) edge (i, j) with probability $\max(0, \min(\vec{v}_i \cdot \vec{v}_j, 1))$. (If $\vec{v}_i \cdot \vec{v}_j$ is negative, (i, j) is never inserted. If $\vec{v}_i \cdot \vec{v}_j \geq 1$, (i, j) is always inserted.) We will refer to this model as the "embedding" of a graph G, and focus on this formulation in our theoretical results. This is a standard model in the literature, and subsumes the classic Stochastic Block Model [HLL83] and Random Dot Product Model [YS07, AFL+18]. There are alternate models that use different functions of the dot product for the edge probability, which are discussed in Section 5.0.2. Matrix factorization is a popular method to obtain such a vector representation: the original adjacency matrix A is "factorized" as $V^T V$, where the columns of V are $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$.

Two hallmarks of real-world graphs are: (i) Sparsity: The average degree is typically constant with respect to n, and (ii) Triangle density: there are many triangles incident to low degree vertices [WS98, SCW+10, SKP12, DPKS12]. The large number of triangles is considered a local manifestation of community structure. Triangle counts have a rich history in the analysis and algorithmics of complex networks. Concretely, we measure these properties simultaneously as follows.

Definition 5.0.1. *For parameters $c > 1$ and $\Delta > 0$, a graph G with n vertices has a* (c, Δ)-triangle foundation *if there are at least Δn triangles contained among vertices of degree at most c. Formally, let S_c be the set of vertices of degree at most c. Then, the number of triangles in the graph induced by S_c is at least Δn.*

Typically, we think of both c and Δ as constants. We emphasize that n is the total number of vertices in G, not the number of vertices in S (as defined above). Refer to real-world graphs in Table 5.2. In Figure 5.1, we plot the value of c vs Δ. (Specifically, the y axis is the number of triangles divided by n.) This is obtained by simply counting the number of triangles contained in the set of vertices of degree at most c. Observe that for all graphs, for $c \in [10, 50]$, we get a value of $\Delta > 1$ (in many cases $\Delta > 10$).

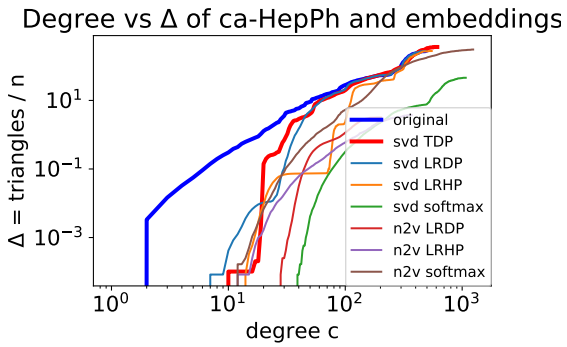

Figure 5.1: figure

Plots of degree c vs Δ: For a High Energy Physics coauthorship network, we plot c versus the total number of triangles only involving vertices of degree at most c. We divide the latter by the total number of vertices n, so it corresponds to Δ, as in Def. 5.0.1. We plot these both for the original graph (in thick blue), and for a variety of embeddings (explained in Section 5.0.2). For each embedding, we plot the maximum Δ in a set of 100 samples from a 100-dimensional embedding. The embedding analyzed by our main theorem (TDP) is given in thick red. Observe how the embeddings generate graphs with very few triangles among low degree vertices. The gap in Δ for low degree is 2-3 orders of magnitude. The other lines correspond to alternate embeddings, using the NODE2VEC vectors and/or different functions of the dot product.

Our main result is that *any* embedding of graphs that generates graphs with (c, Δ)-triangle foundations, with constant c, Δ, must have near linear rank. This contradicts the belief that low-dimensional embeddings capture the structure of real-world complex networks.

Theorem 5.0.2. *Fix $c > 4, \Delta > 0$. Suppose the expected number of triangles in $G \sim \mathcal{G}_V$ that only involve vertices of expected degree c is at least Δn. Then, the rank of V is at least* $\min(1, \text{poly}(\Delta/c))n/\lg^2 n$.

Equivalently, graphs generated from low-dimensional embeddings cannot contain many triangles only on low-degree vertices. We point out an important implication of this theorem for Stochastic Block Models. In this model, each vertex is modeled as a vector in $[0, 1]^d$, where the ith entry indicates the likelihood of being in the ith community. The probability of an edge is exactly the dot product. In community detection applications, d is thought of as a constant, or at least much smaller than n. On the contrary, Theorem 5.0.2 implies that d must be $\Omega(n/\lg^2 n)$ to accurately model the low-degree triangle behavior.

5.0.1 Empirical validation

We empirically validate the theory on a collection of complex networks detailed in Table 5.2. For each real-world graph, we compute a 100-dimensional embedding through SVD (basically, the top 100 singular vectors of the adjacency matrix). We generate 100 samples of graphs from these embeddings, and compute their c vs Δ plot. This is plotted with the true c vs Δ plot. (To account for statistical variation, we plot the *maximum* value of Δ observed in the samples, over all graphs. The variation observed was negligible.) Figure 5.1 shows such a plot for a physics coauthorship network. More results are given in Section 5.3.

Note that this plot is significantly off the mark at low degrees for the embedding. Around the lowest degree, the value of Δ (for the graphs generated by the embedding) is 2-3 order of magnitude smaller than the original value. This demonstrates that the local triangle structure is destroyed around low degree vertices. Interestingly, the total number of triangles is preserved well, as shown towards the right side of each plot. Thus, a nuanced view of the triangle distribution, as given in Def 5.0.1, is required to see the shortcomings of low dimensional embeddings.

5.0.2 Alternate models

We note that several other functions of dot product have been proposed in the literature, such as the softmax function [PARS14, GL16b] and linear models of the dot product [HYL17b]. Theorem 5.0.2 does not have direct implications for such models, but our empirical validation holds for them as well. The embedding in Theorem 5.0.2 uses the *truncated dot product* (TDP) function $max(0, \min(\vec{v}_i \cdot \vec{v}_j, 1))$ to model edge probabilities. We construct

other embeddings that compute edge probabilities using machine learning models with the dot product and Hadamard product as features. This subsumes linear models as given in [HYL17b]. Indeed, the TDP can be smoothly approximated as a logistic function. We also consider (scaled) softmax functions, as in [PARS14], and standard machine learning models (LRDP, LRHP). (Details about these models are given in Section 5.35.3.2.)

For each of these models (softmax, LRDP, LRHP), we perform the same experiment described above. Figure 5.1 also shows the plots for these other models. Observe that *none* of them capture the low-degree triangle structure, and their Δ values are all 2-3 orders of magnitude lower than the original.

In addition (to the extent possible), we compute vector embeddings from a recent deep learning based method (node2vec [GL16b]). We again use all the edge probability models discussed above, and perform an identical experiment (in Figure 5.1, these are denoted by "n2v"). Again, we observe that the low-degree triangle behavior is not captured by these deep learned embeddings.

5.0.3 Broader context

The use of geometric embeddings for graph analysis has a rich history, arguably going back to spectral clustering [Fie73]. In recent years, the Stochastic Block Model has become quite popular in the statistics and algorithms community [HLL83]. and the Random Dot Product Graph model is a generalization of this notion (refer to recent surveys [Abb18, AFL+18]). As mentioned earlier, Theorem 5.0.2 brings into question the standard uses of these methods to model social networks. The use of vectors to represent vertices is sometimes referred to as *latent space models*, where geometric proximity models the likelihood of an edge. Although dot products are widely used, we note that some classic latent space approaches use Euclidean distance (as opposed to dot product) to model edge probabilities [HRH02], and this may avoid the lower bound of Theorem 5.0.2. Beyond graph analysis, the method of Latent Semantic Indexing (LSI) also falls in the setting of Theorem 5.0.2, wherein we have a low-dimensional embedding of "objects" (like documents) and similarity is measured by dot product [lsi19].

5.1 High-level description of the proof

In this section, we sketch the proof of Theorem 5.0.2. The sketch provides sufficient detail for a reader who wants to understand the reasoning behind our result, but is not concerned with technical details. We will make the simplifying assumption that all v_is have the same length L. We note that this setting is interesting in its own right, since it is often the case in practice that all vectors are non-negative and normalized. In this case, we get a stronger rank lower bound that is linear in n. Section 5.1.1 provides intuition on how we can remove this assumption. The

full details of the proof are given in Section 5.2.

First, we lower bound L. By Cauchy-Schwartz, $\vec{v}_i \cdot \vec{v}_j \leq L^2$. Let $X_{i,j}$ be the indicator random variable for the edge (i,j) being present. Observe that all $X_{i,j}$s are independent and $\mathbf{E}[X_{i,j}] = \min(\vec{v}_i \cdot \vec{v}_j, 1) \leq L^2$.

The expected number of triangles in $G \sim \mathcal{G}_V$ is:

$$\mathbf{E}[\sum_{i \neq j \neq k} X_{i,j} X_{j,k} X_{i,k}] \tag{5.1}$$

$$\leq \sum_i \sum_{j,k} \mathbf{E}[X_{j,k}]\mathbf{E}[X_{i,j}]\mathbf{E}[X_{i,k}] \tag{5.2}$$

$$\leq L^2 \sum_i \sum_{j,k} \mathbf{E}[X_{i,j}]\mathbf{E}[X_{i,k}] = L^2 \sum_i (\sum_j \mathbf{E}[X_{i,j}])^2 \tag{5.3}$$

Note that $\sum_j \mathbf{E}[X_{i,j}] = \mathbf{E}[\sum_j X_{i,j}]$ is at most the degree of i, which is at most c. (Technically, the $X_{i,i}$ term creates a self-loop, so the correct upper bound is $c+1$. For the sake of cleaner expressions, we omit the additive $+1$ in this sketch.)

The expected number of triangles is at least Δn. Plugging these bounds in:

$$\Delta n \leq L^2 c^2 n \implies L \geq \sqrt{\Delta}/c \tag{5.4}$$

Thus, the vectors have length at least $\sqrt{\Delta}/c$. Now, we lower bound the rank of V. It will be convenient to deal with the Gram matrix $M = V^T V$, which has the same rank as V. Observe that $M_{i,j} = \vec{v}_i \cdot \vec{v}_j \leq L^2$. We will use the following lemma stated first by Swanapoel, but has appeared in numerous forms previously [Swa14].

Lemma 5.1.1 (Rank lemma)**.** *Consider any square matrix $M \in \mathbb{R}^{n \times n}$. Then*

$$\text{rank}(M) \geq \frac{|\sum_i M_{i,i}|^2}{\left(\sum_i \sum_j |M_{i,j}|^2\right)}$$

Note that $M_{i,i} = \vec{v}_i \cdot \vec{v}_i = L^2$, so the numerator $|\sum_i M_{i,i}|^2 = n^2 L^4$. The denominator requires more work. We split it into two terms.

$$\sum_{\substack{i,j \\ \vec{v}_i \cdot \vec{v}_j \leq 1}} (\vec{v}_i \cdot \vec{v}_j)^2 \leq \sum_{\substack{i,j \\ \vec{v}_i \cdot \vec{v}_j \leq 1}} \vec{v}_i \cdot \vec{v}_j \leq cn \tag{5.5}$$

If for $i \neq j$, $\vec{v}_i \cdot \vec{v}_j > 1$, then (i,j) is an edge with probability 1. Thus, there can be at most $(c-1)n$ such pairs. Overall, there are at most cn pairs such that $\vec{v}_i \cdot \vec{v}_j > 1$. So, $\sum_{\substack{i,j \\ \vec{v}_i \cdot \vec{v}_j > 1}} (\vec{v}_i \cdot \vec{v}_j) \leq cnL^4$. Overall, we lower bound the denominator in the rank lemma by $cn(L^4+1)$.

We plug these bounds into the rank lemma. We use the fact that $f(x) = x/(1+x)$ is decreasing for positive x, and that $L \geq \sqrt{\Delta}/c$.

$$\text{rank}(M) \geq \frac{n^2 L^4}{cn(L^4+1)} \geq \frac{n}{c} \cdot \frac{\Delta^2/c^4}{\Delta^2/c^4 + 1} = \frac{\Delta^2}{c(\Delta^2 + c^4)} \cdot n$$

5.1.1 Dealing with varying lengths

The math behind (5.4) still holds with the right approximations. Intuitively, the existence of at least Δn triangles implies that a sufficiently large number of vectors have length at least $\sqrt{\Delta}/c$. On the other hand, these long vectors need to be "sufficiently far away" to ensure that the vertex degrees remain low. There are many such long vectors, and they can can only be far away when their dimension/rank is sufficiently high.

The rank lemma is the main technical tool that formalizes this intuition. When vectors are of varying length, the primary obstacle is the presence of extremely long vectors that create triangles. The numerator in the rank lemma sums $M_{i,i}$, which is the length of the vectors. A small set of extremely long vectors could dominate the sum, increasing the numerator. In that case, we do not get a meaningful rank bound.

But, because the vectors inhabit low-dimensional space, the long vectors from *different* clusters interact with each other. We prove a "packing" lemma (Lemma 5.2.5) showing that there must be many large positive dot products among a set of extremely long vectors. Thus, many of the corresponding vertices have large degree, and triangles incident to these vertices do not contribute to low degree triangles. Operationally, the main proof uses the packing lemma to show that there are few long vectors. These can be removed without affecting the low degree structure. One can then perform a binning (or "rounding") of the lengths of the remaining vectors, to implement the proof described in the above section.

5.2 Proof of Theorem 5.0.2

For convenience, we restate the setting. Consider a set of vectors $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n \in \mathbb{R}^d$, that represent the vertices of a social network. We will also use the matrix $V \in \mathbb{R}^{d \times n}$ for these vectors, where each column is one of the $\vec{v}_i$s. Abusing notation, we will use V to represent both the set of vectors as well as the matrix. We will refer to the vertices by the index in $[n]$.

Let $\mathcal{G}_V$ denote the following distribution of graphs over the vertex set $[n]$. For each index pair i, j, independently insert (undirected) edge (i, j) with probability $\max(0, \min(\vec{v}_i \cdot \vec{v}_j, 1))$.

5.2.1 The basic tools

We now state some results that will be used in the final proof.

Lemma 5.2.1. *[Rank lemma [Swa14]] Consider any square matrix $A \in \mathbb{R}^{n \times n}$. Then*

$$|\sum_i A_{i,i}|^2 \leq \text{rank}(A) \left(\sum_i \sum_j |A_{i,j}|^2 \right)$$

Lemma 5.2.2. *Consider a set of s vectors $\vec{w}_1, \vec{w}_2, \ldots, \vec{w}_s$ in $\mathbb{R}^d$.*

$$\sum_{\substack{(i,j)\in[s]\times[s]\\ \vec{w}_i\cdot\vec{w}_j<0}} |\vec{w}_i \cdot \vec{w}_j| \leq \sum_{\substack{(i,j)\in[s]\times[s]\\ \vec{w}_i\cdot\vec{w}_j>0}} |\vec{w}_i \cdot \vec{w}_j|$$

Proof. Note that $(\sum_{i\leq s} \vec{w}_i) \cdot (\sum_{i\leq s} \vec{w}_i) \geq 0$. Expand and rearrange to complete the proof. □

Recall that an independent set is a collection of vertices that induce no edge.

Lemma 5.2.3. *Any graph with h vertices and maximum degree b has an independent set of at least $h/(b+1)$.*

Proof. Intuitively, one can incrementally build an independent set, by adding one vertex to the set, and removing at most $b+1$ vertices from the graph. This process can be done at least $h/(b+1)$ times.

Formally, we prove by induction on h. First we show the base case. If $h \leq b+1$, then the statement is trivially true. (There is always an independent set of size 1.) For the induction step, let us construct an independent set of the desired size. Pick an arbitrary vertex x and add it to the independent set. Remove x and all of its neighbors. By the induction hypothesis, the remaining graph has an independent set of size at least $(h-b-1)/(b+1) = h/(b+1) - 1$. □

Claim 5.2.4. *Consider the distribution $\mathcal{G}_V$. Let D_i denote the degree of vertex $i \in [n]$. $\mathbf{E}[D_i^2] \leq \mathbf{E}[D_i] + \mathbf{E}[D_i]^2$.*

Proof. (of Claim 5.2.4) Fix any vertex $i \in [n]$. Observe that $D_i = \sum_{j\neq i} X_j$, where X_j is the indicator random variable for edge (i,j) being present. Furthermore, all the X_js are independent.

$$\begin{aligned}
\mathbf{E}[D_i^2] &= \mathbf{E}[(\sum_{j\neq i} X_j)^2] = \mathbf{E}[\sum_{j\neq i} X_j^2 + 2\sum_{j\neq j'} X_j X_{j'}] \\
&= \mathbf{E}[\sum_{j\neq i} X_j] + 2\sum_{j\neq j'} \mathbf{E}[X_j]\mathbf{E}[X_{j'}] \\
&\leq \mathbf{E}[D_i] + (\sum_{j\neq i} \mathbf{E}[X_j])^2 = \mathbf{E}[D_i] + \mathbf{E}[D_i]^2
\end{aligned}$$

□

A key component of dealing with arbitrary length vectors is the following dot product lemma. This is inspired by results of Alon [Alo03] and Tao [Tao13], who get a stronger lower bound of $1/\sqrt{d}$ for *absolute values* of the dot products.

Lemma 5.2.5. *Consider any set of $4d$ unit vectors $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_{4d}$ in $\mathbb{R}^d$. There exists some $i \neq j$ such that $\vec{u}_i \cdot \vec{u}_j \geq 1/4d$.*

Proof. (of Lemma 5.2.5) We prove by contradiction, so assume $\forall i \neq j, \vec{u}_i \cdot \vec{u}_j < 1/4d$. We partition the set $[4d] \times [4d]$ into $\mathcal{N} = \{(i,j) | \vec{u}_i \cdot \vec{u}_j < 0\}$ and $\mathcal{P} = \{(i,j) | \vec{u}_i \cdot \vec{u}_j \geq 0\}$. The proof goes by

providing (inconsistent) upper and lower bounds for $\sum_{(i,j)\in\mathcal{N}} |\vec{u}_i \cdot \vec{u}_j|^2$. First, we upper bound $\sum_{(i,j)\in\mathcal{N}} |\vec{u}_i \cdot \vec{u}_j|^2$ by:

$$\begin{aligned} &\leq \sum_{(i,j)\in\mathcal{N}} |\vec{u}_i \cdot \vec{u}_j| \qquad (\vec{u}_i\text{s are unit vectors}) \\ &\leq \sum_{i\leq 4d} \|\vec{u}_i\|_2^2 + \sum_{\substack{1\leq i\neq j\leq 4d \\ (i,j)\in\mathcal{P}}} |\vec{u}_i \cdot \vec{u}_j| \qquad (\text{Lemma } 5.2.2) \\ &< 4d + 16d^2/4d = 8d \quad (\text{since } \vec{u}_i \cdot \vec{u}_j < 1/4d) \end{aligned} \tag{5.6}$$

For the lower bound, we invoke the rank bound of Lemma 5.2.1 on the $4d \times 4d$ Gram matrix M of $\vec{u}_1, \ldots, \vec{u}_{4d}$. Note that $\text{rank}(M) \leq d$, $M_{i,i} = 1$, and $M_{i,j} = \vec{u}_i \cdot \vec{u}_j$. By Lemma 5.2.1, $\sum_{(i,j)\in[4d]\times[4d]} |\vec{u}_i \cdot \vec{u}_j|^2 \geq (4d)^2/d = 16d$. We bound

$$\begin{aligned} \sum_{(i,j)\in\mathcal{P}} |\vec{u}_i \cdot \vec{u}_j|^2 &= \sum_{i\leq 4d} \|\vec{u}_i\|_2^2 + \sum_{(i,j)\in\mathcal{P}, i\neq j} |\vec{u}_i \cdot \vec{u}_j|^2 \\ &\leq 4d + (4d)^2/(4d)^2 \leq 5d \end{aligned} \tag{5.7}$$

Thus, $\sum_{(i,j)\in\mathcal{N}} |\vec{u}_i \cdot \vec{u}_j|^2 \geq 16d - 5d = 11d$. This contradicts the bound of (5.6).

□

5.2.2 The main argument

We prove by contradiction. We assume that the expected number of triangles contained in the set of vertices of expected degree at most c, is at least Δn. We remind the reader that n is the total number of vertices. For convenience, we simply remove the vectors corresponding to vertices with expected degree at least c. Let $\hat{V}$ be the matrix of the remaining vectors, and we focus on $\mathcal{G}_{\hat{V}}$. The expected number of triangles in $G \sim \mathcal{G}_{\hat{V}}$ is at least Δn.

The overall proof can be thought of in three parts.

Part 1, remove extremely long vectors: Our final aim is to use the rank lemma (Lemma 5.2.1) to lower bound the rank of V. The first problem we encounter is that extremely long vectors can dominate the expressions in the rank lemma, and we do not get useful bounds. We show that the number of such long vectors is extremely small, and they can removed without affecting too many triangles. In addition, we can also remove extremely small vectors, since they cannot participate in many triangles.

Part 2, find a "core" of sufficiently long vectors that contains enough triangles: The previous step gets a "cleaned" set of vectors. Now, we bucket these vectors by length. We show that there is a large bucket, with vectors that are sufficiently long, such that there are enough triangle contained in this bucket.

Part 3, apply the rank lemma to the "core": We now focus on this core of vectors, where the rank lemma can be applied. At this stage, the mathematics shown in Section 5.1 can be carried out almost directly.

Now for the formal proof. For the sake of contradiction, we assume that $d = \operatorname{rank}(\hat{V}) < \alpha(\Delta^4/c^9) \cdot n/\lg^2 n$ (for some sufficiently small constant $\alpha > 0$).

Part 1: Removing extremely long (and extremely short) vectors

We begin by showing that there cannot be many long vectors in $\hat{V}$.

Lemma 5.2.6. *There are at most $5cd$ vectors of length at least $2\sqrt{n}$.*

Proof. Let $\mathcal{L}$ be the set of "long" vectors, those with length at least $2\sqrt{n}$. Let us prove by contradiction, so assume there are more than $5cd$ long vectors. Consider a graph $H = (\mathcal{L}, E)$, where vectors $\vec{v_i}, \vec{v_j} \in \mathcal{L}$ $(i \neq j)$ are connected by an edge if $\frac{\vec{v_i}}{\|\vec{v_i}\|_2} \cdot \frac{\vec{v_j}}{\|\vec{v_j}\|_2} \geq 1/4n$. We choose the $1/4n$ bound to ensure that all edges in H are edges in G.

Formally, for any edge (i,j) in H, $\vec{v_i} \cdot \vec{v_j} \geq \|\vec{v_i}\|_2 \|\vec{v_j}\|_2/4n \geq (2\sqrt{n})^2/4n = 1$. So (i,j) is an edge with probability 1 in $G \sim \mathcal{G}_V$. The degree of any vertex in H is at most c. By Lemma 5.2.3, H contains an independent set I of size at least $5cd/(c+1) \geq 4d$. Consider an arbitrary sequence of $4d$ (normalized) vectors in I $\vec{u}_1, \ldots, \vec{u}_{4d}$. Applying Lemma 5.2.5 to this sequence, we deduce the existence of (i,j) in I $(i \neq j)$ such that $\frac{\vec{v_i}}{\|\vec{v_i}\|_2} \cdot \frac{\vec{v_j}}{\|\vec{v_j}\|_2} \geq 1/4d \geq 1/4n$. Then, the edge (i,j) should be present in H, contradicting the fact that I is an independent set. □

Denote by V' the set of all vectors in $\hat{V}$ with length in the range $[n^{-2}, 2\sqrt{n}]$.

Claim 5.2.7. *The expected degree of every vertex in $G \sim \mathcal{G}_{V'}$ is at most c, and the expected number of triangles in G is at least $\Delta n/2$.*

Proof. Since removal of vectors can only decrease the degree, the expected degree of every vertex in $\mathcal{G}_{V'}$ is naturally at most c. It remains to bound the expected number of triangles in $G \sim \mathcal{G}_{V'}$. By removing vectors in $V \setminus V'$, we potentially lose some triangles. Let us categorize them into those that involve at least one "long" vector (length $\geq 2\sqrt{n}$) and those that involve at least one "short" vector (length $\leq n^{-2}$) but no long vector.

We start with the first type. By Lemma 5.2.6, there are at most $5cd$ long vectors. For any vertex, the expected number of triangles incident to that vertex is at most the expected square of the degree. By Claim 5.2.4, the expected degree squares is at most $c + c^2 \leq 2c^2$. Thus, the expected total number of triangles of the first type is at most $5cd \times 2c^2 \leq \Delta n/\lg^2 n$.

Consider any triple of vectors $(\vec{u}, \vec{v}, \vec{w})$ where $\vec{u}$ is short and neither of the others are long. The probability that this triple forms a triangle is at most

$$\begin{aligned} & \min(\vec{u} \cdot \vec{v}, 1) \cdot \min(\vec{u} \cdot \vec{w}, 1) \\ \leq\ & \min(\|\vec{u}\|_2 \|\vec{v}\|_2, 1) \cdot \min(\|\vec{u}\|_2 \|\vec{w}\|_2, 1) \\ \leq\ & (n^{-2} \cdot 2\sqrt{n})^2 \leq 4n^{-3} \end{aligned}$$

Summing over all such triples, the expected number of such triangles is at most 4.

Thus, the expected number of triangles in $G \sim \mathcal{G}_{V'}$ is at least $\Delta n - \Delta n/\lg^2 n - 4 \geq \Delta n/2$. □

Part 2: Finding core of sufficiently long vectors with enough triangles

For any integer r, let V_r be the set of vectors $\{\vec{v} \in V' | \|\vec{v}\|_2 \in [2^r, 2^{r+1})\}$. Observe that the V_rs form a partition of V'. Since all lengths in V' are in the range $[n^{-2}, 2\sqrt{n}]$, there are at most $3 \lg n$ non-empty V_rs. Let R be the set of indices r such that $|V_r| \geq (\Delta/60c^2)(n/\lg n)$. Furthermore, let V'' be $\bigcup_{r \in R} V_r$.

Claim 5.2.8. *The expected number of triangles in $G \sim \mathcal{G}_{V''}$ is at least $\Delta n/8$.*

Proof. The total number of vectors in $\bigcup_{r \notin R} V_r$ is at most $3 \lg n \times (\Delta/60c^2)(n/\lg n) \leq (\Delta/20c^2)n$. By Claim 5.2.4 and linearity of expectation, the expected sum of squares of degrees of all vectors in $\bigcup_{r \notin R} V_r$ is at most $(d + c^2) \times (\Delta/20c^2)n \leq \Delta n/10$. Since the expected number of triangles in $G \sim \mathcal{G}_{V'}$ is at least $\Delta n/2$ (Claim 5.2.7) and the expected number of triangles incident to vectors in $V' \setminus V''$ is at most $\Delta n/10$, the expected number of triangles in $G \sim \mathcal{G}_{V''}$ is at least $\Delta n/2 - \Delta n/10 \geq \Delta n/8$. □

We now come to an important claim. Because the expected number of triangles in $G \sim \mathcal{G}_{V''}$ is large, we can prove that V'' must contain vectors of at least constant length.

Claim 5.2.9. $\max_{r \in R} 2^r \geq \sqrt{\Delta}/4c$.

Proof. Suppose not. Then every vector in V'' has length at most $\sqrt{\Delta}/4c$. By Cauchy-Schwartz, for every pair $\vec{u}, \vec{v} \in V''$, $\vec{u} \cdot \vec{v} \leq \Delta/16c^2$. Let I denote the set of vector indices in V'' (this corresponds to the vertices in $G \sim \mathcal{G}_{V''}$). For any two vertices $i \neq j \in I$, let $X_{i,j}$ be the indicator random variable for edge (i, j) being present. The expected number of triangles incident to vertex i in $G \sim \mathcal{G}_{V''}$ is

$$\mathbf{E}[\sum_{j \neq k \in I} X_{i,j} X_{i,k} X_{j,k}] = \sum_{j \neq k \in I} \mathbf{E}[X_{i,j} X_{i,k}] \mathbf{E}[X_{j,k}]$$

Observe that $\mathbf{E}[X_{j,k}]$ is at most $\vec{v_j} \cdot \vec{v_k} \leq \Delta/16c^2$. Furthermore,

$$\sum_{j \neq k \in I} \mathbf{E}[X_{i,j} X_{i,k}] = \mathbf{E}[D_i^2]$$

(recall that D_i is the degree of vertex i.) By Claim 5.2.4, this is at most $c + c^2 \leq 2c^2$. The expected number of triangles in $G \sim \mathcal{G}_{V''}$ is at most $n \times 2c^2 \times \Delta/16c^2 = \Delta n/8$. This contradicts Claim 5.2.8. □

Part 3: Applying the rank lemma to the core

We are ready to apply the rank bound of Lemma 5.2.1 to prove the final result. The following lemma contradicts our initial bound on the rank d, completing the proof. We will omit some details in the following proof, and provide a full proof in the SI.

Lemma 5.2.10. $\operatorname{rank}(V'') \geq (\alpha\Delta^4/c^9)n/\lg^2 n$.

Proof. It is convenient to denote the index set of V'' be I. Let M be the Gram Matrix $(V'')^T(V'')$, so for $i,j \in I$, $M_{i,j} = \vec{v_i} \cdot \vec{v_j}$ By Lemma 5.2.1, $\operatorname{rank}(V'') = \operatorname{rank}(M) \geq (\sum_{i\in I} M_{i,i})^2/\sum_{i,j\in I}|M_{i,j}|^2$. Note that $M_{i,i}$ is $\|\vec{v_i}\|_2^2$, which is at least 2^{2r} for $\vec{v_i} \in V_r$. Let us denote $\max_{r\in R} 2^r$ by L, so all vectors in V'' have length at most $2L$. By Cauchy-Schwartz, all entries in M are at most $4L^2$.

We lower bound the numerator.

$$\begin{aligned}
&\Big(\sum_{i\in I}\|\vec{v_i}\|_2^2\Big)^2 \geq \Big(\sum_{r\in R} 2^{2r}|V_r|\Big)^2 \\
\geq\ &\Big(\max_{r\in R} 2^{2r}(\Delta/60c^2)(n/\lg n)\Big)^2 \\
=\ &L^4(\Delta^2/3600c^4)(n^2/\lg^2 n)
\end{aligned}$$

Now for the denominator. We split the sum into four parts and bound each separately.

$$\begin{aligned}
\sum_{i,j\in I}|M_{i,j}|^2 \ =\ &\sum_{i\in I}|M_{i,i}|^2 + \sum_{\substack{i,j\in I\\ i\neq j, M_{i,j}\in[0,1]}}|M_{i,j}|^2 \\
&+ \sum_{\substack{i,j\in I\\ i\neq j, M_{i,j}>1}}|M_{i,j}|^2 + \sum_{\substack{i,j\in I\\ M_{i,j}<0}}|M_{i,j}|^2 :!
\end{aligned}$$

Since $|M_{i,i}| \leq L^2$, the first term is at most $4nL^4$. For $i \neq j$ and $M_{i,j} \in [0,1]$, the probability that edge (i,j) is present is precisely $M_{i,j}$. Thus, for the second term,

$$\sum_{\substack{i,j\in I\\ i\neq j, M_{i,j}\in[0,1]}}|M_{i,j}|^2 \leq \sum_{\substack{i,j\in I\\ i\neq j, M_{i,j}\in[0,1]}} M_{i,j} \leq 2cn \tag{5.8}$$

For the third term, we observe that when $M_{i,j} > 1$ (for $i \neq j$), then (i,j) is an edge with probability 1. There can be at most $2cn$ pairs (i,j), $i \neq j$, such that $M_{i,j} > 1$. Thus, the third term is at most $2cn \cdot (4L^2)^2 = 32cnL^4$.

Now for the fourth term. Note that M is a Gram matrix, so we can invoke Lemma 5.2.2 on its entries.

$$\begin{aligned}
\sum_{\substack{i,j\in I\\ M_{i,j}<0}}|M_{i,j}|^2 \ &\leq\ L^2\sum_{\substack{i,j\in I\\ M_{i,j}<0}}|M_{i,j}| \\
&\leq\ L^2\Big(\sum_{i\in I}|M_{i,i}| + \sum_{\substack{i,j\in I\\ M_{i,j}>0}}|M_{i,j}|\Big) \\
&\leq\ 4nL^4 + L^2\sum_{\substack{i,j\in I\\ M_{i,j}\in[0,1]}}|M_{i,j}| + 4L^4\sum_{\substack{i,j\in I\\ M_{i,j}>1}}1 \\
&\leq\ 4nL^4 + 2cnL^2 + 8cnL^4
\end{aligned} \tag{5.9}$$

Putting all the bounds together, we get that $\sum_{i,j\in I}|M_{i,j}|^2 \leq n(4L^4+2c+32cL^4+4L^4+2cL^2+8cL^4) \leq 32n(L^4+c(1+L^2+L^4))$. If $L \leq 1$, we can upper bound by $128cn$. If $L \geq 1$, we can upper bound by $128cnL^4$. In either case, $128cn(1+L^4)$ is a valid upper bound.

Crucially, by Claim 5.2.9, $L \geq \sqrt{\Delta}/4c$. Thus, $4^4c^4L^4/\Delta^2 \geq 1$. Combining all the bounds (and setting $\alpha < 1/(128 \cdot 3600 \cdot 4^4)$),

$$\begin{aligned}\text{rank}(V'') &\geq \frac{L^4(\Delta^2/3600c^4)(n^2/\lg^2 n)}{128cn(1+16L^4)} \\ &\geq \frac{L^4(\Delta^2/3600c^4)(n/\lg^2 n)}{128cn(4^4c^4L^4/\Delta^2+16L^4)} \\ &\geq (\alpha\Delta^4/c^9)(n/\lg^2 n)\end{aligned}$$

□

5.3 Details of empirical results

Data Availability: The datasets used are summarized in Tab. 5.2. We present here four publicly available datasets from different domains. The `ca-HepPh` is a co-authorship network `Facebook` is a social network, `cit-HepPh` is a citation network, all obtained from the SNAP graph database [SNA19]. The `String_hs` dataset is a protein-protein-interaction network obtained from [str19]. (The citations provide the link to obtain the corresponding datasets.)

We first describe the primary experiment, used to validate Theorem 5.0.2 on the SVD embedding. We generated a d-dimensional embedding for various values of d using the SVD. Let G be a graph with the $n \times n$ (symmetric) adjacency matrix A, with eigendecomposition $\Psi\Lambda\Psi^T$. Let Λ_d be the matrix with the $d \times d$ diagonal matrix with the d largest magnitude eigenvalues of A along the diagonal. Let Ψ_d be the $n \times d$ matrix with the corresponding eigenvectors as columns. We compute the matrix $A_d = \Psi_d\Lambda_d\Psi_d^T$ and refer to this as the d spectral embedding of G. This is the standard PCA approach.

From the spectral embeddings, we generate a graph from A_d by considering every pair of vertices (i,j) and generate a random value in $[0,1]$. If the $(i,j)^{\text{th}}$ entry of A_d is greater than the random value generated, the edge is added to the graph. Otherwise the edge is not present. This is the same as taking A_d and setting all negative values to 0, and all values greater than 1 to 1 and performing Bernoulli trials for each edge with the resulting probabilities. In all the figures, this is referred to as the "SVD TDP" (truncated dot product) embedding.

5.3.1 Triangle distributions

To generate Figure 5.1 and Figure 5.3, we calculated the number of triangles incident to vertices of different degrees in both the original graphs and the graphs generated from the embeddings. Each of the plots shows the number of triangles in the graph on the vertical axis

Dataset name	Network type	Number of nodes	Number of edges
Facebook [SNA19]	Social network	4K	88K
cit-HePh [Arn19]	Citation	34K	421K
String_hs [str19]	PPI	19K	5.6M
ca-HepPh [SNA19]	Co-authorship	12K	118M

Figure 5.2: Table of datasets used

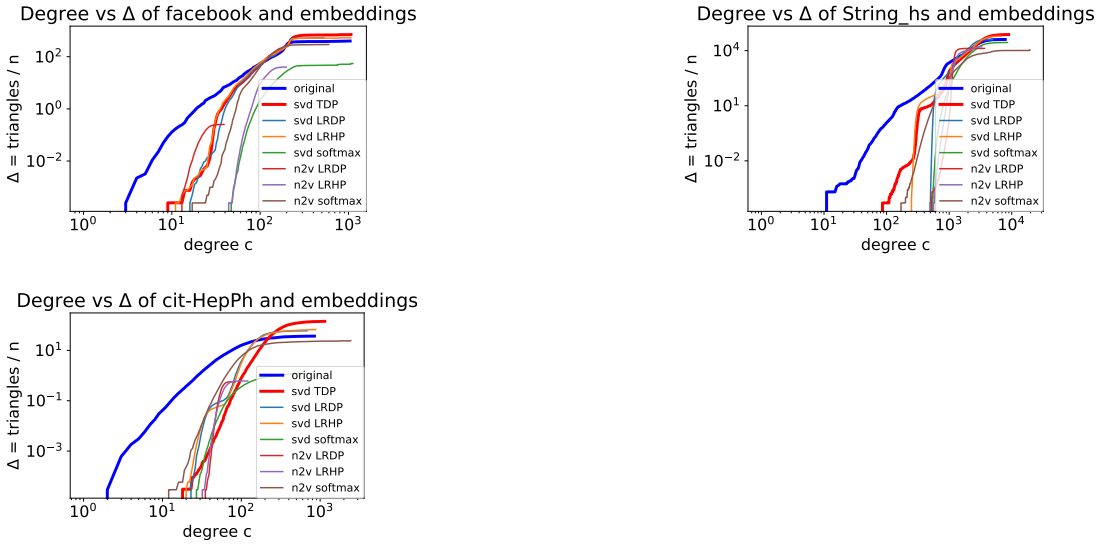

Figure 5.3: Plots of degree c vs Δ: For each network, we plot c versus the total number of triangles only involving vertices of degree at most c. We divide the latter by the number of vertices, so it corresponds to Δ, as in the main definition. In each subfigure, we plot these both for the original graph, and the maximum Δ in a set of 100 samples from a 100-dimensional embedding. Observe how the embeddings generate graphs with very few triangles among low degree vertices. The gap in Δ for low degree is 2-3 orders of magnitude in all instances.

and the degrees of vertices on the horizontal axis. Each curve corresponds to some graph, and each point (x, y) in a given curve shows that the graph contains y triangles if we remove all vertices with degree at least x. We then generate 100 random samples from the 100-dimensional embedding, as given by SVD (described above). For each value of c, we plot the maximum value of Δ over all the samples. This is to ensure that our results are not affected by statistical variation (which was quite minimal).

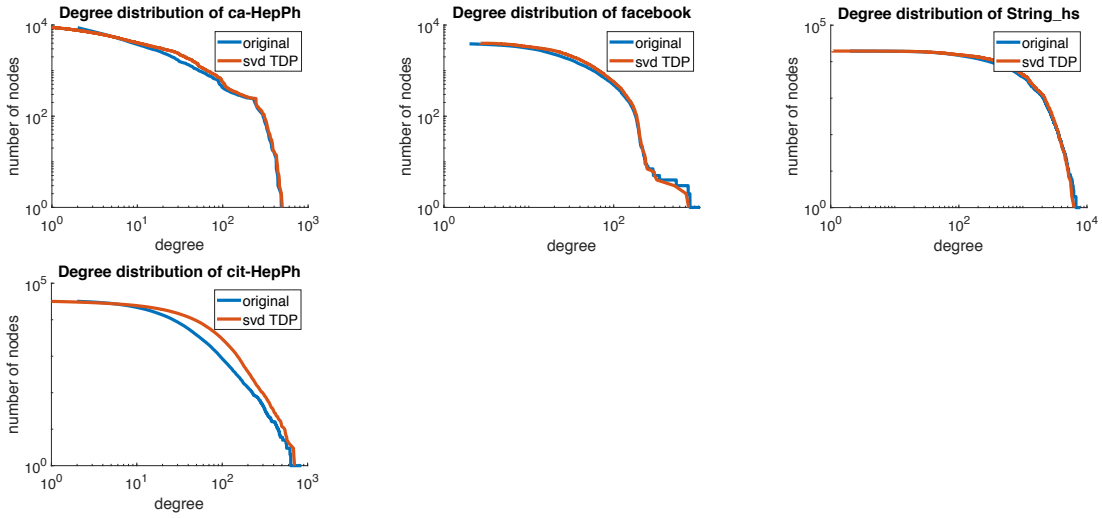

Figure 5.4: Plots of degree distributions: For each network, we plot the true degree distribution vs the expected degree distribution of a 100-dimensional embedding. Observe how the embedding does capture the degree distribution quite accurately at all scales.

5.3.2 Alternate graph models

We consider three other functions of the dot product, to construct graph distributions from the vector embeddings. Details on parameter settings and the procedure used for the optimization are given in the SI.

Logistic Regression on the Dot Product (LRDP): We consider the probability of an edge (i, j) to be the logistic function $L(1 + \exp(-k(\vec{v}_i \cdot \vec{v}_j - x_0)))^{-1}$, where L, k, x_0 are parameters. Observe that the range of this function is $[0, 1]$, and hence can be interpreted as a probability. We tune these parameters to fit the expected number of edges, to the true number of edges. Then, we proceed as in the TDP experiment. We note that the TDP can be approximated by a logistic function, and thus the LRDP embedding is a "closer fit" to the graph than the TDP embedding.

Logistic Regression on the Hadamard Product (LRHP): This is inspired by linear models used on low-dimensional embeddings [HYL17b]. Define the Hadamard product $\vec{v}_i \odot \vec{v}_j$ to be the d-dimensional vector where the rth coordinate is the product of rth coordinates. We now fit a logistic function over linear functions of (the coordinates of) $\vec{v}_i \odot \vec{v}_j$. This is a significantly richer model than the previous model, which uses a fixed linear function (sum). Again, we tune parameters to match the number of edges. The fitting of LRDP and LRHP was done using the Matlab function `glmfit` (Generalized Linear Model Regression Fit) [mat]. The distribution parameter was set to "binomial", since the total number of edges is distributed as a weighted binomial.

Softmax: This is inspired by low-dimensional embeddings for random walk matrices [PARS14, GL16b]. The idea is to make the probability of edge (i, j) to be proportional to softmax, $\exp(\vec{v}_i \cdot \vec{v}_j)/\sum_{k\in[n]} \vec{v}_i \cdot \vec{v}_k$. This tends to push edge formation even for slightly higher dot products, and one might imagine this helps triangle formation. We set the proportionality constant separately for each vertex to ensure that the expected degree is the true degree. The probability matrix is technically undirected, but we symmetrize the matrix.

node2vec experiments: We also applied NODE2VEC, a recent deep learning based graph embedding method [GL16b], to generate vector representations of the vertices. We used the optimized C++ implementation [n2va] for NODE2VEC, which is equivalent to the original implementation provided by the authors [n2vb]. For all our experiments, we use the default settings of walk length of 80, 10 walks per node, p=1 and q=1. The NODE2VEC algorithm tries to model the random walk matrix associated with a graph, not the raw adjacency matrix. The dot products between the output vectors $\vec{v}_i \cdot \vec{v}_j$ are used to model the random walk probability of going from i to j, rather than the presence of an edge. It does not make sense to apply the TDP function on these dot products, since this will generate (in expectation) only n edges (one for each vertex). We apply the LRDP or LRHP functions, which use the NODE2VEC vectors as inputs to a machine learning model that predicts edges.

In Figures 5.1 and 5.3, we show results for all the datasets. We note that for all datasets and all embeddings, the models fail to capture the low-degree triangle behavior.

5.3.3 Degree distributions

We observe that the low-dimensional embeddings obtained from SVD and the truncated dot product can capture the degree distribution accurately. In Figure 5.4, we plot the degree distribution (in loglog scale) of the original graph with the expected degree distribution of the embedding. For each vertex i, we can compute its expected degree by the sum $\sum_i p_{ij}$, where p_{ij} is the probability of the edge (i, j). In all cases, the expected degree distributions is close to the true degree distributions, even for lower degree vertices. The embedding successfully captures the "first order" connections (degrees), but not the higher order connections (triangles). We believe that this reinforces the need to look at the triangle structure to discover the weaknesses of low-dimensional embeddings.

5.3.4 Detailed relationship between rank and triangle structure

For the smallest `Facebook` graph, we were able to compute the entire set of eigenvalues. This allows us to determine how large a rank is required to recreate the low-degree triangle structure. In Figure 5.5, for varying rank of the embedding, we plot the corresponding triangle distribution. In this plot, we choose the embedding given by the eigendecomposition (rather than SVD), since it is guaranteed to converge to the correct triangle distribution for an n-dimensional

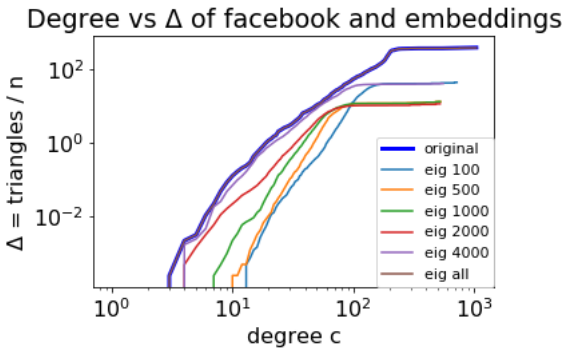

Figure 5.5: Plots of degree c vs Δ, for varying rank: For the `Facebook` social network, for varying rank of embedding, we plot c versus the total number of triangles only involving vertices of degree at most c. The embedding is generated by taking the top eigenvectors. Observe how even a rank of 2000 does not suffice to match the true triangle values for low degree.

embedding (n is the number of vertices). The SVD and eigendecomposition are mostly identical for large singular/eigenvalues, but tend to be different (up to a sign) for negative eigenvalues.

We observe that even a 1000 dimensional embedding does not capture the c vs Δ plots for low degree. Even the rank 2000 embedding is off the true values, though it is correct to within an order of magnitude. This is strong corroboration of our main theorem, which says that near linear rank is needed to match the low-degree triangle structure.

Chapter 6

On the (in)-effectiveness of matrix factorization-based graph embedding methods for community labeling

Graph structured data is ubiquitous. Capturing the graph structure is important for a wide variety of machine learning tasks, such as ranking in social networks, content recommendations, and clustering [EK10b]. A long-studied challenge for building such machine learned models has been to capture the graph structure for use in a variety of modeling tasks. *Graph representation learning*, or *low-dimensional graph embeddings*, provide a convenient solution to this problem. Given a graph G on n vertices, these methods map each vertex to a vector in $\mathbb{R}^d$, where $d \ll n$, in an unsupervised or a self-supervised manner (it is also sometimes referred to as a pre-training procedure). Typically, the goal of the embedding is to represent graph proximity by (a function of) the dot product of vectors, thereby implicitly giving a geometric representation of the graph.[1] The dot product formulation provides a convenient form for building a models (e.g. using deep learning). Moreover, the geometry of the embedding allows efficient reverse-index lookups, using nearest neighbor search. Thus, these embedding vectors are not only used as features for many downstream tasks (such as link prediction, community detection, and clustering), but also used for reverse lookups using nearest neighbor search [CAS16, Twi18].

The study of low-dimensional graph embeddings is an incredibly popular research

[1] Since there is a wide range of methods for Graph representation learning, we refer the reader to the "Shallow embeddings" class in a recent survey [CAEHP+20] for a more comprehensive overview.

area, and has generated many exciting results over the past few years (see surveys [HYL18, CAEHP+20] and a Chapter 23 in [Mur21]). Nonetheless, there is limited principled understanding of the power of low-dimensional embeddings (a few recent papers address this topic [SSSG20b, CMST20, Lou20, GJJ20]). Our work aims to understand the effectiveness of a class of graph embeddings in preserving graph structure as it manifests in performance on different downstream tasks.

Due to an explosion of interest in the area, there are by now a large class of graph embedding methods [Mur21]. For the sake of a principled study, we focus on the important class of *unsupervised low-rank factorization methods*. While there do exist many methods outside this class, such factorization methods cover a number of popular and influential embeddings methods, including GraRep [OCP+16], DeepWalk [PARS14], and Node2Vec [GL16a]. In fact, a recent result shows that many existing embedding techniques can be recast as matrix factorization methods [QDM+18]. One begins with an $n \times n$ promixity matrix M, typically the adjacency matrix, the random walk matrix, or some variant thereof (e.g. Node2Vec uses the matrix for a certain second order random walk). Using optimization techniques, the matrix M is approximated as a Gramian matrix $V^T V$, where $V \in \mathbb{R}^{d \times n}$. The column vectors of V are the embeddings of the vertices. In direct factorizations, one simply tries to minimize $\|V^T V - M\|_2$. More sophisticated softmax factorizations perform non-linear entry-wise transformations on $V^T V$ to approximate M. This class of unsupervised embedding methods is among the most popular and prevalent low-dimensional graph embeddings, and hence this is a particularly useful class to quantify performance for.

Our aim is to study a natural question, albeit one that is somewhat challenging to pose formally: *to what extent do factorization-based embedding methods capture graph structure relevant to downstream ML tasks?*

To this end, we fix the following well-defined *pairwise community labeling* problem. Given two vertices i and j, the binary classification task is to determine whether they belong to the same community. We note that this community labeling problem is an instance of a broad range of community detection problems that have a long history of study in the graph mining literature [LRU20]. We then attempt a rigorous theoretical and empirical understanding of the performance of factorization-based graph embeddings on community labeling.

6.0.1 Formal description of setting

We formally describe the graph embeddings techniques that are studied in this work. The learned factorization approach is to approximate M by the Gramian matrix $V^T V$ (the matrix of dot products). Typically, this matrix V is found by formulating a machine learning problem, which has a loss function that minimizes a distance/norm between $V^T V$ and M. Broadly speaking, we can classify these methods into two categories:

• **Direct factorizations:** Here, we set V as $\text{argmin}_V \|V^T V - M\|_2$, where M is typically (some power of) the graph adjacency matrix. Methods such as Graph Factorization, GraRep [CLX15], and HOPE [OCP+16] would fall under this category.

• **Softmax factorizations:** These methods factorize a stochastic matrix, such as (powers of) the random walk matrix. Since $V^T V$ is not necessarily stochastic, these methods apply the softmax to generate a stochastic matrix. Notable examples are such methods are DeepWalk [PARS14] and Node2vec [GL16a]. Formally, consider the normalized softmax matrix $\text{nsm}(V)$ given by

$$\text{nsm}(V)_{ij} = \frac{\exp(\vec{v}_i \cdot \vec{v}_j)}{\sum_k \exp(\vec{v}_i \cdot \vec{v}_k)} \tag{6.1}$$

Note that $\text{nsm}(V)$ is stochastic by construction. The objective of the learning problem is to minimize $KL(\text{nsm}(V), M)$, which is the sum of row-wise KL-divergence between the rows (this is equivalent to cross-entropy loss).

The recent `NetMF` [QDM+18] method interpolates between these categories and shows that a number of existing methods can be expressed as factorization methods, especially of the above forms. For this study, we only focus on the above two category of unsupervised embedding methods. (We discuss this choice and other methods in §6.0.3.)

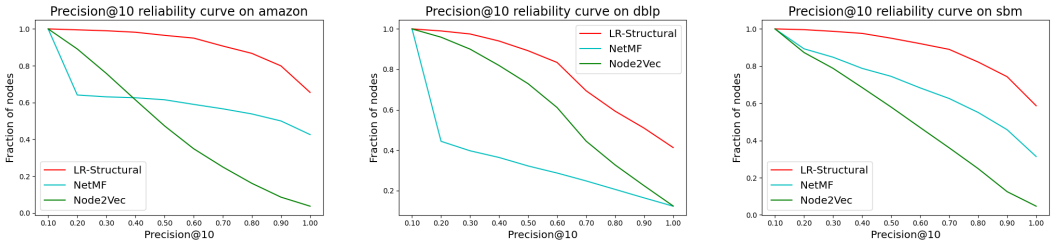

Figure 6.1: Each point, (x, y), on the curve represents the approximate fraction of vertices, y, for which the given method produces a precision@10 score of at least x. `LR-Structural` is plotted against the two best performing embedding methods. 1000 vertices are sampled and for each vertex v sampled, the vertices of the graph $u_1, \ldots, u_n$, are ordered by decreasing score assigned by the given classifier. The precision@10 is the fraction of $u_1, \ldots, u_{10}$ which share a community with v.

Empirical setup: We empirically investigate performance of the above methods on graphs where the ground truth communities correlate well with graph structure. In the case of real data, the ground truth is provided by the existing community labels. With synthetic data, we explicitly construct stochastic block models with well-defined communities. For every pair of vertices i, j, the prediction problem is to determine whether they belong to the same

community (note that they may belong together in multiple communities, but we do not require the community label itself to be determined; just whether they belong in *any* community together).

For both real and simulated data, we note that the ground truth is sparse, i.e. the vast majority of node pairs do not belong to the same community. Hence, it is appropriate to measure the prediction performance using precision-recall curves for this highly imbalanced label distribution [DG06]. Consistent with most of the literature on graph embeddings and the applications that often require nearest neighbor lookups, the main feature we use for prediction is the value of the dot product [CAS16, CAEHP+20].

Theoretical setup: In order to analyze the performance, we provide an abstraction of community structure from a matrix standpoint: this can intuitively be thought of as having many dense blocks in an overall sparse matrix. We then attempt to quantify "how much" community structure can be present in a matrix $V^T V$ or $\text{nsm}(V)$, for *any* matrix $V \in \mathbb{R}^{d \times n}$ (for $d \ll n$). This formulation captures the fundamental notion of a low-rank factorization, without referring to any specific method to compute it. Our results hold for *any* direct/softmax factorization method, regardless of how the embedding is computed. Our formulation theoretically investigates whether it is even possible to recreate community structure using a low rank factorization of the form $V^T V$ or $\text{nsm}(V)$.

6.0.2 Main results

Our main finding is that all of the graph embeddings methods we tested (including `GraRep`, `DeepWalk`, `Node2Vec`, and `NetMF`) perform poorly on the community labeling task, and are handily out-performed by a baseline logistic model `LR-Structural` built using just four classic graph structural features[2]. We observe the same outcome for a series of experiments on real data and synthetic data. Motivated by this empirical finding, we provide a mathematical explanation for this result by providing a theorem which shows that the community structure exhibited by softmax factorizations is unstable under small perturbations of the embedding vectors.

Evaluations on real data: In our experiments, not only do we see poor *absolute* performance on the community labeling task, but a baseline `LR-Structural` based on "classic" graph features handily outperforms the embeddings. We see this difference not only in an overall manner, but across individual nodes in the graph. In particular, Figure 6.1 shows a "reliability plot" for precision@10 for a set of 1000 nodes sampled randomly from each of the graphs. To produce this chart, we first randomly sampled 1000 nodes, each of which has at least 10 neighbors in the same community. Then, each of these vertices selects their top-10 predictions for a same-

[2]The features for a node pair (u, v) are: 1) Personalized-PageRank (PPR) score from u to v, 2) PPR from v to u, 3) cosine similarity among $N(u)$ and $N(v)$ (where $N(\cdot)$ denotes a node's neighborhood), and 4) the size of the cut separating $N(u)$ and $N(v)$

community neighbor using a model, producing a distribution over precision@10. Then for each model, one can produce a reliability plot (see Figure 6.1) that produces points (x, y): for any given value x for precision@10, define y as the fraction of nodes that have precision@10 of at least x. Thus, we'd expect that a highly accurate model would have an almost flat curve. The results in Figure 6.1 show that generally `LR-Structural` can produce high accurate (precision@k ≥ 0.7) community labels for 20–40% more nodes than the embedding-based models. This suggests that the embeddings can retrieve very few of the community neighbors, while classic graph features used in `LR-Structural` (PPR scores, neighborhood similarity) recover much of the community structure.

We emphasize that our empirical analysis is more nuanced than the more common aggregated measurement (such as via the AUC-ROC [CAEHP+20]) as it measures the performance across individual nodes in the graph. We believe this individual measurement is more reflective of our goal, and also, as the label distribution (which pairs co-occur in communities) is highly imbalanced, a P/R-like curve provides a more useful measure than the ROC curves. The latter can be misleading as an ROC curve can still look quite good while misclassifying much of the minority class. [DG06].

For completeness, we also perform experiments with regression models using the Hadamard product of the two vectors. Again, the embeddings gives results of similar quality, showing that linear functions (of the embedding vectors) fail to predict community structure. Refer to Section 6.2 for more details.

Evaluations on SBMs: To further investigate this phenomenon, we also generated synthetic graph instances using Stochastic Block Models (SBMs) that have a very simple planted community structure. For example, we create a graph with $n = 10^5$ vertices and blocks of size 20. The edge density inside a block is 0.3 and we connect the blocks by a sparse Erdős-Rényi graph such that the average number of links within a block is equal to those that go between blocks. We vary the overall edge density (while keeping the ratio between inter-block and intra-block edges constant) and study the precision@10 scores. We observe that while the performance across methods tends to increase with density, `LR-Structural` still outperforms them.

Theoretical explanation: We provide a formulation for what it means for a matrix to exhibit community structure. We emphasize that this is not meant to completely capture the challenging notion of communities (which has a deep and rich history), but rather to give us some formal framework to state our impossibility results. Intuitively, an overall sparse matrix/graph has community structure if a non-trivial fraction of the rows/vertices participate in small dense blocks of entries. We then investigate when $V^T V$ and nsm(V) exhibit community structure. First, we prove that for $d \ll n$, $V^T V$ cannot exhibit community structure, which provides a principled justification for the empirical observation that direct factorization methods perform poorly across all real and synthetic instances.

Interestingly, we also show that despite the negative result for truncated-SVD in [SSSG20b], softmax factorizations *can* exhibit community structure similar to those given by the SBMs discussed earlier. A related observation has been made by [CMST20]. This result suggests that they provide a superior alternative to direct factorizations. On the other hand, we also prove that this community structure is fundamentally unstable under small perturbations of the vectors obtained from softmax factorizations. Meaning, if we take any matrix V such that $\text{nsm}(V)$ has community structure, then with high probability, $\text{nsm}(\widetilde{V})$ does not have such a structure (where $\widetilde{V}$ is a slight random perturbation of $\text{nsm}(V)$).

This strongly suggests that optimization methods cannot produce low dimensional matrices V where $\text{nsm}(V)$ has community structure. This theorem provides a mathematical understanding of the limitations of softmax factorizations.

6.0.3 Justification of Setting & Related Work

The area of graph embeddings has seen a lot of exciting recent progress since a series of papers, DeepWalk [PARS14], LINE [TQW+15b], and Node2Vec [GL16a], showed how to apply deep learning methods developed for NLP (such as Word2Vec [LM14]) for graph embeddings. Given the amount of attention this area has received, we do not cover the entire field, and instead refer the reader to excellent recent surveys for an overview [HYL18, CAEHP+20]. In this section, we discuss justifications for our theoretical and empirical setting, and discuss related work.

Why factorization methods: We note that there are graph embedding methods that are not factorization based (e.g. GraphSage [HYL17a]) as well as factorization methods that do not use direct or softmax factorizations [CMST20], as well as GCNs and GNNs [TW17, CAEHP+20]. For the sake of a principled study, we chose a well-defined subclass of methods that covered many important methods such as `GraRep` [CLX15], `DeepWalk` [PARS14], `Node2Vec` [GL16a], and `NetMF` [QDM+18]. Moreover, the recent `NetMF` algorithm shows how many past methods can be recast as factorization methods.

We do note that these techniques are foundational in that they are widely used, have been influential in motivating new lines of work, and capture the basic premise of mapping a network-based notion of closeness among vertices to geometric closeness.

In particular, `DeepWalk` [PARS14] is the canonical and arguably the most important softmax factorization. It is a seminal result in this area that established a mapping between graph structure and deep learning (specifically, Word2Vec), and has motivated substantial followup work among that line. Among these is `Node2Vec` [GL16a], which is also a widely used method, and forms the core idea for several other embeddings methods. `NetMF` [QDM+18] combines many different algorithms and frames them directly through low-rank matrix factorizations, providing a unifying interpretation for several techniques. These representation learning

methods are all unsupervised (or sometimes called self-supervised), and can be thought of as pre-training methods.

Why community labeling: One central promise of unsupervised graph embedding methods is to preserve network structure in the geometry that can then be useful for downstream tasks. Indeed, there is a rich history of spectral methods dating back several decades that are based on a similar premise [LV99, NJW02, McS01]. In our work, we focus on the task of community labeling both because it is one of the most popular graph mining tasks, and is also directly connected to the core of the question we study here: how well do embeddings preserve the graph structure? This community labeling task also has a rich history of work in the graph mining literature, and we mention only a few salient works here, referring the reader to this survey [FH16]. One of the main findings here has been the usefulness of higher-order graph structure in community detection, as showcased by both theoretical work on Stochastic Block Models [CRV15], and experimental investigations [AL06, LHBH15]. This observation makes the community labeling task an excellent fit for studying the question of how well do graph embeddings capture the graph structure for a downstream task that they are not directly trained for. Moreover, the optimization used to generate embeddings often tries to place similar vertices geometrically closer. Hence, a natural measure of success in preserving structure is to check whether communities are captured by sets of geometrically close vertices. Community labeling is easy to empirically validate with ground truth, so we can do precise studies of the quality of a candidate embedding method.

How we chose our input graphs: Our main question is to what extent embedding methods capture relevant graph structure. Hence, we focus on real and synthetic datasets where the graph structure correlates with community structure. This is measured by the good performance of `LR-Structural` on the community labeling task, since `LR-Structural` uses classic features such as Personalized-PageRank (PPR) scores and cosine similarities of neighborhoods. Our thesis is that if embedding methods cannot perform well on such graphs, then the embedding fails to capture the structure of the graph. We feel that the SBM experiments are quite enlightening, since the underlying community structure is quite simple (and easy to learn through simple structural features).

Other related work: We briefly note a few important representation learning techniques that are beyond the scope of our work. The most prominent among these is the Graph Neural Networks such as [HYL17a, TW17, VCC+18, HLG+20], which can be thought of as a class of learned message passing methods. We refer the reader to a nicely interpretable classification of these methods in a recent survey [CAEHP+20]. The factorization methods we study fall under the "Shallow embeddings" classification there. There are several recent works [GJJ20, Lou20, XHLJ19] that theoretically study the power of GNNs, but this is complementary to our work since we study a different class of methods.

A recent result shows the inability of low-dimensional SVD based embeddings of preserving the triangle structure of real-world networks [SSSG20b]. A followup showed that these impossibility results can be circumvented by alternate embedding methods [CMST20]. Our result can be thought of as a deeper investigation into this issue. First, we look at a class of factorization methods subsuming those used in practice. Secondly, we also focus on a specific downstream ML task, unless previous results that focus solely on the graph structure.

6.1 Mathematical results and interpretation

We define a simple abstraction of community structure in a matrix M. Then, we try to quantify how much community structure Gram matrices and softmax factorizations can possess. We will state these as formal theorems, which are our main mathematical result. The proofs will be given in §5.2.2.

Let us start with an $n \times n$ matrix M that represents the "similarity" or likelihood of connection between vertices. For convenience, let us normalize so that the row sums have absolute value at most one. (So the sum of similarities of a vertex is at most 1.) A communities is essentially a dense block of entries, which motivates the following definition. We use ε to denote a parameter for the threshold of community strength. One should think of ε as a small constant, or something slowly decreasing in n (like $1/\text{poly}(\log n)$).

Definition 6.1.1. *A pair of vertices (i, j) is a* potential community pair *if both M_{ij} and M_{ji} are at least ε.*

Note that we do not expect all such pairs (i, j) to truly be together in a community. Hence, we only consider such a pair a potential candidate. We expect community relationships to be mutual, even if the matrix M is not. A community can be thought of as a submatrix where at least a constant fraction of pairs are potential community pairs. For our purposes, we do not need to further formalize. It is natural to expect that $\Theta(n)$ pairs are community pairs; indeed, most vertices should participate in communities, and will have at least a constant number of community neighbors. Our mathematical analyses shows that direct and softmax factorizations cannot produce these many potential community pairs.

Lower bound for direct factorizations: We first show a strong lower bound for direct factorizations. We prove that the number of potential community pairs in V^TV is linear in the rank, and thus, a low-dimensional factorization cannot capture community structure. The key insight is to use the rotational invariance of Frobenius norms.

Theorem 6.1.2. *Consider any matrix $V \in \mathbb{R}^{d \times n}$ such that row sums in V^TV have absolute value at most 1. Then V has at most $d/2\varepsilon^2$ potential community pairs.*

Proof. Since V^TV has row sums of absolute value at most 1, the spectral radius (largest absolute value of eigenvalue) is also at most 1. (This can be proven directly, but it also a consequence of the Gershgorin circle theorem [Ger20].) Since the rank of V^TV is at most d, V^TV has at most d non-zero eigenvalues. We can express the Frobenius norm squared, $\|V^TV\|_2^2$, by the sums of squares of eigenvalues. By the arguments above, $\|V^TV\|_2^2 \leq d$.

Note that $\|V^TV\|_2^2$ is also the sums of squares of entries. Each potential community pair contributes at least $2\varepsilon^2$ to this sum. Hence, there can be at most $d/2\varepsilon^2$ potential community pairs. □

The instability of softmax factorizations: The properties of softmax factorizations are more nuanced. Firstly, we can prove that softmax factorizations *can* represent community structure quite effectively.

Theorem 6.1.3. *For $d = O(\log n)$, there exists $V \in \mathbb{R}^{d\times n}$ such that $\mathrm{nsm}(V)_{ij}$ exhibits community structure. Specifically, for any natural number $b \leq n$, there exists $V \in \mathbb{R}^{d\times n}$ such that $\mathrm{nsm}(V)$ has n/b blocks of size b, such that all entries within blocks are at least $1/2b$.*

Indeed, this covers the various SBM settings we study, and demonstrates the superiority of softmax factorizations for modeling community structure. We note that a similar theorem (for a different type of factorization) was proved in [CMST20].

On the other hand, we prove that these factorizations are highly *unstable* to small perturbations. Indeed, with a tiny amount of noise, any community pair can be destroyed with high probability.

Formally, our noise model is as follows. Let $\delta > 0$ be a noise parameter. Think of the ith column of V as the d-dimensional vector $\vec{v}_i$, which is the embedding of vertex i. For every vector $\vec{v}_i$, we generate an independent random Gaussian $X_i \sim \mathcal{N}(0, \delta^2)$ and rescale $\vec{v}_i$ as $(1 + X_i)\vec{v}_i$ (formally, we rescale to $e^{X_i}\vec{v}_i$, to ensure that the scaling is positive). We denote this perturbed matrix as $\widetilde{V}^{(\delta)}$. We think of δ as a quantity going to zero, as n becomes large. (Or, one can consider δ as a tiny constant.)

Theorem 6.1.4. *Let c denote some absolute positive constant. Consider any $V \in \mathbb{R}^{d\times n}$. For any $\delta > c\ln(1/\varepsilon)/\ln n$, the following holds in $\mathrm{nsm}(\widetilde{V}^{(\delta)})$. For at least $0.99n$ vertices i, for any pair (i, j), the pair is* not *a potential community pair with probability at least 0.99.*

Thus, with overwhelming probability, any community structure in $\mathrm{nsm}(V)$ is destroyed by adding $o(1)$ (asymptotic) noise. This is strong evidence that either noise in the input or numerical precision in the final optimization could lead to destruction of community structure. These theorems give an explanation of the poor performance of the embeddings methods studied.

6.1.1 Proof ideas

We give the full proof details in §6.4. In this section, we lay out the main ideas in proving Theorem 6.1.4. It helps to begin with the upper bound construction of Theorem 6.1.3. Quite simply, we take n/b random Gaussian vectors, and map vertices in a community/block to the same vector. After doing some calculations of random dot products, one can deduce that the vectors need to have length $\Omega(\sqrt{\ln n})$ for the construction to go through. By carefully look at the math, one also finds that the construction is "unstable". Even perturbing the vectors within a block slightly (so that they are no longer the same vector) affects the block structure. We essentially prove that these properties hold for *any* set of vectors.

We outline the proof of Theorem 6.1.4. Suppose $\text{nsm}(V)_{ij} > \varepsilon$. Note that $\sum_{k\in[n]} \text{nsm}(V)_{ik} = 1$, since $\text{nsm}(V)$ is normalized by construction. Thus, there exists some k such that $\text{nsm}(V)_{ik} \leq 1/n$. We deduce that $\text{nsm}(V)_{ij}/\text{nsm}(V)_{ik} > \varepsilon n$. Writing out the entries and taking logs, this implies $\vec{v}_i \cdot \vec{v}_j - \vec{v}_i \cdot \vec{v}_k > \ln(\varepsilon n)$. Therein lies the power (and eventual instability) of softmax factorizations: ratios of entries are transformed into differences of dot products. By Cauchy-Schwartz, one of $\vec{v}_i, \vec{v}_j, \vec{v}_k$ must have length $\Omega(\sqrt{\ln n})$ (ignoring ε dependencies). By an averaging argument, we can conclude that for a vast majority of community pairs (i, j), $\|\vec{v}_i\|_2 = \Omega(\sqrt{\ln n})$. Note that both $\text{nsm}(V)_{ij}$ and $\text{nsm}(V)_{ji}$ must be at least ε; these quantities have the same numerator, but different denominators. By analyzing these expressions and some algebra, we can prove that $\left|\|\vec{v}_i\|_2 - \|\vec{v}_j\|_2\right| = o(1/\sqrt{\ln n})$.

Thus, we discover the key property of communities expressed by softmax factorizations. Vectors with a community have length $\Omega(\sqrt{\ln n})$, but the differences in lengths must be $O(1/\sqrt{\ln n})$. Asymptotically, this is unstable to perturbations in the length. A vanishingly small change in the vector lengths can destroy the community.

6.2 Empirical verification

6.2.1 Community pair prediction

We study the performance of matrix factorization embeddings at identifying community structure in a graph with many possibly overlapping communities. Formally, we state this as a binary classification task over pairs of vertices. Every dataset consists of a graph, G, and a set of (possibly overlapping) communities, $C_1, C_2, \ldots$. This gives us a ground truth labeling over the pairs from $V \times V$ where positive instances are those (u, v) such that $u, v \in C_k$, for some community C_k. We evaluate the performance of embedding techniques on this binary classification task over $V \times V$ as follows:

1. An embedding method is applied to G to obtain an embedding $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ of the nodes in V.

2. A *pair scoring function*, $f : V \times V \to \mathbb{R}$, is constructed from the embedding of node pairs.

All of the evaluation is done with respect to this pair scoring function. We use this abstraction to encapsulate all of the tasks downstream of the embedding generation itself. It consists of mapping the embedding vectors to predictions in $\mathbb{R}$. Ideally, $f(u, v) > f(x, y)$ whenever u and v share a community and x and y do not. Every violation of this represents some loss in the ability to reconstruct community structure. The construction of f is based on the dot product of the embedding vectors of u and v. The full construction is given in §6.2.3.

6.2.2 Experimental results

We observe that no choice of embedding method were competitive with the baseline `LR-Structural` method on the task of community pair prediction. Across three datasets, `LR-Structural` was consistently able to identify community pairs while the embedding-based methods were not. For each method being compared, one thousand vertices were selected at random from the graph, and we examine the precision of the classifier on the neighborhoods of each selected vertex. Fig. 6.1 shows that `LR-Structural` makes more precise predictions on average than the best performing datasets, while Fig. 6.2 contains the mean precision for each classifier on each dataset.

The methods are evaluated by comparing the distribution of a precision@10 for each method on each dataset. For each of 1000 vertices sampled, v, we order the other vertices of the graph, $u_1, \ldots, u_n$ such that $f(\boldsymbol{v}, \boldsymbol{u}_i) \geq f(\boldsymbol{v}, \boldsymbol{u}_{i+1})$ for all i. We compute the precision of the classifier on $(v, u_1), \ldots, (v, u_n)$ when it predicts positive labels only for those (v, u_i) such that $i \leq 10$. In other words, we sample a vertex at random and report the fraction of its ten nearest neighbors in the embedding space with which it shares a community.

We represent the distribution of values of precision@10 scores as a reliability curve. This is the curve (x, y) such that at least a y fraction of vertices sampled had a precision@10 score score of at least x. Higher y values for a given x indicate better performance. Fig. 6.1 contains the curves for the best performing dot product methods against the baseline, while Fig. 6.2 contains the mean across samples.

Given Theorem 6.1.4, and that each of the embedding methods contains some amount of noise, be it approximation error or the result of randomness over internal sampling procedures, we expect that if (u, v) is a community pair in some ground truth matrix, M, then it is very unlikely that (u, v) is a community pair in any noisy approximation of M. The results in this section bear out this conclusion, as we show that whatever community structure exists in the approximations produced by the embedding methods do not correlate well with the actual community structure in the graphs examined when compared to the performance of more naive methods.

method/dataset	amazon	dblp	sbm
`LR-Structural`	.92	.80	.88
DeepWalk	.44	.49	.35
NetMF	.49	.28	.66
Node2Vec	.49	.61	.55
DeepWalk-hp	.43	.44	.33
NetMF-hp	.41	.37	.76
Node2Vec-hp	.37	.55	.50

Figure 6.2: Average precision@10 across 100 samples for all methods across the three datasets. In all cases, `LR-Structural` is the best performing.

6.2.3 Experimental setup

Community pair prediction methods

We compare the performance of three different embedding methods to a baseline to a simple supervised logistic regression model using common structural graph features. The implementation of the embedding methods is based on [RKS20] and will be made available by the time of publication. Except for setting the dimension to 128 for all embedding methods, the hyperparameters are taken from [RKS20].

`LR-Structural` For the baseline method, we use a logistic regression model trained to predict community pairs based on four tractable graph features found to be useful in the literature:

- Cosine similarity between u's and v's adjacency vectors [SSG17],
- Size of the cut between u's neighborhood and v's neighborhood, and
- Personalized PageRank (PPR) score from u to v, as well as that from v to u [RAL07].

We use the approximation algorithm from [RAL07] to compute sparse approximate PPR vectors. Note that this allows for all features to be computed locally, i.e. the space/time complexity for computing the features for one vertex pair are independent of graph size. The overall space/time costs in practice are on par with the embedding methods.

GraRep This is a direct factorization method (see definition in §6.0.1). The embedding it returns is the concatenation of an embedding fit to a modified version of the k-step random walk matrix for each k up to a hyperparameter K. Consistent with [RKS20], we set $K = 5$. With this parameter setting, GraRep was not able to complete on the test machine since it requires explicitly computing powers of the adjacency matrix which is not feasible with any memory restraints.

DeepWalk This is a softmax factorization technique. It performs random walks to approximate the mean of the first K step random walk matrices (this is the window size parameter). In our experiments we use $K = 5$.

Node2Vec Like DeepWalk, this is a softmax factorization technique which generates an approximation to an average over the first K random walk matrices by performing random walks. However, the random walks performed are a special kind which are controlled by parameters p and q. When $p = q = 1$, the walks are identical to ordinary random walks. In our experiments, we use $p = q = 0.5$ and $K = 5$.

NetMF This is a direct factorization technique which performs SVD on a modified version of the sum of the first K random walk matrices. For our experiments, and consistent with [RKS20], we use $K = 2$.

Pair features

The embedding methods produce a feature vector for each vertex. Since the community pair prediction task requires a set of feature vectors over $V \times V$ and a method to produce scores, we need to turn node features into pair features. Building on strategies proposed in prior work [GL16a], for the embeddings we choose two methods from the literature which are computationally efficient, and relatively accurate. Given a d-dimensional embedding, $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ we compute the feature vector for $(u, v) \in V \times V$ with the *dot product* and *Hadamard product* denoted $\boldsymbol{u} \cdot \boldsymbol{v}$ and $\boldsymbol{u} \circ \boldsymbol{v}$ respectively.

The dot product is the usual inner product over $\mathbb{R}^d$, i.e. $\boldsymbol{u} \cdot \boldsymbol{v} = \sum_{i \in [d]} \boldsymbol{u}(i)\boldsymbol{v}(i)$. It takes the two d-dimensional vector embeddings and maps them to a 1-dimensional feature vector. On, the other hand, the Hadamard product of $\boldsymbol{u}$ and $\boldsymbol{v}$, $\boldsymbol{u} \circ \boldsymbol{v}$, is a d-dimensional vector whose ith coordinate is the product of the ith coordinates of $\boldsymbol{u}$ and $\boldsymbol{v}$.

We will indicate when the pair features used are the dot product or Hadamard product of the endpoints. The actual scoring function, $f : V \times V \to \mathbb{R}$, we use is implicit from the pair features. Whenever the pair features are dot products, $f(u, v) = \boldsymbol{u} \cdot \boldsymbol{v}$. Whenever the pair features are the Hadamard product of $\boldsymbol{u}$ and $\boldsymbol{v}$, we fit a logistic regression model to the features $\boldsymbol{u} \circ \boldsymbol{v}$ to produce a weight vector β, and so the pair scoring function becomes $f(u, v) = \sigma(\beta \cdot (\boldsymbol{u} \circ \boldsymbol{v}))$ where $\sigma = \exp(x)/(1 + \exp(x))$ is the sigmoid function.

In order to compute the weight vector, β, in the Hadamard product case, we select a training set of size $50n$ by choosing 50 vertices which participate in a community with at least twenty members, $v_1, \ldots, v_{50}$, and compute $\boldsymbol{u} \circ \boldsymbol{v}_i$ for each $u \in V$. This collection of $50n$ feature vectors, along with ground truth labeling of community pairs, are given to an sklearn LogisticRegression object which produces β.

Implementation

All our experiments were run on AWS using machines with up to 96 cores and 1 TB memory. The methods were implemented in python based on the karateclub package (cite) (the code used to generate results is available anonymously before publication at https://anonymous.4open.science/r/0bee6b7b-870f-4b7e-ac8f-ee3f336295e2/). Except for setting the dimension to 128 for all methods, the default hyperparameters were used. Any method which did not complete in 24 hours or which required more memory was considered not complete.

Datasets

We show the performance of the various embedding methods contrasted with `LR-Structural` on three datasets: two publicly available real world datasets with ground truth community labels, and one synthetic stochastic block model (SBM).

- dblp: a co-authorship network of 317K computer science authors with communities defined as venues
- amazon: a network of 335K products on amazon with a link representing frequent co-purchasing. Communities are product categories.
- sbm: synthetic dataset with 100K vertices and communities of size 20. Edges are randomly generated with an inter-community edge probability of 0.3 and intra-community edge probability of $0.3/n$.

Statistic	**dblp**	**amazon**	**sbm**
number of nodes	317K	334K	100K
number of edges	1M	1M	585K
num communites	13K	75K	5K
max size	75K	53K	20
median size	8	5	20
mean comm density	0.09	0.10	0.30

Table 6.1: Dataset summary

"Mean comm density" is an average weighted by community size of the edge density of each community.

6.3 Stochastic block models

6.3.1 Setup

We complement our experimental results on real datasets with measuring performance of graph embedding methods on synthetic datasets generated according to the popular Stochastic Block Model (SBM). An n-vertex graph, G, is generated according to an SBM by partitioning the vertex set, $[n]$ into k equal sized communities, $C_1, C_2, \ldots, C_k$. The distribution of edges is controlled by two parameters: p and q. For two vertices in the same community, u, v, p controls the probability of edge (u, v) being added to the graph. For u, v not in the same community, the edge (u, v) is added with probability q. We chose the SBM parameters in order to approximate some of the features we observed in the empirical datasets.

All SBMs have one hundred thousand vertices and communities (or "blocks") of size 20. In Tab. 6.1, the average community size for all datasets is much smaller. The parameter q is set so that the average number of neighbors a vertex has inside its community is equal to that outside of its community. In real data, there are often 3 or 4 times as many inter-community neighbors as intra-community neighbors. The three SBM graphs generated, sbm_sparse, sbm, sbm_dense, have average degrees 4, 12 and 20 respectively.

6.3.2 Results

Motivated by Theorem 6.1.3, we can study the embeddings' resilience to noise in the simulated setting. Specifically, we simulate noise in the community structure of the original graph. We generate three SBMs to mimic real datasets and perform community pair prediction on them. Fig. 6.3 shows the results from this experiment. Note that the accuracy of the embeddings at pairwise community labeling decreases as the internal density of the communities decreases.

Here we demonstrate the robustness of our results. `LR-Structural` is able to outperform the factorization- based embedding methods across a regime of parameters commonly encountered in sparse datasets. There is a wide body of work on the relationship between triangles in the graph and community structure ([SSSG20b] recently explored this in the context of graph embeddings). As the density increases, the probability that a vertex participates in an intra-block triangle increases. In sbm_sparse, very few vertices participate in triangles, while in sbm_dense, many more do. In all cases, `LR-Structural` is the best performing.

6.4 Proof of instability in softmax factorizations

We restate and prove the upper bound theorem, showing that softmax factorizations can recreate community structure.

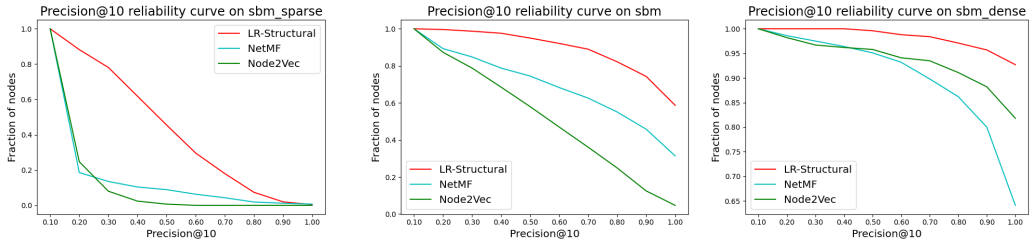

Figure 6.3: Precision@10 reliability curves for sbm datasets. All datasets are stochastically generated datasets with size 10,000 and disjoint communities of size 20. The average degrees are 4, 12 and 20 from left to right. Curves are generated from 100 samples.

Theorem 6.1.3. *For $d = O(\log n)$, there exists $V \in \mathbb{R}^{d\times n}$ such that $\mathrm{nsm}(V)_{ij}$ exhibits community structure. Specifically, for any natural number $b \leq n$, there exists $V \in \mathbb{R}^{d\times n}$ such that $\mathrm{nsm}(V)$ has n/b blocks of size b, such that all entries within blocks are at least $1/2b$.*

Proof. We apply the probabilistic method. We select V from a random distribution, and prove that the desired community structure is exhibited with high probability.

Let $d = c \ln n$, for some sufficiently large constant c. We will construct n/b blocks of vertices, each with b vertices. All the vertices in the bth block will be represented by the same vector $\vec{v}_b$. We will set each $\vec{v}_b$ to be a uniform random Gaussian vector of length $2\sqrt{\ln n}$. Thus, for any two vertices i, j within a block, $\vec{v}_i \cdot \vec{v}_j = 4 \ln n$. For two vertices i, j that occur in different blocks, the dot product $\vec{v}_i \cdot \vec{v}_j$ is normally distributed with mean zero and variance $(4 \ln n)/d = 4/c$. By tail bounds for the Gaussian, $\Pr[\vec{v}_i \cdot \vec{v}_j \geq (16 \ln n)/c] \leq 1/n^4$, where the probability is over the choice of the vectors. By a union bound over all (at most) n^2 pairs, with probability at least $1 - 1/n^2$, for every pair i, j in distinct blocks, $\vec{v}_i \cdot \vec{v}_j < (16 \ln n)/c$, which is at most $\ln n$ (for sufficiently large c).

Note that, for all vertices i, we can split the sum $\sum_k \exp(\vec{v}_i \cdot \vec{v}_k)$ into vertices within i's block and outside i's block. The total sum outside the block is at most $n \times \exp(\ln n) = n^2$. The sum within the block is $b \exp(4 \ln n) = bn^4$. For i, j within a block, $\mathrm{nsm}(V)_{ij} \geq \exp(4 \ln n)/(bn^4 + n^2) \geq 1/2b$. Thus, $\mathrm{nsm}(V)$ exhibits the desired community structure. □

We note some features of this construction. Firstly, the vectors are of non-constant length. Moreover, the vectors within a community are extremely close to each other compared to their length; in the construction, they are actually identical.

We essentially prove that both these properties are necessary for *any* V where $\mathrm{nsm}(V)$ exhibits community structure. The second property of closeness is extremely sensitive to noise, and we prove that even small amounts of noise can destroy the community structure. In practice, we believe that such unstable solutions are hard to find, and the ambient noise in data forces

solutions that are stable to perturbations. This point is validated in our experiments, where the solutions do not exhibit community structure.

Lemma 6.4.1. *For at least* $0.99n$ *vertices* i*, the following property holds. If* (i, j) *is a potential community pair, then* $\|\vec{v}_i\|_2^2 \geq \sqrt{\ln(\varepsilon n/100)}$ *and* $\left|\|\vec{v}_i\|_2 - \|\vec{v}_j\|_2\right| \leq 2\ln(1/\varepsilon)/\sqrt{\ln(\varepsilon n/100)}$.

Proof. Recall that $\mathrm{nsm}(V)_{i,j} = \exp(\vec{v}_i \cdot \vec{v}_j)/\sum_k \exp(\vec{v}_i \cdot \vec{v}_k)$. There are n vertices, so by Markov's inequality, for at least $0.99n$ vertices k, $\mathrm{nsm}(V)_{i,k} \leq 100/n$. Pick any such k, and note that $\mathrm{nsm}(V)_{i,j}/\mathrm{nsm}(V)_{i,k} = \exp(\vec{v}_i \cdot \vec{v}_j - \vec{v}_i \cdot \vec{v}_k) \geq \varepsilon n/100$. Taking logs, $\vec{v}_i \cdot (\vec{v}_j - \vec{v}_k) \geq \ln(\varepsilon n/100)$. By Cauchy-Schwartz, $\|\vec{v}_i\|_2\|\vec{v}_j - \vec{v}_k\|_2 \geq \ln(\varepsilon n/100)$. By the triangle inequality, $\|\vec{v}_i\|_2(\|\vec{v}_j\|_2 + \|\vec{v}_k\|_2) \geq \ln(\varepsilon n/100)$.

For convenience, let $N := \varepsilon n/100$. Suppose that $\|\vec{v}_i\|_2 < \sqrt{\ln N}/2$ and $\|\vec{v}_j\|_2 < \sqrt{\ln N}/2$. This implies that $\|\vec{v}_j\|_2 + \|\vec{v}_k\|_2 \geq \sqrt{\ln N}$, and thus, $\|\vec{v}_k\|_2 \geq \sqrt{\ln N}/2$.

Thus, one of the following holds. Either for every potential community pair (i, j),

$$\max(|\vec{v}_i\|_2, \|\vec{v}_j\|_2) \geq \sqrt{\ln N}/2,$$

or for at least $0.99n$ vertices k, $\|\vec{v}_k\|_2 \geq \sqrt{\ln N}/2$. Regardless, for at least $0.99n$ vertices i, for every potential community pair (i, j), $\max(|\vec{v}_i\|_2, \|\vec{v}_j\|_2) \geq \sqrt{\ln N}/2$.

Consider a potential community pair (i, j) where $\max(|\vec{v}_i\|_2, \|\vec{v}_j\|_2) \geq \sqrt{\ln N}/2$. Wlog, let $\|\vec{v}_i\|_2$ be larger. Recall that $\mathrm{nsm}(V)_{ij} = \exp(\vec{v}_i \cdot \vec{v}_j)/\sum_k \exp(\vec{v}_i \cdot \vec{v}_k) \geq \varepsilon$.

$$\begin{aligned}
& \exp(\vec{v}_i \cdot \vec{v}_j) \geq \varepsilon \sum_k \exp(\vec{v}_i \cdot \vec{v}_k) \\
\implies \quad & \exp(\vec{v}_i \cdot \vec{v}_j) \geq \varepsilon \exp(\vec{v}_i \cdot \vec{v}_i) \\
\implies \quad & \vec{v}_i \cdot \vec{v}_j \geq \ln \varepsilon + \|\vec{v}_i\|_2^2 \\
\implies \quad & \|\vec{v}_i\|_2\|\vec{v}_j\|_2 \geq \ln \varepsilon + \|\vec{v}_i\|_2^2 \quad \text{(Cauchy-Schwartz)} \\
\implies \quad & \|\vec{v}_j\|_2 \geq \|\vec{v}_i\|_2 - (\ln 1/\varepsilon)/\|\vec{v}_i\|_2 \qquad (6.2)
\end{aligned}$$

Since $\|\vec{v}_i\|_2 \geq \sqrt{\ln N}/2$ (and $\|\vec{v}_i\|_2 \geq \|\vec{v}_j\|_2$), $|\|\vec{v}_i\|_2 - \|\vec{v}_j\|_2| \leq 2\ln(1/\varepsilon)/\sqrt{\ln N}$ □

We now state the perturbation instability theorem (we skip the proof), which should be intuitively clear by Lemma 6.4.1. For (almost all) community pairs (i, j), both $\|\vec{v}_i\|_2$ and $\|\vec{v}_j\|_2$ are at least $\Omega(\sqrt{\ln n})$, but the *difference* between the lengths is $O(1/\sqrt{\ln n})$. Infinitesimal perturbations will destroy such a property, and thus the community pair will be lost.

Theorem 6.1.4. *Let* c *denote some absolute positive constant. Consider any* $V \in \mathbb{R}^{d\times n}$*. For any* $\delta > c\ln(1/\varepsilon)/\ln n$*, the following holds in* $\mathrm{nsm}(\widetilde{V}^{(\delta)})$*. For at least* $0.99n$ *vertices* i*, for any pair* (i, j)*, the pair is* not *a potential community pair with probability at least* 0.99*.*

Chapter 7

Evaluating embeddings for link prediction

Link prediction is perhaps the most important prediction problem for modern network data. If our data is somehow incomplete, e.g. edges are missing, then link prediction is the problem of predicting whether the edge (u, v) is in the graph for some pair of vertices u and v. Many pracitcal problems reduce to link prediction: product recommendation, identifying network intrusion, studying protein function, etc.

There are many approaches to link prediction - we could consider many features which do not have network structure - but we shall focus here on *structural* link prediction approaches. In other words, we are interested in how we can predict future links in the graph only from the observable graph structure. One of the first works to address this problem in the modern machine learning context is [LNK07]. They showed that simple structural pairwise-features, such as the Jacard similarity between two sets of neighbors, can be very successful unsupervised predictors in a wide variety of real world networks. Observe that there need not be any correlation between past links and future links in the graph, but the work of Liben-Nowell helped establish that real networks have common, specific structural characteristics which we have observed previously in this part.

Since 2007, there have been various link prediction methods. Some touting the superiority of supervised approaches [AHCSZ06], and a plethora of unsupervised embedding methods have we discussed throughout this part. While each new algorithm touts some kind of improvement over the previous ones, a debate over how best to evaluate link prediction methods is hardly settled.

There are a plethora of different evaluation criteria to consider, see [LLC10, ME11, YLC15] for some excellent discussions. In this chapter we focus on just one aspect of this

discussion: the use of ROC AUC as an evaluation metric. This gives a one number summary of a binary predictor and is one of the most consistently used throughout the literature. We recommend against its use since it is known to be distorting on sparse data [DG06]. We shall recap the reasons why in the next section.

The discussion on how to evaluate link prediction methods has evolved alongside the methods themselves. New work is analyzed using new methods. Little effort has been made to understand the performance of the older, simpler methods using new perspectives. This problem is especially apparent in the world of graph embeddings where we see an explosion of complex, high tech deep learning methods coming from a simple matrix factorization approach which is decades old.

In this chapter, we shall recap why AUC is not appropriate for sparse networks, and show how reliability curves can reveal some real deficiencies in the performance of embeddings for link prediction.

7.1 The problem with AUC

ROC AUC stands for Receiver Operating Characteristic Area Under Curve. It is defined as the area under a certain curve, but it turns out to have a much simpler equivalent interpretation. Suppose we have some binary classification task on some set of objects, $x_1, \ldots, x_n$. The objects have hidden ground truth labels in $\{0, 1\}$. Let y_i be the ground truth label of x_i. Our classifier produces scores in $\mathbb{R}$, $\hat{y}_i$ for each x_i. We hope that the scores are such that if $\hat{y}_i > \hat{y}_j$ then $y_i \geq y_j$. The ROC AUC can be used to evaluate the extent to which this is true.

ROC AUC is equivalent to the expectation of the following experiment: select a random x_i such that $y_i = 1$ and a random x_j such that $y_j = 0$, and output the indicator random variable for the event that $\hat{y}_i \geq \hat{y}_j$. Note that the value of AUC then does not depend at all on the proportion of positive instances (i.e. those x_i such that $y_i = 1$) and negative instances ($y_i = 0$). It only depends on the distribution of scores among positive instances and the distribution among negative instances. For this reason, AUC is sometimes recommended on sparse datasets. As our next example will show, this thinking is dangerous.

Networks are incredibly sparse, even compared to sparse datasets found in other domains. To see the potential distorting effects of AUC, consider the following example. Suppose we are given a sparse graph with dn edges, and two different link prediction models. Each model assigns a score to each pair of vertices in the graph, and the more accurate model should rank pairs corresponding to edges above those which do not. In the ranking induced by Model A, the top $10dn$ pairs are non-edges, followed by the dn edges and then the remaining $\Theta(n^2)$ non-edge pairs. Model A has a ROC AUC of $1 - O(d/n)$ since there is only a d/n probability of picking one of the non-edges ranked above the edges from among all non-edge pairs. In Model B, the

first $dn/2$ pairs are edges, followed by half of the non-edge pairs, then the remaining edges and last the remaining non-edge pairs. Here the probability that a random edge is ranked higher than a random non-edge is only $\sim .75$. Model A has near perfect ROC AUC, but does it really offer better performance than Model B?

The problem with AUC is that it is not sensitive to the unique problems posed by sparse data. Identifying the few true instances from among the vast quantity of negative instances should be the focus of any classifier. Having a high AUC does not guarantee this. We can have an AUC of 0.999999, but still, when we examine the ranking of scores output by our classifier, it may requires sifting through hundreds of negatives at the top before we find a single true positive prediction.

7.2 Reliability curves

Reliability curves come out of our experience with item recommendation in sparse networks. In this context, we are interested in providing good recommendations to each user of our service. This boils down to predicting good links for every vertex in the graph. Reliability curves show how reliably a classifier is able to do this across the whole dataset.

Given a classifier, we compute reliability curves as follows: for every vertex we calculate a precision@d score. We assume the degree of v, d_v is given and calculate a precision@d_v for each v. The precision@k is the fraction of these pairs which are edges in the graph. The reliability curve provides a visualization of the distribution we observe over the precision@d scores. Each point (x, y) on the reliability curve indicates that y% of the vertices had a precision@d of at least x. The reliability curves are monotone decreasing from the point $(0, 1)$ to $(1, 0)$, and for each x-value along the way, a higher y-value indicates better performance.

The reliability curves thus measure precisely what AUC does not - the quality of the top few predictions as ranked by the classifier. Moreover, reliability curves provide a view normalized over vertices so that we can see if the performance varies for different vertices in the graph.

7.3 Experimental results

We evaluate the performance of each of the embeddings as features in a logistic regression model in two different ways: the ROC AUC scores of Fig. 7.1 and the reliability curves of Fig. 7.2. We present these two as a contrast. We show that despite the apparently impressive ROC AUC scores for the embedding methods, the reliability curves show that the nearest neighbors in the embeddings do not often contain very many vertices which are actually neighbors in the graph.

dataset	deepwalk	netmf	node2vec	structural
amazon	.91	.86	.87	1.00
dblp	.97	.95	.99	1.00
blogcatalog	.78	.87	.83	.96
ppi	.66	.93	.87	.97

Figure 7.1: ROC-AUC scores of hadamard product embedding models. The scores are calculated on the test set consisting of half true-labeled pairs and half negatively labeled pairs

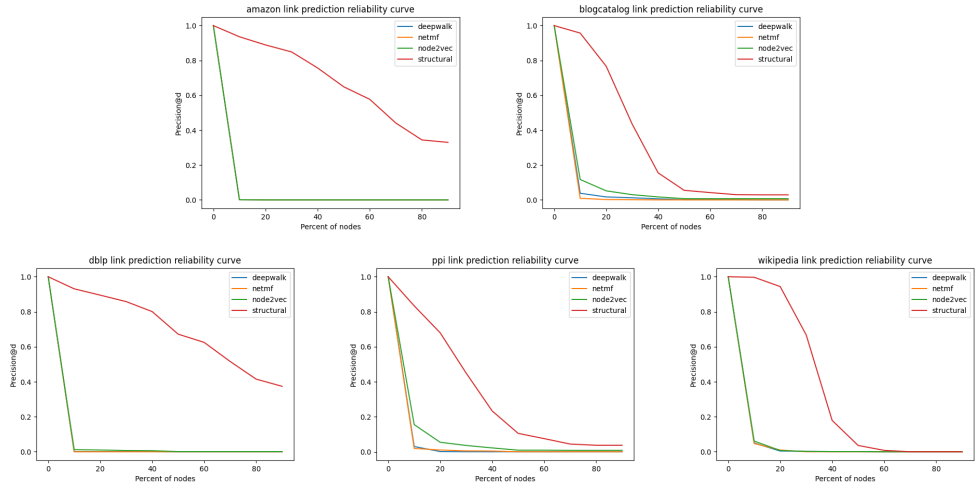

Figure 7.2: Reliability curves from 1K vertex samples. For each vertex, v, in the sample, the top d_v neighbors according to the classifier is selected and a precision@d is calculated for each vertex. Each point, (x, y), on the curve represents the approximate fraction of vertices, y, for which the given method produces a precision@d score of at least x.

7.3.1 Experimental setup

For each dataset, we generate an embedding for each vertex using the complete dataset for use in a downstream logistic regression model. In order to construct such a link prediction model, we need to transform the per-vertex embedding vectors into features across all pairs of vertices. Past work shows that the hadamard product of the embedding vectors usually offers the best performance.

The embedding vectors are used as features in training a logistic regression model. The training data consists of half of all edges in the graph and an equal number of non-edge pairs. All of the models were trained using the liblinear solver implemented in sklearn's LogisticRegression package. We use the remaining half of edges and another set of non-edge pairs as the test set.

The ROC AUC scores of Fig. 7.1 are calculated on the test sets. Across the half of

edges not in the training set, and an equal number of non-edge pairs, we see relatively high AUC scores.

The reliability curves (Fig. 7.2) are calculated from a sample of one thousand vertices. For each vertex, v, sampled, we calculate the scores output by the classifier for all pairs $(v, u_1), \ldots, (v, u_n)$. We assume the degree of v, d_v is given and calculate a precision@d_v for each such v in the sample. The precision@k is the fraction of these pairs which are edges in the graph. The reliability curve provides a visualization of the distribution we observe over the precision@d scores in our sample. Each point (x, y) on the reliability curve indicates that $y\%$ of the vertices sampled had a precision@d of at least x. The reliability curves are monotone decreasing from the point $(0, 1)$ to $(1, 0)$, and for each x-value along the way, a higher y-value indicates better performance.

7.3.2 Datasets

We test all the methods using publicly available real world datasets. In order to show that our results are broadly applicable, we choose a variety of massive sparse graph across different domains. See Tab. 5.2 for a summary of the datasets.

amazon [**LK14**] This is a product co-purchasing network for the retail website amazon. Nodes represent product and edges indicate products which are frequently purchased together.

blogcatalog [**RA15**] This is a social network in which vertices represent users of the bloggling websit BlogCatalog and consists of links between users.

dblp [**LK14**] This is a co-authorship network where each vertex represents an author with a publication in the DBLP computer science bibliography. Edges indicate that two authors collaborated on at least one publication.

ppi [**GL16a**] In this protein-protein interaction network, vertices represent proteins in the human body and edges represent observed interactions between them.

wikipedia [**GL16a**] This network represents word co-occurrences for a subset of Wikipedia. Vertices represent words and edges represent their co-occurence in articles.

Table 7.1: Summary of datasets

Name	n	m
amazon	335K	926K
blogcatalog	10K	334K
dblp	317K	1M
ppi	4K	38K
wikipedia	5K	92K

Bibliography

[Abb18] Emmanuel Abbe. Community detection and stochastic block models: Recent developments. *Journal of Machine Learning Research*, 18:1–86, 2018. 96

[ABS15] Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. *jacm*, 62(5):42:1–42:25, 2015. 12

[ACL06] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486. IEEE, 2006. 5, 7, 16

[AF02] David Aldous and James Allen Fill. Reversible markov chains and random walks on graphs, 2002. Unfinished monograph, recompiled 2014, available at `http://www.stat.berkeley.edu/~aldous/RWG/book.html`. 38

[AFL+18] Avanti Athreya, Donniell E. Fishkind, Keith Levin, Vince Lyzinski, Youngser Park, Yichen Qin, Daniel L. Sussman, Minh Tang, Joshua T. Vogelstein, and Carey E. Priebe. Statistical inference on random dot product graphs: a survey. *Journal of Machine Learning Research*, 18:1–92, 2018. 93, 96

[AFNS06] N. Alon, E. Fischer, I. Newman, and A. Shapira. A combinatorial characterization of the testable graph properties : it's all about regularity. *stoc*, pages 251–260, 2006. 11

[AGPT16] Reid Andersen, Shayan Oveis Gharan, Yuval Peres, and Luca Trevisan. Almost optimal local graph clustering using evolving sets. *Journal of the ACM (JACM)*, 63(2):1–31, 2016. 5, 7, 12

[AHCSZ06] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. Link prediction using supervised learning. In *SDM06: workshop on link analysis, counter-terrorism and security*, volume 30, pages 798–805, 2006. 126

[AL06] Reid Andersen and Kevin J Lang. Communities from seed sets. In *Proceedings of the 15th international conference on World Wide Web*, pages 223–232, 2006. 115

[Alo03] N. Alon. Problems and results in extremal combinatorics, part i, discrete math. *Discrete Math*, 273:31–53, 2003. 99

[Arn19] Citation network dataset. https://aminer.org/citation, 2019. 104

[ASN+13] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. Distributed large-scale natural graph factorization. In *www*, pages 37–48, 2013. 92

[AST90] Noga Alon, Paul Seymour, and Robin Thomas. A separator theorem for nonplanar graphs. *Journal of the American Mathematical Society*, 3(4):801–808, 1990. 79

[AST94] Noga Alon, Paul D. Seymour, and Robin Thomas. Planar separators. *SIAM J. Discrete Math.*, 7(2):184–193, 1994. 55

[BA99] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999. 92

[BCG10] Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. Fast incremental and personalized pagerank. *PVLDB*, 4(3):173–184, 2010. 93

[BFU99] Andrei Z. Broder, Alan M. Frieze, and Eli Upfal. Static and dynamic path selection on expander graphs: A random walk approach. *Random Struct. Algorithms*, 14(1):87–109, 1999. 12

[BMS+16] Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. Recent advances in graph partitioning. *Algorithm engineering*, pages 117–158, 2016. 3

[BSS10] I. Benjamini, O. Schramm, and A. Shapira. Every minor-closed property of sparse graphs is testable. volume 223, pages 2200–2218, 2010. 9, 47, 58

[CAEHP+20] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *arXiv preprint arXiv:2005.03675*, 2020. 7, 109, 112, 113, 114, 115

[CAS16] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016. 109, 112

[CF06] D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys*, 38(1), 2006. 92

[CGR+14] Artur Czumaj, Oded Goldreich, Dana Ron, C Seshadhri, Asaf Shapira, and Christian Sohler. Finding cycles and trees in sublinear time. *Random Structures & Algorithms*, 45(2):139–184, 2014. 10, 11, 59

[CLX15] Shaosheng Cao, Wei Lu, and Qiongkai Xu. GraRep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management - CIKM '15*, pages 891–900. ACM Press, 2015. 111, 114

[CLX16] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *aaai*, pages 1145–1152, 2016. 92

[CMST20] Sudhanshu Chanpuriya, Cameron Musco, Konstantinos Sotiropoulos, and Charalampos E Tsourakakis. Node embeddings and exact low-rank representations of complex networks. *arXiv preprint arXiv:2006.05592*, 2020. 110, 113, 114, 115, 117

[CPS15] Artur Czumaj, Pan Peng, and Christian Sohler. Testing cluster structure of graphs. In *stoc*, pages 723–732, 2015. 12

[CRV15] Peter Chin, Anup Rao, and Van Vu. Stochastic block model and community detection in sparse graphs: A spectral algorithm with optimal rate of recovery. In *Conference on Learning Theory*, pages 391–423, 2015. 115

[CS10] Artur Czumaj and Christian Sohler. Testing expansion in bounded-degree graphs. *Combinatorics, Probability & Computing*, 19(5-6):693–709, 2010. 12

[CSS09] Artur Czumaj, Asaf Shapira, and Christian Sohler. Testing hereditary properties of nonexpanding bounded-degree graphs. *SIAM Journal on Computing*, 38(6):2499–2510, 2009. 11, 58

[DG06] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006. 112, 113, 127

[Die10] Reinhard Diestel. *Graph Theory, Fourth Edition*. Springer, 2010. 11

[DP09] D. P. Dubhashi and A. Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge, 2009. 51, 52, 54, 71

[DPKS12] N. Durak, A. Pinar, T. G. Kolda, and C. Seshadhri. Degree relations of triangles in real-world networks and graph models. In *Conference on Information and Knowledge Management (CIKM)*, 2012. 93

[EHNO11] Alan Edelman, Avinatan Hassidim, Huy N. Nguyen, and Krzysztof Onak. An efficient partitioning oracle for bounded-treewidth graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques RANDOM 2011, Princeton, NJ, USA, August 17-19, 2011. Proceedings*, pages 530–541, 2011. 11, 59

[EK10a] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, 2010. 92

[EK10b] David Easley and Jon Kleinberg. *Networks, crowds, and markets*, volume 8. Cambridge university press Cambridge, 2010. 109

[FH16] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics reports*, 659:1–44, 2016. 115

[Fie73] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(98):298—305, 1973. 96

[FLVW17] Hendrik Fichtenberger, Reut Levi, Yadu Vasudev, and Maximilian Wötzel. On testing minor-freeness in bounded degree graphs with one-sided error. *CoRR*, abs/1707.06126, 2017. 10, 11, 59

[FLVW18] Hendrik Fichtenberger, Reut Levi, Yadu Vasudev, and Maximilian Wötzel. A sublinear tester for outerplanarity (and other forbidden minors) with one-sided error. In *ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 52:1–52:14, 2018. 10, 11

[Ger20] Gershgorin circle theorem. https://en.wikipedia.org/wiki/Gershgorin_circle_theorem, 2020. 116

[GGL+13] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Zadeh. Wtf: The who to follow service at twitter. In *www*, pages 505–514, 2013. 93

[GJJ20] Vikas K Garg, Stefanie Jegelka, and Tommi Jaakkola. Generalization and representational limits of graph neural networks. *arXiv preprint arXiv:2002.06157*, 2020. 110, 115

[GL16a] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864. ACM, 2016. 7, 110, 111, 114, 121, 130

[GL16b] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *kdd*, pages 855–864. ACM, 2016. 92, 95, 96, 106, 107

[Gol17] O. Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. 11, 56, 58, 62, 90

[GR99] O. Goldreich and D. Ron. A sublinear bipartite tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999. 11, 12, 58

[GR02] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. 9, 11, 55

[GR11] O. Goldreich and D. Ron. On testing expansion in bounded-degree graphs. *LNCS*, 6650, 2011. 12

[HKNO09] A. Hassidim, J. Kelner, H. Nguyen, and K. Onak. Local graph partitions for approximation and testing. In *focs*, pages 22–31, 2009. 11, 55, 56

[HLG+20] Weihua Hu*, Bowen Liu*, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020. 115

[HLL83] P. W. Holland, K. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5:109–137, 1983. 93, 96

[HRH02] P. D. Hoff, A. E. Raftery, and M. S. Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97:1090–1098, 2002. 96

[HT74] John Hopcroft and Robert Tarjan. Efficient planarity testing. *Journal of the ACM (JACM)*, 21(4):549–568, 1974. 9, 55

[HYL17a] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Neural Information Processing Systems (NeurIPS)*, page 11, 2017. 114, 115

[HYL17b] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Neural Information Processing Systems*, NIPS'17, pages 1025–1035, USA, 2017. Curran Associates Inc. 92, 95, 106

[HYL18] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. page 24, 2018. 7, 109, 114

[KKR12] Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce A. Reed. The disjoint paths problem in quadratic time. *J. Comb. Theory, Ser. B*, 102(2):424–435, 2012. 11, 18

[KPS13] Satyen Kale, Yuval Peres, and C. Seshadhri. Noise tolerance of expanders and sublinear expansion reconstruction. *SIAM J. Comput.*, 42(1):305–323, 2013. 12, 17, 38

[KR96] Jon M. Kleinberg and Ronitt Rubinfeld. Short paths in expander graphs. In *focs*, pages 86–95. IEEE Computer Society, 1996. 12, 13

[KS08] S. Kale and C. Seshadhri. Testing expansion in bounded degree graphs. *Proc. 35th ICALP*, pages 527–538, 2008. 12

[KSS18] Akash Kumar, C. Seshadhri, and Andrew Stolman. Finding forbidden minors in sublinear time: A $o(n^{1/2 + o(1)})$-query one-sided tester for minor closed properties on bounded degree graphs. In *focs*, pages 509–520, 2018. 8, 50, 58, 59

[KSS19a] Akash Kumar, C Seshadhri, and Andrew Stolman. Random walks and forbidden minors ii: a poly (d ε-1)-query tester for minor-closed properties of bounded degree graphs. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 559–567, 2019. 47

[KSS19b] Akash Kumar, C. Seshadhri, and Andrew Stolman. Random walks and forbidden minors II: a poly$(d\,\epsilon^{-1})$-query tester for minor-closed properties of bounded degree graphs. In *STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 559–567, 2019. 8, 58, 59, 79

[KSS20] Akash Kumar, C Seshadhri, and Andrew Stolman. Random walks and forbidden minors i: An n^1/2+o(1)-query one-sided tester for minor closed properties on bounded degree graphs. *SIAM Journal on Computing*, (0):FOCS18–216, 2020.

[KSS21] Akash Kumar, C Seshadhri, and Andrew Stolman. Random walks and forbidden minors iii: poly (d/{\epsilon})-time partition oracles for minor-free graph classes. *arXiv preprint arXiv:2102.00556*, 2021. 8

[KUK17] Isabel M. Kloumann, Johan Ugander, and Jon Kleinberg. Block models and personalized pagerank. *Proceedings of the National Academy of Sciences (PNAS)*, 114(1):33–38, 2017. 93

[Kur30] K. Kuratowski. Sur le problème des courbes gauches en topologie. *Fundamenta Mathematica*, 15:271–283, 1930. 9, 55

[LHBH15] Yixuan Li, Kun He, David Bindel, and John E Hopcroft. Uncovering the small community structure in large networks: A local spectral approach. In *Proceedings of the 24th international conference on world wide web*, pages 658–668, 2015. 115

[LK14] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014. 130

[LLC10] Ryan N Lichtenwalter, Jake T Lussier, and Nitesh V Chawla. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 243–252, 2010. 126

[LM14] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014. 114

[LNK07] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007. 126

[Lou20] Andreas Loukas. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representations*, 2020. 110, 115

[Lov06] L. Lovász. Graph minor theory. *Bulletin of the American Mathematical Society*, 43(1):75–86, 2006. 11

[LR15] Reut Levi and Dana Ron. A quasi-polynomial time partition oracle for graphs with an excluded minor. *ACM Transactions on Algorithms (TALG)*, 11(3):24, 2015. 11, 56, 58

[LRU20] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive data sets*. Cambridge university press, 2020. 110

[LS90a] László Lovász and Miklós Simonovits. The mixing rate of markov chains, an isoperimetric inequality, and computing the volume. In *Proceedings [1990] 31st annual symposium on foundations of computer science*, pages 346–354. IEEE, 1990. 5, 12, 17, 38, 40, 41

[LS90b] László Lovász and Miklós Simonovits. The mixing rate of markov chains, an isoperimetric inequality, and computing the volume. In *focs*, pages 346–354, 1990. 59, 60, 82

[LS93] László Lovász and Miklós Simonovits. Random walks in a convex body and an improved volume algorithm. *Random structures & algorithms*, 4(4):359–412, 1993. 5

[lsi19] Latent semantic analysis. https://en.wikipedia.org/wiki/Latent_semantic_analysis, 2019. 96

[LT80] Richard J. Lipton and Robert Endre Tarjan. Applications of a planar separator theorem. *SIAM J. Comput.*, 9(3):615–627, 1980. 55

[LV99] László Lovász and Katalin Vesztergombi. Geometric representations of graphs. *Paul Erdos and his Mathematics*, 1999. 115

[mat] Matlab glmfit function. https://www.mathworks.com/help/stats/glmfit.html. 106

[McS01] Frank McSherry. Spectral partitioning of random graphs. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 529–537. IEEE, 2001. 115

[ME11] Aditya Krishna Menon and Charles Elkan. Link prediction via matrix factorization. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 437–452. Springer, 2011. 126

[Mur21] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2021. 110

[n2va] Node2vec c++ code. https://github.com/snap-stanford/snap/tree/master/examples/node2vec. 107

[n2vb] Node2vec code. https://github.com/eliorc/node2vec. 107

[New03] M. E. J. Newman. The structure and function of complex networks. *SIAM REVIEW*, 45:167–256, 2003. 92

[NJW02] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002. 115

[NS10] Asaf Nachmias and Asaf Shapira. Testing the expansion of a graph. *Inf. Comput.*, 208(4):309–314, 2010. 12

[NS13] Ilan Newman and Christian Sohler. Every property of hyperfinite graphs is testable. *sicomp*, 42(3):1095–1112, 2013. 47, 56, 57, 58

[OCP+16] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1105–1114. ACM, 2016. 110, 111

[PARS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *kdd*, pages 701–710, 2014. 7, 92, 95, 96, 106, 110, 111, 114

[QDM+18] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining - WSDM '18*. ACM Press, 2018. 7, 110, 111, 114

[RA15] Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. 130

[RAL07] Fan Chung Reid Andersen and Kevin Lang. Using pagerank to locally partition a graph. *Internet Mathematics*, 4:35–64, 2007. 120

[RKS20] Benedek Rozemberczki, Oliver Kiss, and Rik Sarkar. Karate Club: An API Oriented Open-source Python Framework for Unsupervised Learning on Graphs. In *Proceedings of the 29th ACM International on Conference on Information and Knowledge Management (CIKM '20)*. ACM, 2020. 119, 120, 121

[RS95a] N. Robertson and P. D. Seymour. Graph minors. XII. Distance on a surface. *Journal of Combinatorial Theory Series B*, 64(2):240–272, 1995. 55

[RS95b] N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory Series B*, 63(1):65–110, 1995. 55

[RS04] N. Robertson and P. D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory Series B*, 92(1):325–357, 2004. 9, 10, 55

[SCW+10] A. Sala, L. Cao, C. Wilson, R. Zablit, H. Zheng, and B. Y. Zhao. Measurement-calibrated graph models for social network experiments. In *www*, pages 861–870, 2010. 93

[SKP12] C. Seshadhri, Tamara G. Kolda, and Ali Pinar. Community structure and scale-free collections of Erdös-Rényi graphs. *Physical Review E*, 85(5):056109, May 2012. 93

[SNA19] SNAP. Stanford Network Analysis Project, 2019. Available at `http://snap.stanford.edu/`. 104

[Spia] D. Spielman. Lecture notes on spectral graph theory. http://www.cs.yale.edu/homes/spielman/eigs/. 40, 82

[Spib] Daniel A. Spielman. *Spectral and Algebraic Graph Theory: Incomplete Draft, dated December 4, 2019.* 5

[SSG17] Aneesh Sharma, C. Seshadhri, and Ashish Goel. When hashes met wedges: A distributed algorithm for finding high similarity vectors. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 431–440, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee. 120

[SSSG20a] C Seshadhri, Aneesh Sharma, Andrew Stolman, and Ashish Goel. The impossibility of low-rank representations for triangle-rich complex networks. *Proceedings of the National Academy of Sciences*, 117(11):5631–5637, 2020. 8

[SSSG20b] C. Seshadhri, Aneesh Sharma, Andrew Stolman, and Ashish Goel. The impossibility of low-rank representations for triangle-rich complex networks. *Proceedings of the National Academy of Sciences*, 117(11):5631–5637, 2020. 110, 113, 115, 123

[ST04] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '04, page 81–90, New York, NY, USA, 2004. Association for Computing Machinery. 5, 6, 7

[ST12] D. Spielman and S.-H. Teng. A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. *sicomp*, 42(1):1–26, 2012. 12, 16, 38, 40, 58, 59, 60, 82

[ST13] Daniel A Spielman and Shang-Hua 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. 5

[str19] String database. http://version10.string-db.org/, 2019. 104

[Swa14] K. Swanapoel. The rank lemma. https://konradswanepoel.wordpress.com/2014/03/04/the-rank-lemma/, 2014. 97, 98

[Tao13] T. Tao. A cheap version of the kabatjanskii-levenstein bound for almost orthogonal vectors. https://terrytao.wordpress.com/2013/07/18/a-cheap-version-of-the-kabatjanskii-levenstein-bound-for-almost-orthogonal-vectors/, 2013. 99

[TQW+15a] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *www*, pages 1067–1077, 2015. 92

[TQW+15b] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077, 2015. 114

[Tre05] Luca Trevisan. Approximation algorithms for unique games. In *focs*, pages 197–205. IEEE, 2005. 12

[TW17] N Kipf Thomas and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2, 2017. 114, 115

[Twi18] Embeddings@twitter. https://blog.twitter.com/engineering/en_us/topics/insights/2018/embeddingsattwitter.html, 2018. 92, 109

[VCC+18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. 115

[Wag37] K. Wagner. Über eine eigenschaft der ebenen komplexe. *Mathematische Annalen*, 114:570–590, 1937. 9, 55

[WF94] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994. 92

[WS98] D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998. 92, 93

[XHLJ19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. 115

[YI15] Yuichi Yoshida and Hiro Ito. Testing outerplanarity of bounded degree graphs. *Algorithmica*, 73(1):1–20, 2015. 11, 59

[YLC15] Yang Yang, Ryan N Lichtenwalter, and Nitesh V Chawla. Evaluating link prediction methods. *Knowledge and Information Systems*, 45(3):751–782, 2015. 126

[YS07] Stephen J. Young and Edward R. Scheinerman. Random dot product graph models for social networks. In *Algorithms and Models for the Web-Graph*, pages 138–149, 2007. 93