Finding Dense Structures in Graphs and Matrices

Aditya Bhaskara

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Computer Science
Adviser: Moses S. Charikar

September 2012

© Copyright by Aditya Bhaskara, 2012.

All rights reserved.

Abstract

We will study several questions with a common theme of finding *structure* in graphs and matrices. In particular, in graphs we study problems related to finding *dense* induced subgraphs. Many of these questions have been studied extensively, such as the problem of finding large cliques in a graph, and more recently, the small-set-expansion conjecture. Problems of this nature also arise in many contexts in practice, such as in finding *communities* in social networks, and in understanding properties of the web graph.

We then study questions related to the spectra of matrices. Singular values of matrices are used extensively to *extract structure* from matrices (for instance, in principal component analysis). We will study a generalization of the maximum singular value, namely the $q \mapsto p$ *norm* of a matrix A (denoted $\|A\|_{q\mapsto p}$) and the complexity of approximating this quantity. The question of approximating $\|A\|_{q\mapsto p}$ turns out to have many flavors for different values of p, q, which we will explore in detail.

The technical contributions of the thesis can be summarized as follows:

1. We study in detail the *densest k-subgraph* problem. Given a graph G, the aim is to find an induced subgraph on k vertices with as many edges as possible. The approximability of densest k-subgraph is a notorious open problem, with the best algorithms having a *polynomial* approximation ratio, i.e., n^c for some constant c, while the best hardness results rule out a small constant factor (roughly 1.4) approximation.

   In the thesis, we will present the best known algorithm for the problem, which gives roughly an $n^{1/4}$ factor approximation. Further, our algorithm and its analysis point to a simple average case (or *planted*) version of the problem, which seems beyond our current algorithmic techniques.

2. Next, we explore the complexity of computing $\|A\|_{q\mapsto p}$ of matrices A for different ranges of the parameters p, q. For $p \leq q$, we develop a better understanding of the complexity: in particular, for an important special case in which A has non-negative entries, we give a polynomial time algorithm to compute the norm up to any precision. We also prove that without such a restriction, the problem is hard to approximate to an *almost polynomial* factor.

   For $p > q$, these quantities are called *hypercontractive* norms, and computing these would have applications to questions such as certifying the 'restricted isometry property', and to certifying small-set expansion.

3. Finally, we propose and study a problem which can be seen as a 'matrix variant' of the so-called maximum density subgraph problem for graphs. We will call this QP-Ratio – a 'ratio' version of the familiar quadratic programming problem. The problem is a close cousin of many questions we study in the thesis, and it seems to highlight the difficulty in capturing the constraint $x_i \in \{-1, 0, 1\}$ using convex programming relaxations.

Acknowledgements

"Five years have past; five summers with the length of five long winters ...", as the poet said. The last five years have been a wonderful experience, both academically and otherwise, and I would like to thank some of the several people who have contributed to it. I have been fortunate to have the advice of many brilliant researchers, and the undying support of my friends and family to keep my spirits up throughout my stay at Princeton.

First of all, I would like to thank my advisor Moses Charikar for his wonderful guidance and support throughout the five years. His clarity of thought and excellent intuition about different kinds of problems have been a great source of inspiration. The calm and confident attitude towards research, and the ability to come up with the right question, are qualities I have admired greatly, and I can only hope that some of it has rubbed off on me. His persistent attitude of "what is the simplest case we cannot solve?" shows itself in many places in this thesis. I would also like to thank Moses for introducing me to many researchers right from the early days – much of my work is due to discussions and collaboration with them.

Sanjeev Arora has been a great source of inspiration and guidance throughout my stay at Princeton. I would like to thank him for the several discussions about my research, and at many times, for guiding the directions it took. His grand view of theory and its place in science as a whole has often been of great source of inspiration to me personally. I feel fortunate for having the opportunity to interact with him, and in hindsight, somewhat regret not being able to pursue some of the research directions he suggested due to various reasons.

Much of my interest in theory and the appreciation of its beauty began with my mentors at IIT Bombay during my undergraduate studies. I would like to thank Abhiram Ranade, Ajit Diwan, Sundar Vishwanathan, Krishna S, and many others for their excellent courses. I would also like to thank them for all the advice, and more importantly, for giving me a first perspective towards the field.

Next, I would like to thank my mentors during my internships. At MSR India, I had the opportunity to interact with Ravi Kannan, whose sharp intuition and breadth of knowledge have always amazed me. His ability to listen to a problem and attack it from first principles is something I found remarkable, and I hope to have more opportunities to interact with him.

At MSR Silicon Valley, I had the pleasure of meeting Kunal Talwar. His friendly nature and interest in many kinds of problems led to some nice collaborations, and I thoroughly enjoyed working with him. I would also like to thank Udi Wieder and Uriel Feige for several interesting discussions during my stay at MSR.

I would like to thank the many faculty for making Princeton a wonderful place to do theory. I would also like to thank my fellow students for making the experience more fun, with the great discussions and coffee. Special thanks are due to Rajsekar Manokaran, David Steurer, Moritz Hardt, Sina Jafarpour, Sushant Sachdeva, Huy Nguyen, Hossein Bateni, Eden Chlamtac, Shi Li, and many others. I thank Eden Chlamtac, Seshadri Comandur and Wolfgang Mulzer for all the advice during the initial phases of grad school.

In the spirit of saving the best for the last(!), a notable absentee from the list above is Aravindan Vijayaraghavan, who has been a wonderful friend and collaborator throughout my stay at Princeton. I would like to thank him for the innumerable discussions about theory, life and almost everything which we've had over the years.

I have also been very fortunate to have great collaborators, and would like to thank them: Sanjeev Arora, Moses Charikar, Eden Chlamtac, Daniel Dadush, Devendra Desai, Uriel Feige, Venkatesan Guruswami, Ravi Kannan, Ravishankar Krishnaswamy, Rajsekar Manokaran, Srikanth Srinivasan, Kunal Talwar, Aravindan Vijayaraghavan, Udi Wieder and Yuan Zhou. Special thanks are due to Eden Chlamtac, Aravindan, and Ravishankar Krishnaswamy for being great friends and for sharing many 'emotional' moments before deadlines.

Life in the department has been great fun due to the many wonderful people. I would in particular like to thank Anirudh Badam, Ana and CJ Bell, Chris Park, Sid Sen, Indraneel Mukherjee, Srinivas Narayana, and several others for sharing many great moments, and helping me out on several occasions.

From an administrative point of view, life at Princeton has been a breeze, and I would like to thank Melissa Lawson and Mitra Kelly for being so helpful and putting up with my pestering on several occasions! I also sincerely acknowledge the following NSF grants which funded me through my stay: MSPA-MCS 0528414, CCF 0832797, and AF 0916218.

Finally, I would like to thank several of my friends for a happy life outside the department. My apartment mates over the years: Arul, who I stayed with for two years and was a constant source of good conversation and great movie recommendations; Aravindan, Anirudh and Rajsekar, who were always ready to help out and great fun to share a place with. I also thank some of the friends who, over the years have influenced me quite a bit, and made the place feel like home: Ketra, Arthi, Anirudh, Karthikeyan, the Kalyanaramans, Ravi, and many more!

Blood, as they say, runs thicker than water. I would like to thank my friends from IIT Bombay for having remained in touch over the years, and often giving a non-theorist's view of my research. In particular, I'd like to thank Aditya Parameswaran, B. Aditya Prakash, Alekh Agarwal, Rajhans Samdani, and the rest of the gang.

Last, and most important, I would like to thank my family for their affection and support throughout my stay away from home. My parents for their constant love, and putting up with my tendency to not answer phone calls! My sister for being the ideal elder sibling, and the constant go-to person for advice. It is to them that I dedicate this thesis.

To my family.

Contents

Chapter 1

Introduction

We will study theoretical questions with the general motivation of finding "structure" in large matrices and graphs. With the increasing amout of data available for analysis, this question is becoming increasingly crucial. Thus indentifying the essential problems and understanding their complexity is an important challenge.

The thesis broadly studies two themes: the first is on finding dense subgraphs, which is a crucial subroutine in many algorithms for the clustering and partitioning of graphs. The second is on problems related to *finding structure* in matrices. In particular, we will study generalizations of singular vectors and problems related to graph spectra. There are also intimate connections between these two themes, which we will explore.

From the point of view of complexity, many problems in these areas are surprisingly ill-understood, i.e., the algorithmic results are often very far from known inapproximability results, despite a lot of effort on both fronts. To cope with this, many average case hardness assumptions have been proposed, and a study of these will play a crucial part in the thesis.

1.1 Background and overview

Let us now describe the two themes and try to place our results in context.

1.1.1 Dense subgraphs

Finding *dense* structures in graphs has been a subject of study since the origins of graph theory. Finding large cliques in graphs is one of the basic questions whose complexity has been thoroughly explored. From a practical standpoint, many questions related to clustering of graphs involve dividing the graph into small pieces with many edges inside the pieces and not many edges across. A useful primitive in these algorithms is finding dense subgraphs.

Recently, there has been a lot of interest in understanding the *structure* of graphs which arise in different applications – from social networks, to protein interaction graphs, to the world wide web. Dense subgraphs can give valuable information about interaction in these networks, be it sociological or an improved understanding of biological systems.

There have been many formulations to capture the objectives in these applications. One which is quite natural is the maximum density subgraph problem (see below, Section 1.2.1 for the formal definition). Here the aim is to find a subgraph with the maximum *density*, which is the ratio of the number of edges to vertices. Such a subgraph can in fact be found efficiently (in poly time).

However, in many realistic settings, we are interested in a variant of the problem in which we also have a bound on the *size* of the subgraph we output – i.e., at most k vertices, for some parameter k (for instance, because the subgraph of maximum density might not say much useful about the structure of the graph). This variant is called the Densest k-subgraph problem (or DkS, which is defined formally in Section 1.2.2), and has been well-studied in both theory and practice.

The approximability of DkS is an important open problem and despite much work, there remains a significant gap between the currently best known upper and lower bounds. The DkS problem will be one of the main protagonists in the thesis, and our contributions here are as follows: first, we give the best known approximation algorithms for the problem. Second, our results suggest the importance of studying an average case, or *planted* version of the problem (See section 2.4). This leads us to a simple distribution over instances on which our current algorithmic arsenal seems unsuccessful. In fact, some works (see Section 2.4.4) have explored the consequences of assuming the problem to be *hard* on these (or slight variants of these) distributions.

From an algorithm design point of view, coming up with such distributions is important because they act as testbeds for new algorithms. Much progress on questions such as unique games has arisen out of a quest to come up with "hard instances" for current algorithms, and try to develop new tools to deal with these. (For instance, [11, 8, 14]).

We will formally describe our contributions in Section 1.3.

1.1.2 Structure in matrices

We will also study questions with the general goal being to *extract structure* from matrices, or objects represented as matrices. Many questions of this kind, such as low rank approximations, have been studied extensively. The spectrum (set of eigenvalues) of a matrix plays a key role in these problems, and we will explore questions which are related to this.

Spectral algorithms have recently found extensive applications in Computer Science. For instance, spectral partitioning is a tool which has been very successful in practical problems such as image segmentation [69] and clustering [53]. The Singular Value Decomposition (SVD) of matrices is used extensively in machine learning, for instance, in extracting features and classifying documents. In theory, graph spectra and their extensions have had diverse applications, such as the analysis of Markov chains [52] and graph partitioning [1].

The first problem we study is a generalization of singular values. It is the question of computing the $q \mapsto p$ operator norm of a matrix (defined formally below, Section 1.2.5). Apart from being a natural optimization problem, being able to approximate this norm has some interesting consequences: we will give an application to 'oblivous routing' with an ℓ_p norm objective (Section 5.4). We will also see how computing so-called *hypercontractive norms* have an application in compressed sensing, namely for the problem of certifying that a matrix satisfies the "restricted isometry" property (Section 5.6.1). Computing such norms is also related to the so-called 'small set expansion' problem (defined below, Section 1.2.3) [14].

We try to understand the complexity of computing $q \mapsto p$ norms. The problem has very different *flavors* for different values of p, q. For instances, it generalizes the largest singular value, which is has a 'continuous optimization' feel, and the *cut norm* which has a 'CSP' flavor. For $p \leq q$, we give a *characterization* of the complexity of computing $q \mapsto p$ norms: we refer to Section 1.3 for the details.

Second, we propose and study a new problem called QP-Ratio (see Section 1.2.6 for the definition), which we can view as a *matrix analogue* of the maximum density subgraph problem we encountered in the context of graphs. It can also be seen as a ratio version of the familiar quadratic programming problem (hence the name). Our interest in the problem is two-pronged: first, it is a somewhat natural cousin of well-studied problems. Second, and more important, familiar tools to obtain convex programming relaxations appear to perform poorly for such *ratio* objectives, and thus the goal is to develop convex relaxations to capture such questions.

Let us elaborate on this: semidefinite programming has had reasonable success in capturing *numeric* problems (such as quadratic programming) subject to $x_i \in \{0, 1\}$ or $x_i \in \{-1, 1\}$ type constraints. Can we do the same with an $x_i \in \{-1, 0, 1\}$

constraint? To our knowledge, this seems quite challenging, and once again, it turns out to be a question in which the gap in our understanding is quite wide (between upper and lower bounds). Furthermore, there is an easy-to-describe *hard distribution* which seems beyond our present algorithmic toolkit. In this sense, the situation is rather similar to the DkS problem.

Hardness results and a tale of many conjectures. Although the thesis will focus primarily on algorithmic results, we stress that for many of the questions we consider, proving hardness results based on $P \neq NP$ has proven extremely challenging. However the questions are closely related to two recent conjectures, namely small-set expansion (which says that a certain expansion problem is hard to approximate) and Feige's *Random k-SAT* hypothesis (which says that max k-SAT is hard to approximate up to certain parameters, even when the clause-literal graph is generated *at random*).

To what extent dow we believe these conjectures? Further, how do we *compare* the two assumptions? The former question is believed to be essentially equivalent to the unique games conjecture [8]. It is believed to be hard (in that it does not have poly time algorithms), however it has sub-exponential time algorithms (at least for a well-studied range of parameters). The second problem (random k-SAT) is *harder* in this sense – current algorithmic tools (for instance, linear and semidefinite programming relaxations, see Section 1.2.7) do not seem to help in this case, and there is no known sub-exponential time algorithm for it. In this sense, this hardness assumption may be more justified. However we have very limited understanding of how to prove the average case hardness of problems, and hence we seem far from proving the validity of the hypothesis.

1.2 Dramatis personae

In order to give a preview of our results, we now introduce the various characters (problems and techniques) that play central roles in this thesis. In the subsequent chapters, we will study them in greater detail, and explain the contexts in which they arise. The various connections between these problems will also become apparent later.

1.2.1 Maximum Density Subgraph

Given a graph $G = (V, E)$, the problem is to find a subgraph H so as to maximize the ratio $|E(H)|/|V(H)|$, where $E(H)$ and $V(H)$ denote, respectively the edges and vertices in the subgraph H.

This problem can be solved in polynomial time, and we outline algorithms in Section 2.3.

1.2.2 Densest k-Subgraph (DkS)

This is a *budgeted* version of the maximum density subgraph. Here, given G and a parameter k, the problem is to find a subgraph H on at most k vertices so as to maximize $E(H)$. This *hard constraint* on the size of the subgraph is what makes the problem difficult.

Our results on this problem are outlined in detail in Section 1.3, and we will discuss algorithms and the complexity of DkS in detail in Chapter 3.

1.2.3 Small Set Expansion (SSE)

This problem is closely related to graph expansion and to DkS. One version of it is the following: let $\varepsilon, \delta > 0$ be parameters. Given a graph G and the promise that there exists a *small* (i.e., at most δn) sized set S which does not *expand*, i.e.,

$|E(S, V \setminus S)| \leq \varepsilon |E(S,V)|$, find a set T of size at most δn with expansion at most $9/10$ i.e., a set with at least $1/10$ of its edges staying inside.

The small set expansion conjecture states that for any $\varepsilon > 0$, there exists a $\delta > 0$ (small enough) such that it is hard to solve the above problem in polynomial time. A lot of recent work [8, 66] has studied this question and its connection to the unique games conjecture.

1.2.4 Random Graph Models

A very useful source of intuition for the problems we consider in the thesis is the analysis of random graphs. The main model we use is the Erdős-Renyi model (also called $G(n,p)$).

We say that a graph G is "drawn from" $G(n,p)$ if it is generated by the following random process (so formally, the graph is a "random variable"): we fix n vertices indexed by $[n]$, and an edge is placed between each pair i, j with probability p indepedent of all other pairs.

Many "generative" models much more complicated than this have been studied for understanding partitioning problems, but we will not encounter them much in the thesis.

1.2.5 Operator Norms of Matrices

Consider a matrix $A \in \Re^{m \times n}$. We can view it as an operator $A : \Re^n \mapsto \Re^m$. We will study the $\ell_q \mapsto \ell_p$ norm of this operator. More precisely, we wish to compute the maximum *stretch* (in the ℓ_p norm) caused by A to a unit vector (in the ℓ_q norm). Formally,

$$\|A\|_{q \mapsto p} := \max_{x \in \Re^n, x \neq 0} \frac{\|Ax\|_p}{\|x\|_q}.$$

Operator norms arise in various contexts, and they generalize, for instance, the maximum singular value ($p = q = 2$), and the so-called *Gröthendieck problem* ($q = \infty, p = 1$). We will study the complexity of approximating the value of $\|A\|_{q \mapsto p}$ for different values of p, q and show how the problem has very different flavors for different p, q.

1.2.6 Quadratic Programming

Given an $n \times n$ real matrix A, The problem of *quadratic programming* is to find

$$QP(A) := \max_{x_i \in \{-1,1\}} \sum_{i,j} a_{ij} x_i x_j .$$

The best known approximation algorithm has a ratio of $O(\log n)$ [60], which is also essentially optimal [9].

We will study a hybrid of this and the Maximum density subgraph problem, which we call *QP-Ratio*. Formally given $n \times n$ matrix A as before,

$$\textsf{QP-Ratio}: \quad \max_{\{-1,0,1\}^n} \frac{\sum_{i \neq j} a_{ij} x_i x_j}{\sum x_i^2} \tag{1.1}$$

1.2.7 Lift and Project Methods

Linear (LP) and Semidefinite programming (SDP) relaxations have been used extensively in approximation algorithms, and we will assume familiarity with these ideas. Starting with early work on cutting plane methods, there have been attempts to *strengthen* relaxations by adding more constraints.

Recently, more systematic ways of obtaining relaxations have been studied, and these are called LP and SDP *hierarchies.* (Because they give a hierarchy of relaxations, starting with a *basic* LP/SDP and converging to the integer polytope of the solutions). They will not be strictly necessary in understanding out results, but might help placing some of them in context. We refer to a recent survey by Chlamtac and Tulsiani [30] for details.

1.3 Results in the thesis and a roadmap

We now outline the results presented in the chapters to follow. We will also point to the papers in which these results first appeared, and the differences in presentation we have chosen to make.

Densest k-subgraph. We will give an algorithm for densest k-subgraph with an approximation factor of $O(n^{1/4+\varepsilon})$ and running time $n^{O(1/\varepsilon)}$. The algorithm is motivated by studying an average case version of the problem, called the Dense vs. Random question. We will outline the problem and discuss its complexity in Chapters 2 and 3.

A bulk of this material is from joint work with Charikar, Chlamtac, Feige and Vijayaraghavan [17]. The main algorithm is presented in [17] as a rounding algorithm starting with the Sherali Adam *lift* of the standard linear program for DkS. In the thesis, we choose to present a fully combinatorial algorithm which could be of independent interest, even though it is on exactly the same lines as the LP-based algorithm.

Matrix norms. We will study the approximability of $q \mapsto p$ norms of matrices in Chapter 5. The main results are the following: we give an algorithm for non-negative matrices which converges in polynomial time for the case $p \leq q$. For this range, we also prove strong inapproximability results when we do not have the non-negativity restriction. These results are joint work with Vijayaraghavan [20]. We will also briefly study hypercontractive norms (the case $p > q$), discuss questions related to computing them, and outline some recent work on the problem due to others.

Ratio variant of quadratic programming. Finally, we study the QP-Ratio problem in Chapter 6. We will see an $O(n^{1/3})$ factor approximation for the problem using an SDP based algorithm. We also point out why it is difficult to capture this problem using convex programs, and give various evidences for its hardness. This is joint work with Charikar, Manokaran and Vijayaraghavan [19].

Chapters 2 and 4 introduce the problems we study in greater detail and give the necessary background.

Chapter 2

Finding Dense Subgraphs and Applications

We start with discussing questions related to finding *dense* subgraphs, i.e., sets of vertices in a graph s.t. the induced subgraph has *many* edges. Such problems arise in many contexts, and are important from both a theoretical and a practical standpoint.

In this chapter, we will survey different problems of this flavor, and the relationships between them. We will also discuss known results about these, and the main challenges. A question which we will highlight is the Densest k-subgraph (DkS) problem, for which we will outline the known results and our contributions.

2.1 Motivation and applications

We will describe a couple of the algorithmic applications we outlined in the introduction (from social networks and web graphs). They shed light into the kind of formalizations of these questions we should try to study.

A lot of data is now available from 'social networks' such as Facebook. These are graphs in which the vertices represent members (people) of the network and edges represent relationships (such as being friends). A very important problem in this setting is that of finding "communities" (i.e., finding a set of people who share, for instance, a common interest). Empirically, a community has more edges than typical subgraphs in the graph of this size (we expect, for instance, more people in a community to be friends with each other).

Thus finding communities is precisely the problem of finding vertices in a graph with many edges (i.e., dense subgraphs). This line of thought has been explored in many works over the course of the last decade or so, and we only point to a few [33, 10, 44]

A second application is in the study of the web graph – this is the graph of pages on the world wide web, with the edges being links between pages (formally, it is a directed graph). The graph structure of the web has been very successful in extracting many useful properties of webpages. One of the principal ones is to guage the "importance" or "popularity" of a page based on how many pages link to it (and recursively, how many *important* pages link to it). This notion, called the *page rank* (see [57, 25]) has been extremely successful in search engines (in which the main problem is to show the *most relevant* search results).

One of the loop-holes in this method, is that an adversary could create a collection of pages which have an abnormally high number of links *between them* (and some links outside), and this would end up giving a very high pagerank to these pages (thus placing them on top of search results!). To combat this, the idea proposed by Kumar *et al.* [58] is to find small subgraphs which are *too dense*, and label these as candidates for "link spam" (i.e., the spurious edges).

These are a couple of the *algorithmic* applications. As we mentioned in the introduction, the *inability* to solve these problems also has 'applications'. We will discuss a couple of recent works in this direction in Section 2.4.4.

We will now study several questions with this common theme, and survey various results which are known about them.

2.2 Finding cliques

The decision problem of finding a CLIQUE of a specified size in a graph is one of the classic NP complete problems [43]. The approximability of CLIQUE (finding a clique of size "close" to the size of the maximum clique) has also been explored in detail. In a sequence of works culminating with that of Hästad [48], it was shown that it is hard to approximate the size of the largest clique to a factor better than $n^{1-\varepsilon}$, for any constant ε.

While the inapproximability result suggests that "nothing non-trivial" can be done about the clique problem, we mention a rather surprising (folklore) result which is interesting from the point of view of finding dense subgraphs.

Theorem 2.1 (Folklore)**.** *Given a graph G on n vertices with a clique of size k, there exists an algorithm which runs in time $n^{O(\log n/\varepsilon)}$, and returns an* almost clique*, i.e., it returns a subgraph on at least k vertices with minimum degree at least $(1-\varepsilon)k$.*

The algorithm is also quite simple: say we are given $G = (V, E)$. If the minimum degree is at least $(1-\varepsilon)|V|$, return the entire graph. Else, pick some vertex $v \in V$ of degree $< (1-\varepsilon)|V|$, and recurse on two instances defined as follows. The first is the graph obtained by removing v from G. The second is one containing v and its neighborhood, and the induced subgraph on this set. (This is equivalent to guessing if the vertex v is in the clique or not). Thus if there is a clique of size k the algorithm returns an almost clique. The analysis of the running time is a little tricky – it crucially uses the fact that in *one* of the instances in the recursion, the size of the graph drops by a factor $(1-\varepsilon)$.

2.2.1 Planted clique.

Another problem which has been well-studied is a natural "average case" version of CLIQUE. In random graphs $G(n, 1/2)$ (every edge is picked with probability 1/2 i.i.d.), it is easy to argue that the size of the maximum clique is at most $(2+o(1))\log n$. However, it is not known how to distinguish between the following distributions using a polynomial time algorithm:

YES. G is picked from $G(n, 1/2)$, and a clique is *planted* on a random set S of $n^{1/2-\varepsilon}$ vertices. (Here $\varepsilon > 0$ is thought of as a small constant).

NO. G is picked from $G(n, 1/2)$.

In the above, by a clique being *planted*, we mean that we add edges between every pair of vertices in the picked set S. It is known that spectral approaches [5], as well as approaches based on natural semidefinite programming relaxations [38] do not give a polynomial time algorithm for $\varepsilon > 0$. Frieze and Kannan [40] showed that if a certain "tensor maximization" problem could be solved efficiently, then it is possible to break the $n^{1/2}$-barrier. However, the complexity of the tensor maximization problem is also open.

Such *planted* problems will play an important role in our study of finding dense subgraphs. We will define a generalization of planted clique – planted dense subgraph (or the "Dense vs. Random" question) and see how solving it is crucial to making progress on the Densest k-subgraph problem.

2.3 Maximum density subgraph

A natural way to formulate the question of finding dense subgraphs is to find the subgraph of maximum "density". For a subgraph, its density is defined to be the ratio of the number of edges (induced) to the number of vertices (this is also the average degree). We can thus define the "max density subgraph" problem. The objective, given a graph $G = (V, E)$, is to find

$$\max_{S \subseteq V} \frac{E(S,S)}{|S|},$$

where $E(S, S)$ denotes the number of edges with both end points in S.

For this question, it turns out that a flow based algorithm due to Gallo *et al.* [42] can be used to find the optimum exactly. Charikar [26] showed that a very simple greedy algorithm: one which removes the vertex of least degree vertex at each step, and outputs the best of the considered subgraphs, gives a factor 2 approximation. Due to its simplicity, it is often useful in practice.

Another very simple algorithm which gives a factor 2 approximation is to set a *target* density ρ, and repeatedly remove vertices of degree $\rho/2$. It can be shown that if there is a subgraph of density ρ to begin with, we end up with a non-empty subgraph.

2.3.1 Finding *small* subgraphs

While finding subgraphs with a high density is interesting, there are many applications in which we wish to find subgraphs with many edges and are *small*. For instance, in the question of small-set expansion (Section 1.2.3), we need to return a set of at most a certain size.

So also in practice, for instance the example of detecting link spam, the set of pages which "cause" link spam is assumed to be small – for instance we would not want to classify all the webpages belonging to an organization (which typically involve many edges between each other) as link spam. (For e.g., in the implementation in [58] they set a bound of 150 nodes).

Thus a natural question to ask is to find a subgraph with at most a certain number of vertices and as many edges as possible. This was formulated and studied by Feige, Kortsarz and Peleg [35], and is precisely the Densest k-subgraph problem we defined in Section 1.2.2.

2.4 Densest k-Subgraph

The DkS problem can also seen as an optimization version of the decision problem CLIQUE. Since it is one of the main protagonists in the thesis, we now discuss earlier work on the problem, and our contributions. Also we will see the relevance of understanding the complexity of the problem by showing connections to other well-studied problems.

2.4.1 Earlier algorithmic approaches

As mentioned above, the problem was studied from an algorithmic point of view by [35]. They gave an algorithm with an approximation ratio of $n^{1/3-\varepsilon}$ for a small constant $\varepsilon > 0$ (which has been estimated to be roughly 1/60). The algorithm is a combination (picking the best) of five different (all combinatorial) algorithms, each of which performs better than the rest for a certain range of the parameters.

Other known approximation algorithms have approximation guarantees that depend on the parameter k. The greedy heuristic of Asahiro et al. [12] obtains an $O(n/k)$ approximation. Linear and semidefinite programming (SDP) relaxations were studied by Srivastav and Wolf [71] and by Feige and Seltser [36], where the latter authors showed that the integrality gap of the natural SDP relaxation is $\Omega(n^{1/3})$ in the worst case. In practice, many heuristics have been studied for the problem, mostly using greedy and spectral methods.

2.4.2 Our contributions

One of our main results in this thesis is a polynomial time $O(n^{1/4+\varepsilon})$ approximation algorithm for DkS, for any constant $\varepsilon > 0$. More specifically, given $\varepsilon > 0$, and a graph G with a k-subgraph of density d, our algorithm outputs a k-subgraph of density $\Omega\left(d/n^{1/4+\varepsilon}\right)$ in time $n^{O(1/\varepsilon)}$. In particular, our techniques give an $\widetilde{O}(n^{1/4})$-approximation algorithm running in $O(n^{\log n})$ time.

Even though the improvement in the approximation factor is not dramatic, we believe our methods shed new light on the problem. In particular, our algorithm for DkS is inspired by studying an average-case version we call the 'Dense vs Random' question (see Section 3.2.2 for a precise definition). Here the aim is to distinguish random graphs (which do not whp contain dense subgraphs) from random graphs with a *planted* dense subgraphs (similar to the planted clque problem of Section 2.2.1). Thus we can view this as the question of efficiently *certifying* that random graphs do not contain dense subgraphs. Our results suggest that these random instances the *most difficult* for DkS, and thus a better understanding of this planted question is crucial for further progress on DkS.

Broadly speaking, our algorithms involve cleverly counting appropriately defined subgraphs of constant size in G, and use these counts to identify the vertices of the dense subgraph. A key notion which comes up in the analysis is the following:

Definition 2.2. *The* log-density *of a graph $G(V, E)$ with average degree D is $\log_{|V|} D$. In other words, if a graph has log-density α, its average degree is $|V|^{\alpha}$.* [1]

In the Dense vs Random problem alluded to above, we try to distinguish between G drawn from $G(n, p)$, and G drawn from $G(n, p)$ with a k-subgraph H of certain density planted in it. The question then is, how dense should H be so that we can distinguish between the two cases w.h.p.?

We prove that the important parameter here is the *log density.* In particular, if the log-density of G is α and that of H is β, with $\beta > \alpha$, we can solve the distinguishing problem in time $n^{O(1/(\beta-\alpha))}$. Our main technical contribution is that a result of this nature can be proven for arbitrary graphs.

[1] We will ignore low order terms when expressing the log-density. For example, graphs with constant average degree will be said to have log-density 0, and cliques will be said to have log-density 1.

Maximum Log-density Subgraph. Our results can thus be viewed as attempting to find the subgraph of G with the maximum *log density*, i.e., the subgraph H that maximizes the ratio $\log|E(H)|/\log|V(H)|$. This is similar in form to the Maximum density subgraph question 2.3, but is much stronger. We note that as such, our results do not give an algorithm for this problem, and we pose it as an interesting open question.

Open Problem 2.3. *Is there a polynomial time algorithm to compute the* maximum log-density *subgraph? I.e., the following quantity*

$$\max_{H \subseteq G} \frac{\log|E(H)|}{\log|V(H)|}?$$

2.4.3 Related problems

A problem which is similar in feel to DkS is the small set expansion conjecture (SSE), which we defined in Section 1.2.3. Note the following basic observation

Observation 2.4. *A constant factor approximation algorithm for DkS implies we can solve the SSE problem (as stated in Section 1.2.3).*

This is because a C-factor approximation for DkS implies that we can find δn sized subset with at least $\frac{1}{C} \cdot (1-\varepsilon)nd$ edges, which is what we need to find.

Thus the SSE problem is in some sense *easier* than DkS. (Reductions to other statements of SSE can be found in [67]). Furthermore, lift and project methods have been successful to solve SSE in subexponential time [15], however such methods do not seem to help for DkS [18].

Charikar *et al.* [27] recently showed an approximation preserving reduction from DkS to the maximization version of Label cover (called Max-REP), for which they obtained an $O(n^{1/3})$ approximation. I.e., they proved that Max-REP is at least as hard as DkS.

2.4.4 Towards hardness results

In addition to being NP-hard (as seen from the connection to CLIQUE), the DkS problem has also been shown not to admit a PTAS under various complexity theoretic assumptions. Feige [37] showed this assuming random 3-SAT formulas are hard to refute, while more recently this was shown by Khot [55] assuming that NP does not have randomized algorithms that run in sub-exponential time (i.e. that $NP \not\subseteq \cap_{\varepsilon>0} BPTIME(2^{n^{\varepsilon}})$).

Recently, Manokaran *et al.* [3] showed that it is hard to approximate DkS to any constant factor assuming the stronger Max-k-AND hypothesis of Feige [37]. Though this is a somewhat more non-standard assumption, it is believed to be *harder* than assumptions such as the Unique games conjecture (or SSE). [For instance, $n^{1-\varepsilon}$ rounds of the Lasserre hierarchy do not help *break* this assumption – see a recent blog post by Barak comparing such assumptions [13].]

Note however, that the best hardness results attempt to rule out constant factor approximation algorithms, while the best algorithms we know give an $O(n^{1/4})$ factor approximation. So it is natural to ask where the truth lies! We conjecture that it is impossible to approximate DkS to a factor better than n^{ε} (some constant $\varepsilon > 0$) in polynomial time under a reasonable complexity assumption.

One evidence for this conjecture is that strong Linear and Semidefinite relaxations (which seem to capture many known algorithmic techniques) do not seem to help. In a recent work, Guruswami and Zhou (published together with results on LP hierarchies in [18]) showed that the integrality gap for DkS remains $n^{\Omega(1)}$ even after $n^{1-\varepsilon}$ rounds of the Lasserre hierarchy. This suggests that approximating DkS to a factor say polylog(n) may be a much harder problem, than say Unique games (or SSE).

2.4.5 Hardness on average, and the consequences

Let us recall the state of affairs in our understanding of the Densest k-subgraph problem: even for the Dense vs. Random question (which is an average case version of DkS), existing techniques seem to fail if we want to obtain a "distinguishing ratio" better than $n^{1/4}$. While this may cause despair to an algorithm designer, the average-case hardness of a problem is good news for cryptography. Public key cryptosystems are often based on problems for which it is easy to come up with *hard instances.*

The recent paper of Applebaum *et al.* does precisely this starting with the assumption that the planted densest subgraph (a bipartite variant of the Dense vs. Random problem we study) is computationally hard.

In a very different setting, Arora *et al.* [7] recently use a similar assumption to demonstrate that detecting malice in the pricing of financial derivatives is computationally hard. I.e., a firm which prices derivatives could gain unfairly by bundling certain goods together, while it is computationally difficult to *certify* that the firm deviated from a random bundling.

The applications of these hardness assumptions provides additional motivation for the study of algorithms for these problems. We will now present our algorithmic results for the DkS problem.

Chapter 3

The Densest k-Subgraph Problem

We begin with some definitions, and set up notation which we use for the remainder of the chapter. We then start out with the description and analysis of our algorithm. As outlined earlier, the ideas are inspired by an average case version of the problem. This is described first, in Section 3.2.2, followed by the general algorithm in Section 3.3.

Along the way, we will see the "bottlenecks" in our approach, as well as ways to get around them if we allow more running time. More specifically, we will analyze a trade-off between the run time and the approximation factor which can be obtained by a simple modification of the above algorithm. We then end with a comment on spectral approaches. These are simple to analyze for average case versions of the problem (in some cases they beat the bounds obtained by the previous approach).

3.1 Notation

Let us introduce some notation which will be used in the remainder of this chapter. Unless otherwise stated, $G = (V, E)$ refers to an input graph on n vertices, and k refers to the size of the subgraph we are required to output. Also, $H = (V_H, E_H)$ will denote the densest k-subgraph (breaking ties arbitrarily) in G, and d denotes the average degree of H. For $v \in V$, $\Gamma(v)$ denotes the set of neighbors of v, and $\Gamma_H(v)$

denotes the set of neighbors *in* H, i.e., $\Gamma_H(v) = \Gamma(v) \cap V_H$. For a set of vertices $S \subseteq V$, $\Gamma(S)$ denotes the set of all neighbors of vertices in S.

Recall from before, that the *log-density* of a graph G is defined to be

$$\rho_G := \frac{\log(|E_G|/|V_G|)}{\log |V_G|}.$$

Finally, for any number $x \in \Re$, will use the notation $\mathrm{fr}(x) = x - \lfloor x \rfloor$.

In many places, we will ignore leading constant factors (for example, we may find a subgraph of size $2k$ instead of k). It will be clear that these do not seriously affect the approximation factor.

3.2 Average case versions

The average case versions of the DkS problem will be similar in spirit to the planted clique problem discussed in Section 2.2.1. We first define the simplest variant, which we will call the *Random in Random* problem, and then define a more sophisticated version, which will be useful in our algorithm for DkS in arbitrary graphs.

3.2.1 Random planting in a random graph

We pose this as a question of distinguishing between two distributions over instances. In the first, the graph is random, while in the second, there is a "planted" dense subgraph):

$\mathcal{D}_1$: Graph G is picked from $G(n,p)$, with $p = n^{\theta-1}$, $0 < \theta < 1$.

$\mathcal{D}_2$: G is picked from $G(n, n^{\theta-1})$ as before. A set S of k vertices is chosen arbitrarily, and the subgraph on S is replaced with all edges within S are removed, and instead one puts a random graph H from $G(k, k^{\theta'-1})$ on S.[1]

[1]We also allow removing edges from S to $V \setminus S$, so that tests based on simply looking at the degrees do not work.

Note that in $\mathcal{D}_2$, the graph we pick first (G) has a log-density θ, while the one we plant (H) has a log-density θ'. To see we have planted something non-trivial, observe that for $G \sim \mathcal{D}_1$, a k-subgraph would have expected average degree $kp = kn^{\theta-1}$. Further, it can be shown that *any* k-subgraph in G will have an average degree at most $\max\{kn^{\theta-1}, 1\} \times O(\log n)$, w.h.p. Thus we will pick θ' so as to satisfy $k^{\theta'} \geq kn^{\theta-1}$.

One case in which this inequality is satisfied is that of $\theta' > \theta$ (we will think of both of these as constants). In this case we will give a simple algorithm to distinguish between the two distributions. Thus we can *detect* if the planted subgraph has a higher log-density than the *host* graph. Our approach for the distinguishing problem will be to look for constant size subgraphs H' which act as 'witnesses'. If $G \sim \mathcal{D}_1$, we want that w.h.p. G does not have a subgraph isomorphic to H', while if $G \sim \mathcal{D}_2$, w.h.p. G should have such a subgraph. It turns out that whenever $\theta' > \theta$, such H' can be obtained, and thus we can solve the distinguishing problem.

Standard results in the theory of random graphs (c.f. [70] or the textbook of Bollobás [23]) shows that if a graph has log-density greater than r/s (for fixed integers $0 < r < s$) then it is expected to have constant size subgraphs in which the ratio of edges to vertices is $s/(s-r)$, and if the log-density is smaller than r/s, such subgraphs are not likely to exist (i.e., the occurrence of such subgraphs has a *threshold behavior*). Hence such subgraphs can serve as witnesses when $\theta < r/s < \theta'$.

Observe that in the approach outlined above, r/s is rational, and the size of the witnesses increases as r and s increase. In general, if θ and θ' are constants, the size of r, s is roughly $O(\frac{1}{\theta'-\theta})$. Thus the algorithm is polynomial when the log-densities are a constant apart. This is roughly the intuition as to why we obtain an $n^{1/4+\varepsilon}$ approximation in roughly $n^{1/\varepsilon}$ time, and to why the statement of Theorem 3.5 involves a rational number r/s, with the running time depending on the value of r.

We can also consider the "approximation factor" implied by the above distinguishing problem. I.e., let us consider the ratio of the densities of the densest k-subgraphs in the two distributions (call this the 'distinguishing ratio'). From the discussions above, it would be $\min_{\theta'}(k^{\theta'}/\max\{kn^{\theta-1}, 1\})$, where θ' ranges over all values for which we can distinguish (for the corresponding values of k, θ). Since this includes all $\theta' > \theta$, it follows from a straightforward calculation that the distinguishing ratio is never more than

$$k^{\theta}/\max\{kn^{\theta-1}, 1\} \leq n^{\theta(1-\theta)} \leq n^{1/4}.$$

3.2.2 The **Dense vs. Random** question

The random planted model above, though interesting, does not seem to say much about the general DkS problem. We consider an 'intermediate' problem, which we call the Dense vs. Random question. The aim is to distinguish between $\mathcal{D}_1$ exactly as above, and $\mathcal{D}_2$ similar to the above, except that the planted graph H is an *arbitrary* graph of log-density θ' instead of a random graph. Now, we can see that simply looking for the occurrence of subgraphs need not work, because the planted graph could be very dense and yet not have the subgraph we are looking for. As an example, a K_4 (complete graph on 4 vertices) starts "appearing" in $G(n, p)$ at log-density threshold 1/3, while there could be graphs of degree roughly $n^{1/2}$ without any K_4's.

To overcome this problem, we will use a different idea: instead of looking for the *presence* of a certain structure, we will carefully *count* the number of a certain type of structures. Let us illustrate with an example. Let us fix $\theta = 1/2 - \varepsilon$, for some constant ε, and let $\theta' = 1/2 + \varepsilon$ (i.e., the planted graph is arbitrary, and has average degree $k^{1/2+\varepsilon}$). The question we ask is the following: consider a pair of vertices u, v. How many common neighbors do they have? For a graph from $\mathcal{D}_1$, the expected number of common neighbors is $np^2 \ll 1$, and thus we can conclude by a standard Chernoff bound that for *any* pair u, v, the number of common neighbors is at most $O(\log n)$ w.h.p. Now what is such a count for a graph from $\mathcal{D}_2$? Let us focus on the

k-subgraph H. Note that we can do a double counting as follows:

$$\sum_{u,v \in V_H} |\Gamma_H(u) \cap \Gamma_H(v)| = \sum_{u \in V_H} \binom{|\Gamma_H(u)|}{2} \geq k\binom{d}{2}, \tag{3.1}$$

where d is the average degree of H, which is chosen to be $k^{1/2+\varepsilon}$. The last inequality is due to convexity of the function $\binom{x}{2}$. Thus *there exists* a pair $u, v \in V_H$ such that $|\Gamma_H(u) \cap \Gamma_H(v)| \geq \frac{1}{k^2} \cdot k\binom{d}{2} \geq k^{\varepsilon}$. Now if we knew that $k^{\varepsilon} \gg \log n$, we can use this "count" as a test to distinguish! More precisely, we will consider the quantity $\max_{u,v \in G} |\Gamma(u) \cap \Gamma(v)|$, and check if it is $\geq k^{\varepsilon}$. In our setting, we think of $k = (\log n)^{\omega(1)}$, and ε as a constant, and thus we will always have $k^{\varepsilon} \gg \text{polylog}(n)$.

General Idea. In general for a rational number r/s, we will consider special constant-size trees, which we call *templates.* In a *template witness* based on a tree T, we fix a small set of vertices U in G, and count the number of trees isomorphic to T whose set of leaves is exactly U. The templates are chosen such that a random graph with log-density $\leq r/s$ will have a count at most poly-logarithmic for *every* choice of U, while we will show by a counting argument that in any graph on k vertices with log-density $\geq r/s + \varepsilon$, there exists a set of vertices U which coincide with the leaves of at least k^{ε} copies of T.

As another example, when the log-density $r/s = 1/3$, the template T we consider is a length-3 path (which is a tree with two leaves, namely the end points). For any 2-tuple of vertices U, we count the number of copies of T with U as the set of leaves, i.e., the number of length-3 paths between the end points. Here we can show that if $G \sim \mathcal{D}_1$, with $\theta \leq 1/3$, *every* pair of vertices has at most $O(\log^2 n)$ paths of length 3, while if $G \sim \mathcal{D}_2$, with $\theta' = 1/3 + \varepsilon$, there exists some pair with at least $k^{2\varepsilon}$ paths. Since $k = (\log n)^{\omega(1)}$, we have a distinguishing algorithm.

Let us now consider a general log-density threshold r/s (for some relatively prime integers $s > r > 0$). The tree T we will associate with the corresponding template

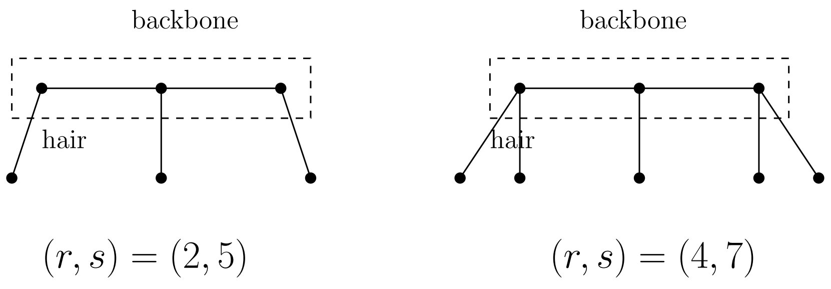

Figure 3.1: Example of caterpillars for certain r, s.

witness will be a caterpillar – a single path called the *backbone* from which other paths, called *hairs*, emerge. In our case, the hairs will all be of length 1. More formally,

Definition 3.1. *An* (r, s)-caterpillar *is a tree constructed inductively as follows: Begin with a single vertex as the leftmost node in the backbone. For* s *steps, do the following: at step* i*, if the interval* $[(i-1)r/s, ir/s]$ *contains an integer, add a hair of length* 1 *to the rightmost vertex in the backbone; otherwise, add an edge to the backbone (increasing its length by* 1*).*

See the figure for examples of caterpillars for a few values of r, s. The inductive definition above is also useful in deriving the bounds we require. Some basic properties of an (r, s) caterpillar are as follows:

1. It is a tree with $s + 1$ vertices (i.e., s edges).

2. It has $r + 1$ leaves, and $s - r$ "internal" vertices.

3. The internal vertices form the "backbone" of the caterpillar, and the leaves are the "hairs".

We will refer to the leaves as $v_0, v_1, \ldots, v_r$ (numbered left to right). Now the distinguishing algorithm, as alluded to earlier, is the following:

Thus we need to prove the *soundness and completeness* of the procedure above. These will be captured in the following lemmas. In what follows, we will write $\delta :=$

```
procedure Distinguish$(G, k, r, s)$
 // graph $G = (V, E)$, size parameter $k$, parameters $r, s$
begin
1   For every $(r+1)$-tuple of (distinct) leaves $U$, count number of
    $(r, s)$-caterpillars with $U$ as the leaves.
2   If for some $U$ the count is $> k^{\varepsilon}$, return YES.
3   Else return NO.
```

r/s. Also, we say that a caterpillar is *supported on* a leaf tuple U if it has U as its set of leaves.

Lemma 3.2. *Let $G \sim G(n,p)$ with $p \leq n^{\delta}/n$. Then with high probability, we have that for* any *$(r+1)$-tuple of leaves U, the number of (r,s) caterpillars supported on U is at most $O(\log n)^{(r-s)}$.*

Lemma 3.3. *Let H be a graph on k vertices with log-density at least $\delta + \varepsilon$, i.e., average degree $\geq k^{\delta+\varepsilon}$. Then there exists an $(r+1)$-tuple of leaves U with at least k^{ε} caterpillars supported on it.*

From the lemmas above, it follows that our algorithm can be used to solve the Dense vs. Random problem when $k = (\log n)^{\omega(1)}$. Let us now give an outline of the proofs of these lemmas. The point here is to see how to translate these ideas into the general algorithm, so we will skip some of the straightforward details in the proofs. Lemma 3.3 is proved using a simple counting argument which we will see first.

Proof of Lemma 3.3. Write $d = k^{\delta+\varepsilon}$, the average degree of H. For a moment, suppose that the minimum degree is at least $d/4$.[2] Now let us count the total number of (r,s) caterpillars in the graph. Let us view the caterpillar as tree with the first backbone vertex as the root. There are k choices in the graph for the root, and for every 'depth-one' neighbor, there are at least $d/4$ choices (because of the minimum degree assumption), so also for depth two, and so on.

[2]We can ensure this by successively removing vertices in H of degree at most $d/4$. In this process, we are left with at least half the edges, and a number of vertices which is at least $d \geq k^{\Omega(1)}$. The log-density is still $\geq \delta + \varepsilon$, thus we can work with this graph.

The argument above is correct up to minor technicalities: to avoid potentially picking the same vertex, we should never pick a neighbor already in the tree. This leads to the choices at each step being $d/4 - s$, which is still roughly $d/4$. Second, each caterpillar may be counted many times because of permutations in picking the leaves. However a judicious upper bound on the multiplicity is $s!$, which is only a constant.

Thus the total number of caterpillars is $\Omega(kd^s)$. Now we can do a double-counting as in Eq. (3.1): Each caterpillar is supported on some $(r+1)$-tuple of leaves, and thus we have $\sum_{(r+1) \text{ tuples } U} \text{count}(U) \geq \Omega(kd^s) \geq k^{r+1}k^{\varepsilon s}$. Thus there exists a U such that count(U) is at least $k^{\varepsilon s}$. $\square$

Let us now prove Lemma 3.2. The idea is to prove that for a given fixing of the leaves, the expected *number of candidates* for each backbone vertex in the caterpillar is at most a constant. We can then use Chernoff bounds to conclude that the number is at most $O(\log n)$ w.h.p. for every backbone vertex and every fixing of the leaves. Thus for every set of leaves, the number of caterpillars supported on them is at most $O(\log n)^{s-r}$ w.h.p. (since there are $(s-r)$ backbone vertices).

We begin by bounding the number of candidates for the rightmost backbone vertex in a *prefix* of the (r, s) caterpillar (as per the above inductive construction). For each $t = 1, \ldots, r$, let us write $S_{v_0,\ldots,v_{\lfloor tr/s \rfloor}}(t)$ for the set of such candidates at step t (given the appropriate prefix of leaves). Further, we will ask that the candidate vertices for these backbone vertices come from disjoint sets $V_0, V_1, \ldots, V_{\lfloor tr/s \rfloor}$. This will ensure that the events that $u \in S(t-1)$ and $(u, v) \in E$ are independent. This does not affect our counts in a serious way, because we are partitioning into a constant number of sets, the counts we are interested in are preserved up to a constant. More precisely, if we randomly color the vertices of a graph G with C colors, and G has M copies of a C-vertex template. Then w.h.p, there exist $\frac{M}{C^C}$ 'colorful' copies of the template (a colorful copy is one in which each vertex of the template has a different color).

The following claim implies upper bounds the cardinality of these sets (with high probability). (Recall the notation fr$x = x - \lfloor x \rfloor$.)

Claim 3.4. *In $G(n,p)$, for $p \leq n^{r/s-1}$, for every $t = 1, \dots, s$ and for any fixed sequence of vertices $U_i = v_0, \dots, v_{\lfloor tr/s \rfloor}$, for every vertex $v \in V \setminus U_i$ we have*

$$Pr[v \in S_{v_0,\dots,v_{\lfloor tr/s \rfloor}}(t)] \leq n^{\mathrm{fr}(tr/s)-1}(1+o(1)).$$

Intuitively, the claim follows from two simple observations: (a) For any set of vertices $S \subseteq V$ in $G(n,p)$, w.h.p. the neighborhood of S has cardinality at most $pn|S|$ (since the degree of every vertex is tightly concentrated around pn), and (b) for every vertex set S, the expected cardinality of its intersection with the neighborhood of any vertex v is at most $\mathbb{E}[|S \cap \Gamma(v)|] \leq p|S|$. Applying these bounds inductively to the construction of the sets $S(t)$ when $p = n^{r/s-1}$ then implies $|S(t)| \leq n^{\mathrm{fr}(tr/s)}$ for every t.

Proof. We prove the claim by induction. For $i = 1$, it follows by definition of $G(n,p)$: $\Pr[v \in S_{v_0}(1)] = p \leq n^{r/s-1}$. For $t > 1$, assume the claim holds for $t-1$. If the interval $[(t-1)r/s, tr/s]$ contains an integer (for $1 < t \leq s$ it must be $\lceil (t-1)r/s \rceil$), then $S(t) = S(t-1) \cap \Gamma(v_{\lceil (t-1)r/s \rceil})$. Thus, by definition of $G(n,p)$ and the inductive hypothesis,

$$\begin{aligned}\Pr[v \in S_{v_0,\dots,v_{\lfloor tr/s \rfloor}}(t)] &= p \cdot \Pr[v \in S_{v_0,\dots,v_{\lfloor (t-1)r/s \rfloor}}(t-1)] \\ &\leq n^{r/s-1} n^{\mathrm{fr}((t-1)r/s)-1}(1+o(1)) \\ &= O(n^{\mathrm{fr}(tr/s)-1}(1+o(1))).\end{aligned}$$

Otherwise, if the interval $[(t-1)r/s, tr/s]$ does not contain an integer, then $S(t) = \Gamma(S(t-1))$. In this case, by the inductive hypothesis, the cardinality of the set $|S(t-1)|$ is tightly concentrated around $n^{\mathrm{fr}((t-1)r/s)}$ (using Chernoff-Hoeffding bounds). It can be checked that this gives

$$\begin{aligned}\Pr[v \in S_{v_0,\ldots,v_{\lfloor tr/s\rfloor}}(t)] &= \Pr[\exists u \in S_{v_0,\ldots,v_{\lfloor tr/s\rfloor}}(t-1) : (u,v) \in E \cap v \in V_{\lfloor tr/s\rfloor}] \\ &\leq O(pn^{\mathrm{fr}((t-1)r/s)}(1+o(1))) \\ &\leq O(n^{\mathrm{fr}(tr/s)-1}(1+o(1))).\end{aligned}$$

Note, that in the last step, we use the fact that u, v are from disjoint sets $V_{\lfloor tr/s\rfloor}$ and $V_{\lfloor tr/s\rfloor}$ respectively, and hence, the two event $u \in S(t-1)$ and $(u,v) \in E$ are independent. The various vertices in G all have the same probability of membership in $S(t-1)$. Further, even for these independent variables, tight concentration is only achieved when the expected size of the set is $n^{O(1)}$. However, this is guaranteed by the inductive hypothesis, assuming r and s are relatively prime). □

Now by symmetry, the same bounds can be given when constructing the candidate sets in the opposite direction, from right to left (note the symmetry of the structure). Thus, in the leftmost vertex set is also $n^{-1}(1+o(1))$. Moreover, once all the leaves are fixed, every candidate for an internal vertex can be described, for some $t \in [1, s-1]$, as the rightmost backbone vertex in the tth prefix, as well as the leftmost backbone vertex in the $(s-t)$th prefix starting from the right. deviate much from their respective expectations, as guaranteed with high probability), By Claim 3.4, the probability of this event is at most

$$n^{\mathrm{fr}(tr/s)-1} n^{\mathrm{fr}((s-t)r/s)-1}(1+o(1)) = n^{-1}(1+o(1)).$$

Thus, since the (r,s)-caterpillar has $s-r$ internal vertices and $r+1$ leaves, it follows by standard probabilistic arguments that, for some universal constant $C > 0$, the probability that total number of caterpillars for any sequence of leaves exceeds $(\log n)^{s-r}$ is at most $(s-r)n^{r+1}n^{-C\log\log n}$, which is $o(1)$ for any constants r, s.

This completes the proof of Lemma 3.2.

3.3 Algorithm for general graphs

Let us now see how to use such ideas to obtain an approximation algorithm for DkS in arbitrary graphs. In general, simple counting does not work so we need more sophisticated averaging arguments. The algorithm will still involve the caterpillar graphs defined in Section 3.2.2, and the main theorem we will prove can be stated as follows (to be precise, we give a *family* of algorithms, parametrized by a rational number r/s).

Theorem 3.5. *Let s, r be relatively prime natural numbers with $r < s$, and let $k = n^{1-\frac{r}{s}}$. Let G be an undirected graph which contains a k-subgraph H of average degree d. Then there is an algorithm running in time $n^{O(r)}$ that finds a k-subgraph of average degree $\Omega(\max\{1, \frac{d}{k^{r/s}\log n}\})$.*

Thus if $k = n^{1-\frac{r}{s}}$, the theorem gives an approximation ratio (up to a logarithmic factor) of $n^{\frac{r}{s}\left(1-\frac{r}{s}\right)} \le n^{1/4}$. In general the k could be arbitrary, and thus we will pick an r, s s.t. k is *close* to $n^{(s-r)/s}$, and thus we obtain the following corollary.

Corollary 3.6. *There exists an algorithm with run time $n^{O(1/\varepsilon)}$, which gives an approximation ratio of $n^{1/4+\varepsilon}$ to the Densest k-subgraph problem.*

We remark that our results have been stated for the *unweighted* DkS problem. This is not a serious restriction, since we can bucket the edges into $O(\log n)$ levels according to the edge weights (which we assume are all positive), and output the densest of the k-subgraphs obtained by applying the algorithm to each of the graphs induced by the edges in a bucket. This incurs a loss of an additional $O(\log n)$ factor in the approximation.

The rest of the section will be about proving Theorem 3.5. We will start with some simplifying assumptions, and justify why they can be made wlog. Details of these steps are often straightforward, and the reader could choose to focus on just the *statements* in the first read.

A couple of simple observations before we begin:

1. Note that we can always return a k-subgraph with density (average degree) $\Omega(1)$ (we could return a tree or a matching). Thus we will assume throughout the proof of Theorem 3.5 that $d \geq Ck^{r/s}$, for a parameter $C = \omega(1)$.

2. A greedy algorithm obtains an approximation ratio of $O(n/k)$. More precisely, we can start with the maximum density subgraph, and if it has a size $> k$, sample k vertices randomly (or in a greedy way based on the degree) to obtain a density which is at least a k/n factor of the max density subgraph. Thus it is an n/k-factor approximation.

3.3.1 Some simplifications and preprocessing

We will start with a couple of simple observations about the problem, and assumptions on the input we can make without loss of generality (see the lemmas for precise statements).

1. The input graph G (and hence H) is bipartite.

2. It suffices to find a subgraph of size *at most* k, rather than exactly k. In what follows, we use 'k-subgraph' more loosely to mean a subgraph on at most k vertices.

3. The minimum degree in H is $d/4$.

4. If we wish to recover a k-subgraph of average degree C, the maximum degree in G can be assumed to be $D \leq \frac{Cn}{k} = Cn^{r/s}$.

The first assumption will be useful because we will define sets of vertices S in a specific way and analyze the graph between S and $\Gamma(S)$. It will be cleaner to have

$S, \Gamma(S)$ to not intersect (and this would happen if G were bipartite, from the way S will be defined).

The other assumptions will also help simplify the presentation greatly. Let us now prove them in order.

Lemma 3.7. *Given an $f(n)$-approximation for DkS on n-vertex bipartite graphs, we can approximate DkS on arbitrary graphs within a $\Omega(f(2n))$-factor.*

Proof. Take two copies of the vertices of G, and connect copies of vertices (in two different sides) which are connected in G. Thus the densest $2k$-subgraph in the new bipartite graph has at least the same average degree as the densest k-subgraph in G. Now take the subgraph found by the bipartite DkS approximation on the new graph, and collapse the two sides (this cannot reduce the degree of any vertex in the subgraph). Note that the subgraph found may be a constant factor larger than k, in which case, as before, we can greedily prune vertices and lose a constant factor in the average degree. □

Thus we can think of doing a pre-processing step in which we make the graph bipartite before applying our algorithm.

Lemma 3.8. *Given an algorithm which, whenever G contains a k-subgraph of average degree $\Omega(d)$ returns a k'-subgraph of average degree $\Omega(d')$, for some (non specific) $k' < k$), we can also find a (exactly) k-subgraph with average degree $\Omega(d')$ for such G.*

Proof. Apply the algorithm repeatedly, each time removing from G the edges of the subgraph found. Continue until the union of all subgraphs found contains at least k vertices. The union of subgraphs each of average degree d' has average degree at least d'. Hence there is no loss in approximation ratio if we reach k vertices by taking unions of smaller graphs. Either we have not removed half the edges from the optimal solution (and then all the subgraphs found – and hence their union – have average degree $\Omega(d')$), or we have removed half the edges of the optimal solution, in which case our subgraph has at least $dk/2$ edges.

Note that this algorithm may slightly overshoot (giving a subgraph on up to $2k$ vertices), in which case we can greedily prune the lowest degree vertices to get back a k-subgraph with the same average degree (up to a constant factor). □

We can now view our algorithm as a sub-routine which keeps finding k'-subgraphs of *high enough* density, and we can patch them together as above.

Lemma 3.9. *Write $\theta = r/s$, and suppose $\theta < 5/6$. Now given a G with a k-subgraph of average degree Ck^{θ}, there exists a k'-subgraph with $k' = n^{1-\theta'} \le k$, and* minimum degree *at least $(C/10) \cdot (k')^{\theta'}$.*

Proof. Let H be the densest k-subgraph in G. Iteratively remove from H any vertex which has average degree less than (1/10)th of the *current* average degre of H. [Note that this is a slight modification of the 'standard' trick of removing vertices of degree smaller than some factor of the *original* average degree]. This procedure will terminate with the graph being non-empty because the average degree in the process is monotonically increasing. Now how many of the edges remain in total?

Consider a step in which the number of vertices remaining is N (we started with $N = k$). If $N \ge k/2$, the number of edges removed so far is at most $(k/2) \cdot d/10 \le kd/20$. Thus the number of edges has decreased by a factor at most $9/10$ (because the number of edges to start with is $kd/2$).

By repeating this argument, we can conclude that if the number of vertices decreases from k to $k/2^r$, the number of edges remains at least $(9/10)^r E$. Or in other words, if the number of vertices drops to k/n^{ρ}, the number of edges is at least $E/n^{\rho/3}$. Thus the average degree changes from d to $d \cdot n^{2\rho/3}$. Let us call this number d'.

We claim that if $k = n^\theta$ and $d = Cn^{\theta(1-\theta)}$, with $\theta < 5/6$, then for any $\rho \geq 0$, we have $d' \geq Cn^{(\theta-\rho)(1-\theta+\rho)}$. That is, we wish to show

$$\begin{aligned} \theta(1-\theta) + \frac{2\rho}{3} &\geq \theta(1-\theta) + \theta\rho - \rho(1-\theta+\rho) \\ \iff \frac{2\rho}{3} &\geq \rho(2\theta + \rho - 1) \end{aligned}$$

For $\rho > 0$, this is equivalent to $2\theta - \rho \leq 5/3$, which is true since $\theta < 5/6$. This gives the desired bound on the minimum degree. □

A quick note, we assumed in the lemma that $\theta < 5/6$. This is not critical, because we are shooting for an $n^{1/4}$ approximation to DkS, and if $\theta > 5/6$, the greedy n/k approximation gives a better ratio. Further, by changing the constant $1/10$ in our proof, we can make the constant $5/6$ as close to 1 as we wish.

Thus the lemma suggests that we should run the distinguishing algorithm for every $k' \leq k$ (there at most n choices; also the choice of k' will determine the caterpillar used in the algorithm). Hence in what follows, we will assume that we are working with the "right" k, i.e., there exists an H of size k with minimum degree $(C/4)k^{r/s}$. For convenience, we define a short notation for this.

Definition 3.10. *A subgraph H is said to be a (k, d)-subgraph if it is on precisely k vertices, and the* minimum degree *is $\geq d$.*

Lemma 3.11. *Let G be a graph, and let k, C be parameters. Suppose G has a (k, d)-subgraph with $d > 10C$. Then there exists a polynomial time algorithm which produces either:*

1. *a k-subgraph of density $\geq C/2$, or*

2. *a graph G' on at least $n - 3k/4$ vertices which contains a $(\Omega(k), \Omega(d))$-subgraph, and the additional property that the* maximum degree *in G' is $\leq Cn/k$.*

Thus this lemma should be seen as a pre-processing step. Suppose we are shooting to find a C-dense k-subgraph. Then we can assume the maximum degree in G to be at most Cn/k. A subtle point is that the statement involves (k, d)-subgraphs (which involves the minimum degree of the subgraphs – this is important).

Proof. Let U be the set of $k/2$ vertices of highest degree in G (breaking ties arbitrarily), and let $D = \min_{u \in U} \deg(u)$. Let U' be the set of $k/2$ vertices of G of highest degree into U. Let H' be the graph induced on $U \cup U'$.

Claim. If $D > Cn/k$, then H' is a C-dense subgraph.

This is easy to see, because U has at least $kD/2$ edges going into $V(G)$, and hence the *best* $k/2$ vertices account for at least $\frac{k}{2n} \cdot \frac{kD}{2}$ of these edges, which implies that the average degree is at least $\frac{kD}{4n} = \Omega(C)$. This proves the claim.

Next, we successively move vertices in $V(G) \setminus U$ which have a degree $\geq C$ into U, into U. If in this process we ended up moving at least $k/4$ vertices into U, we are done, because we obtained an $\Omega(C)$-dense subgraph.

Thus if this process did not succeed, we have the condition at the end, that no vertex in $G \setminus U$ has more than a degree C into U. Since we have removed at most $3k/4$ vertices so far, we still have $k/4$ vertices in H, which was the (k, d)-subgraph in G. Thus the subgraph formed by these is an $(\Omega(k), \Omega(d))$-subgraph in $G \setminus U$. This completes the proof. $\square$

Thus in what follows, unless otherwise stated, G will be a bipartite graph, $k = n^{1-r/s}$, and G has a (k, d)-subgraph for some $d > k^{r/s}$. The *core* of the algorithm is based on the following theorem, which will be the subject of the next section.

Theorem 3.12. *Suppose G is as above, with $d \geq C^2 k^{r/s}$. Then there exists an algorithm which runs in time $O(n^r)$ and finds a k-subgraph of density $\Omega(C)$.*

Note that this does not quite imply Theorem 3.5, because when C is $\gg \log^2 n$ (say a power of n), the approximation factor could be much worse than $\widetilde{O}(k^{r/s})$.

However the theorem above has an easy corollary and the two together easily imply Theorem 3.5.

Corollary 3.13. *Given G as above, with $d \geq \rho k^{r/s}$ and suppose $\rho > 10 \log^2 n$. Then there exists a randomized algorithm which finds a k-subgraph of density $\geq \rho / \log n$ w.h.p.*

Proof. The idea is to "sparsify" G by subsampling edges, and appeal to Theorem 3.12. More formally, suppose G' is a graph we obtain by sampling each edge from G with a probability $p := 10 \log^2 n/\rho$. Now, the following simple claims follow from Chernoff bounds. Let H be the (k, d) subgraph which is assumed to exist in G, and let H' be the subgraph in G' with the same vertex set.

Claim 1. H' is a $(k, \log^2 n)$-subgraph w.h.p.

To prove this, note that the degree of any vertex in H' can be written as a sum of independent Bernoulli random variables, each with probability p, and the expected value is $10 \log^2 n$. Thus the probability that it is smaller than $\log^2 n$ is at most $1/n^{\omega(1)}$, and hence we can take a union bound to conclude this statement for all the vertices of H'.

Claim 2. Let S be a set of k vertices with at least $2k \log n$ edges in G'. Then S has at least $\frac{k\rho}{40 \log n}$ edges in G.

To see this, consider the contrapositive. Suppose some set of vertices had $< \frac{k\rho}{40 \log n}$ edges in G. Then the expected number of edges in G' is at most $\frac{k}{4 \log n}$, and thus the probability that it is $> 2k \log n$ is at most $e^{-k \log n} < n^k$. Thus we can take a union bound over all $\binom{n}{k}$ choices of S, and this proves the claim.

Now given the two claims, the corollary follows, because we can first sparsify, and use Theorem 3.12 to obtain a $(2 \log n)$-dense subgraph, and Claim 2 then gives the desired result. □

Thus proving Theorem 3.12 will now give Theorem 3.5. This will be the subject of the remainder of the section.

3.3.2 An outline of the core algorithm

We will now prove Theorem 3.12. Recall that H is the densest k-subgraph (breaking ties arbitrarily), and it has minimum degree $d = C^2 k^{r/s}$. From Lemma 3.11, we may assume that G has a *maximum* degree at most $Cn/k = Cn^{r/s}$, and the aim is to recover a k-subgraph of average degree $\Omega(C)$.

Intuition. The algorithm is based on the intuition that a larger number of caterpillars are supported on leaf tuples in H than leaf tuples from G (as in the random graph case). More specifically, we *guess* the leaf tuple, one leaf at a time, and maintain a 'set of candidates' for the rightmost backbone vertex at each step. (Note that this is very similar to the proof of Lemma 3.2 in which we kept count of these sets).

The idea is that these *candidate sets* give a good estimate of the dense subgraph H. In particular, we will maintain a good lower bound on the intersection of the candidate sets and H (i.e., we prove that for *some choice* of guesses for the leaves, we will have a good lower bound). Then we perform a simple greedy search starting with these candidates, and prove that in one such search, we should have found a C-dense subgraph. (Else the bounds we obtain on the sizes of the candidate sets will lead to a contradiction).

We can now outline the algorithm. The iterative definition of the caterpillar structure is very helpful in describing it easily (Definition 3.1). We will denote by L_i the number of *hair steps* (i.e., steps in which a leaf/hair was added) up to iteration i, and by B_i the number of *backbone steps* (those in which a new vertex was added to the backbone). Since every step is one of the two kinds, $L_i + B_i = i$. At step i we will have a set of vertices, called S_i which is the set of candidates for the rightmost backbone vertex after fixing the leaves chosen so far.

We now state the algorithm, assuming we have a procedure DkS-Local, which is a simple greedy search starting with a set of vertices. (It is very similar in spirit to finding the maximum density subgraph, which we discussed in Section 2.3).

```
procedure DkS-Cat($G, k, r, s$)     // graph $G$, parameters $k, r, s$
begin
1   Start with $i = 0$ and $S_0 = V$.
2   for $i = 1, \ldots, s$ do
3       Consider iteration $i$ in the construction of an $(r, s)$-caterpillar (Def. 3.1).
4       for each choice of $u_1, \ldots, u_{L_i} \in V$ do
5           Compute the set of candidates $S_i$ for the rightmost internal vertex
            after fixing $u$'s as leaves.
6           If DkS-Local $(S, k)$ returns a $C$-dense subgraph, return it and exit.
```

Our analysis proceeds inductively, and loosely speaking, it will argue that:

> If no C-dense subgraph was found until (and including) step i in the algorithm, then there exists a set of L_i leaves for which S_i (after fixing this set as the leaves) satisfies a certain condition.

This condition will be s.t. if $i = s$, we obtain a contradiction for our chosen values for the parameter d. This implies that we must have found a C-dense subgraph in some step of the algorithm.

We will formally get to the condition on S_i in Section 3.3.4, but roughly, it says:

$$\frac{|S_i \cap H|}{|S_i|} > \frac{k^{\frac{ir}{s}+1-L_i}}{n^{\frac{ir}{s}+1-L_i}}.$$

Note that for $i = s$, we have $L_i = r+1$, thus the ratio is 1, and this is a contradiction because we cannot have $|S_i \cap H| > |S_i|$!

The factors in the numerator and denominator now ought to ring a bell: $n^{\frac{ir}{s}+1-L_i}$ is precisely the number of candidates for S_i we expect to see in an n-vertex *random graph* with log-density r/s (as in Section 3.2.2).

Thus in a way, we ensure that we can find a set of leaves so that the set of candidates behaves *at most as badly* as it does in the random planted version of the problem. Though it is only an artefact of the analysis, this suggests that random instances are perhaps the *most difficult*.

3.3.3 A crucial subroutine

We now describe the procedure DkS-Local we used crucially in the algorithm. As outlined earlier, the procedure has the property that if it ends up *not* finding a dense enough subgraph, then it guarantees that certain bounds hold for the input subset. This "win-win" style argument is key to our analysis.

For simplicity of analysis, we will view the procedure DkS-Local (S, k) as consisting of two algorithms (both of which return k-subgraphs), one for the case when the underlying step (in the main algorithm) is a hair step and the other for when it is a backbone step. So we can view DkS-Local as a procedure which runs both these and outputs the denser subgraph. Let us describe the two procedures (called Dks-Local-Hair and Dks-Local-BB).[3]

```
procedure Dks-Local-Hair($S, k$)      // set $S \subseteq V$, size parameter $k$
begin
1    Consider the bipartite subgraph induced on $S, \Gamma(S)$.
2    Pick the $k$ vertices in $\Gamma(S)$ with the highest degree into $S$, call this $B_s$.
3    Pick the $k$ vertices in $S$ with the highest degree into $B_s$, call it $A_s$.
4    Return the subgraph induced on $A_s, B_s$.
```

[3]As in our paper [17], it is possible to perform just one procedure which captures both these, but it is somewhat more tricky to analyze.

```
procedure Dks-Local-BB$(S, k)$    // set $S \subseteq V$, size parameter $k$
begin
1    Consider the bipartite subgraph induced on $S, \Gamma(S)$.
2    Let $H'$ be the maximum density subgraph in this bipartite graph. Let its
     vertex sets be $A_0 \subseteq S$ and $B_0 \subseteq \Gamma(S)$.
3    Let $A_1$ be the $\min\{k, |A_0|\}$ vertices in $A_0$ of the largest degree to $B_0$. Now
     let $B_1$ be the $|A_1|$ vertices in $B_0$ with largest degree to $A_1$.
4    Return the graph induced on $A_1, B_1$.
```

The following lemmas will analyze the two procedures above. In these, by DkS-Local, we mean the corresponding (hair or back-bone) version.

Lemma 3.14 (Backbone Lemma A)**.** *Let $S \subseteq V$, and suppose $|S| \le k$. Then either DkS-Local (S, k) finds a C-dense subgraph, or we have*

$$|\Gamma(S) \cap H| \ge \frac{d}{C} \cdot |S \cap H|.$$

Proof. Suppose $|\Gamma(S) \cap H| < \frac{d}{C} \cdot |S \cap H|$. Then the graph induced on $(S \cap H, \Gamma(S) \cap H)$ will have density at least C. This is because the number of edges is at least $d|S \cap H|$ (by the assumption on minimum degree in H), and the number of vertices is $\le |S \cap H|(1 + \frac{d}{C})$.

Now since $|S| \le k$, $A_1 = A_0 \le k$, and the output subgraph will have a density at least C. □

Lemma 3.15 (Backbone Lemma B)**.** *Let $S \subseteq V$, and $|S| > k$. Then DkS-Local (S, k) finds a k-subgraph of density at least $\frac{d|S \cap H|}{|S|}$.*

Proof. In this case, the second step of Algorithm 4 will find a subgraph with density at least $\frac{d|S \cap H|}{k}$, because the graph with $S \cap H$ on one side and $\Gamma(S) \cap H$ on the other will have at least $d|S \cap H|$ edges and at most k vertices.

Now if $|A_0 \leq k$, we are done, because we will output a subgraph of density at least $\frac{d|S\cap H|}{k}$, which is larger than the density we want. Else, our procedure will result in a density which is at most a $k/|A_0|$ smaller. Thus the density output is at least

$$\frac{k}{|A_0|} \cdot \frac{d|S \cap H|}{k} \geq \frac{d|S \cap H|}{|S|}.$$

This proves the lemma. ☐

Lemma 3.16 (Hair Lemma). *Let $S \subseteq V$. Then either* DkS-Local (S, k) *finds a C-dense subgraph, or there exists a $v \in H$ and a factor $f \geq 1/2$ s.t.*

$$|\Gamma(v) \cap S| \leq f \cdot \frac{C(|S| + k)}{k}$$
$$|\Gamma(v) \cap S \cap H| \geq f \cdot \frac{d|S \cap H|}{k}$$

Proof. Consider the iteration $k' = k$ in DkS-Local. The vertices picked by the algorithm will have as much total degree into S as the vertices of H have into S (because we pick the k *best* vertices). Thus if we did not find a C-dense subgraph, we must have

$$\sum_{v \in H} |\Gamma(v) \cap S| \leq C(k + |S|).$$

But now, since the minimum degree in H is d, we have that the number of edges between S and H is at least $|S \cap H|d$. (Note that the graph being bipartite makes the analysis here very clean). Now, we can rewrite this observation as

$$\sum_{v \in H} |\Gamma(v) \cap S \cap H| \geq d|S \cap H|.$$

These inequalities allow us to conclude that there exists a v with a good *ratio* of $|\Gamma(v) \cap S \cap H|/|\Gamma(v) \cap H|$. However we want a slightly stronger property. We thus prove the following simple claim.

Claim. Let $a_1, \ldots, a_k \geq 0$ and $b_1, \ldots, b_k \geq 0$ be real numbers which satisfy $\sum_i a_i \leq A$, and $\sum_i b_i \geq B$. Then there exists an index j s.t. $b_j \geq \frac{B}{2k}$, and $\frac{a_j}{b_j} \leq \frac{2A}{B}$.

Proof. (of Claim). By our assumption, we have

$$\sum_i \max\left\{b_i - \frac{B}{2k}, 0\right\} \geq \frac{B}{2}.$$

Now suppose we remove indices with summand being zero (the a_i are all ≥ 0, so the sum of a_i still remains $\leq A$). Thus there exists an index j s.t. $b_j > \frac{B}{2k}$, and $\frac{a_j}{b_j - (B/2k)} \leq \frac{2A}{B}$. This j clearly satisfies the condition we want. This proves the claim. $\square$

Now we can apply the claim with a_i being the numbers $|\Gamma(v) \cap S|$, and b_i being $|\Gamma(v) \cap S \cap H|$ for different v (there are k of them). This gives the desired claim. $\square$

3.3.4 Analysis of the core algorithm

We recall the opening paragraph of Section 3.3.2. Let us write $D = Cn/k = Cn^{r/s}$. Thus D is an upper bound on the degree in G. Our main theorem about Algorithm 2 is the following.

Theorem 3.17. *Suppose Algorithm 2 did not output a C-dense subgraph up to (and including) step i. Then there exist a set of leaves L_i, and a factor $f_i > 1$ for which the set of candidates for the last back-bone vertex S_i satisfies the two conditions*

$$|S_i| \leq f_i \cdot n^{\frac{ir}{s}+1-L_i} \tag{3.2}$$

$$|S_i \cap H| > f_i \cdot k^{\frac{ir}{s}+1-L_i} \tag{3.3}$$

Before we go to the proof, let us observe a couple of other simple properties of (r, s)-caterpillars which we will use in the proof.

Observation 3.18. *The number of hair steps up to (and including) step i is precisely $\lceil \frac{ir}{s} \rceil$ for $1 \le i < s$. Thus we have that for $0 \le i < s$,*

1. *if step $(i+1)$ is a hair step, then $L_i \le \frac{(i+1)r}{s}$.*
2. *if step $(i+1)$ is a backbone step, then $L_i \ge \frac{(i+1)r}{s}$.*

Proof. The fact that $L_i = \lceil \frac{ir}{s} \rceil$ follows easily from the definition of the caterpillar – the number of leaves increases iff there is an integer in the range $[(i-1)r/s, ir/s]$, and the number is 1 at $i = 1$. From this, it follows that if the $(i+1)$st step is a leaf step, then $\lceil \frac{ir}{s} \rceil \le (i+1)r/s$ (because there is an integer in the interval $[ir/s, (i+1)r/s]$). Part (b) follows similarly. □

We are now ready to prove the theorem. It will be by induction, and after all the earlier lemmas, it is now quite simple to prove.

Proof. For $i = 0$, the claim is trivially true. Now suppose the claim is true inductively for i and let us consider the $i+1$th step. Let L_i be a set of leaves for which the inequalities (3.2) and (3.3) hold for the corresponding S_i. We consider the two natural cases.

Case 1. $(i+1)$th step is a hair step.

Now by Lemma 3.16, we obtain that there exists a $v \in H$ (which will be the 'right' choice of u_{L_i+1}) such that for some $f \ge 1/2$, we have

$$|\Gamma(v) \cap S_i| \le f \cdot \frac{C(|S_i| + k)}{k}$$
$$|\Gamma(v) \cap S_i \cap H| \ge f \cdot \frac{d|S_i \cap H|}{k}$$

Now the point to observe is that the recursive upper bound on $|S_i|$ is $f_i n^{\frac{ir}{s}+1-L_i}$ with $f_i \ge 1$, and in our setting, this number is $\ge k$. This is because $k = n^{1-r/s}$, and we have $\frac{ir}{s} + 1 - L_i \geq 1 - \frac{r}{s}$, because of Observation 3.18. Thus we can conclude

$$|S_{i+1}| \leq 2Cff_i \cdot n^{\frac{(i+1)r}{s}+1-L_{i+1}}.$$

Now for the lower bound. By the inductive assumption and Lemma 3.16, we have

$$|S_{i+1} \cap H| \geq C^2 ff_i \cdot k^{\frac{(i+1)r}{s}+1-L_{i+1}}.$$

Now noticing that $f \geq 1/2$ and $C > 2$, we have $C^2 f \geq 1$ and $2Cf \leq C^2 f$, thus we can set $f_{i+1} = f_i \cdot C^2 f$, and maintain the recursive bound.

Case 2. $(i+1)$th step is a backbone step.

Now we have to be more careful. Suppose $|S_i| \leq k$, then we are in good shape. Because the maximum degree is D, we have

$$|S_{i+1} \leq D|S| \leq Cn^{r/s} \cdot |S_i| \leq Cf_i \cdot n^{\frac{(i+1)r}{s}+1-L_{i+1}},$$

and by Lemma 3.14 we have

$$|S_{i+1} \cap H| \geq \frac{d}{C} \cdot |S_i \cap H| \geq Ck^{r/s} \cdot k^{\frac{(i+1)r}{s}+1-L_{i+1}}.$$

(The last step uses the lower bound on d). This gives the required inductive bound.

Now suppose $|S_i| > k$. In this case, we claim that DkS-Local always finds a C-dense subgraph. We can use Lemma 3.15 to conclude that we obtain a subgraph of density at least $\frac{|S_i \cap H|}{|S_i|} \cdot d$. Since $d > C^2 k^{r/s}$, and from the inductive bounds on S_i, it suffices to show that

$$\left(\frac{k}{n}\right)^{\frac{ir}{s}+1-L_i} \cdot k^{r/s} \geq 1.$$

Since $n/k = n^{r/s}$, we can cancel r/s in the exponent, and reduce to showing

$$1 - \frac{r}{s} \geq \frac{ir}{s} + 1 - L_i \quad \text{i.e., } L_i \geq \frac{(i+1)r}{s}.$$

This also follows from Observation 3.18, and this completes the proof by induction.

□

This completes the description and analysis of our caterpillar based algorithm.

3.4 LP hierarchies: a different view

Our algorithm as stated is purely combinatorial, and is based on going through the caterpillar structures, and performing greedy *local* algorithms at each step. It turns out that the entire procedure can be seen as the rounding of a *lifted* version of a natural Linear Program (LP) for the Densest k-subgraph problem.

We will assume basic familiarity with lift and project methods in this section. See the survey of Chlamtac and Tulsiani [30] for an exposition.

Suppose as before, G is a graph which contains a (k, d)-subgraph, as before. Now consider the following LP (as a feasibility question)

$$\sum_{i \in V} y_i \leq k, \text{ and}$$

$$\exists \{y_{ij} \mid i, j \in V\} \text{ s.t.}$$

$$\forall i \in V \quad \sum_{j \in \Gamma(i)} y_{ij} \geq d y_i$$

$$\forall i, j \in V \; y_{ij} = y_{ji}$$

$$\forall i, j \in V \; 0 \leq y_{ij} \leq y_i \leq 1$$

As is, the LP has a gap of $\sqrt{n}$, however after $1/\varepsilon$ rounds of the Sherali-Adams hierarchy, we can bound the gap by a much better factor. In fact we can use arguments very similar to those from our algorithm to obtain a caterpillar-based rounding algorithm for the LP above. The analogy to the combinatorial algorithm is as follows

1. The fixing of the leaves is seen as *conditioning* on picking a vertex (as we have seen, our algorithm intended to pick the leaves from the optimum set H).
2. We can use the *LP value* of a set $LP(S) := \sum_{i \in S} y_i$ as a proxy for $|S \cap H|$ in our bounds.
3. The algorithm will go through the caterpillar structure. At each hair step, it will *condition* on some vertex in the graph, and maintains a lower bound on $LP(S)/|S|$ (much like in the combinatorial algorithm).
4. The procedure DkS-Local has to be performed here as well. In this case, it can be viewed as a rounding algorithm, and if the procedure fails, we obtain bounds on $LP(S)/|S|$. The win-win argument in this case works as a "round or bound" method.

The LP based approach is detailed in our paper [17], and we do not include it here.

3.5 A trade-off between run time and the approximation factor

We have seen that the *log-density* is a natural *barrier* towards counting based algorithms. So a natural question which arises is, can we obtain a better approximation ratio at the expense of "mildly exponential" running time?

It turns out that the answer is yes, and in fact we can extend our algorithms in a fairly natural way in order to achieve this. We obtain a modification of our caterpillar-based algorithm, which yields an approximation ratio of $O(n^{1/4-\varepsilon})$ approximation which runs in time $2^{n^{O(\varepsilon)}}$. The main idea is that for each leaf, rather than picking an individual vertex, we will pick a cluster of roughly $O(n^{\varepsilon})$ vertices (which is responsible for the increased running time). The cluster will be used similarly to a single leaf vertex: rather than intersecting the current set with the neighborhood of a single vertex, we will intersect the current set with the neighborhood of the cluster (i.e., the union of all neighborhoods of vertices in the cluster).

Sketch of the main idea. Let us now see why we could expect such a procedure to work (and why we *need* to look at sets of size roughly n^{ε}). Let us go back to the random planted model, and suppose we are given a random graph G with log-density ρ. Let r/s denote a rational number roughly equal to $\rho + \delta$ (for some small constant δ). Suppose we call an $(r+1)$ tuple of leaves 'special' if there is an (r, s) caterpillar 'supported' on it, in the sense of Section 3.2.2. Since $r/s > \rho$, most $(r+1)$-tuples of vertices are not special.

The crucial observation is the following: say we pick a set S of vertices, and ask how many $(r+1)$ tuples from S are special. This number turns out to be 'large' (appropriately defined later) roughly iff $|S| > n^{\delta}$.

Now suppose we had planted a random graph H on k vertices and log-density $\rho - \varepsilon$ in G. Further suppose δ is such that $k^{\varepsilon+\delta} \ll n^{\delta}$ (since k is much smaller than n, we can always choose this so, by setting δ to be a constant multiple of ε). By the above claim, sets in H of size $k^{\varepsilon+\delta}$ would have a 'large' number of special tuples (since $r/s = (\text{log-density of } H) + \varepsilon + \delta$). But this number, by choice is much smaller than n^{δ}, thus if there was no H planted, sets of size $k^{\varepsilon+\delta}$ would not have many special tuples!

This gives a distinguishing algorithm which runs in time roughly $\binom{n}{k^{\varepsilon+\delta}}$. Let us now develop the algorithm in detail, for arbitrary graphs. The main result of this section is as follows.

Theorem 3.19. *For every $\varepsilon > 0$, there is a randomized $O(2^{n^{O(\varepsilon)}})$-time algorithm which for every graph G with high probability finds an $\widetilde{O}(n^{1/4-\varepsilon})$ approximation to DkS.*

We will not try to optimize the constant in the $O(\varepsilon)$ term, though it is somewhat interesting since the term appears in the exponent. The theorem follows as a corollary to the following:

Theorem 3.20. *For every $0 < \varepsilon < 1/2$, and every $0 < \beta < 1$, there is a randomized algorithm which for every instance of DkS with $k = n^{1-\beta}$, finds a k-subgraph whose average degree is within $O(n^{\beta(1-\beta)(1-\varepsilon)})$ of the optimum (w.h.p.). Moreover, this algorithm runs in time at most $2^{n^{O(\varepsilon)}}$.*

Let us see that Theorem 3.19 follows easily assuming this.

Proof of Theorem 3.19. For $\varepsilon > 1/10$, we can check every subgraph in time $2^n < 2^{n^{10\varepsilon}}$. Thus, we may assume $\varepsilon \leq 1/10$. Second, we can assume that $\beta \in (\frac{1}{2}-\frac{1}{5}, \frac{1}{2}+\frac{1}{5})$, because otherwise the earlier algorithms give a factor strictly better than $1/4 - 1/10$. In this case $\beta(1-\beta) > 1/5$, thus the algorithm from Theorem 3.20 gives an approximation factor of $n^{\beta(1-\beta)(1-\varepsilon)} < n^{1/4-\varepsilon/5}$.

Writing $\varepsilon = 5\varepsilon'$ and using the above, we obtain an $n^{1/4-\varepsilon'}$ approximation for $\varepsilon' > 1/10$. $\square$

As before, we will perform some additional preprocessing steps which will simplify the algorithm and its analysis. Suppose $k = n^{1-\beta}$, and let $d = k^{\beta(1-\varepsilon)}$ denote the average degree of the densest k-subgraph. Then it suffices to consider the case in which the instance has the following additional properties.

1. G is bipartite, and the optimum subgraph H has k vertices on each side of the bipartition.[4]

2. The *minimum* degree of vertices in H is d.

3. If we wish to extract a C-dense k-subgraph, we may assume that the maximum degree in G is at most Cn/k.

The justification for these assumptions is precisely as in Section 3.3.1.

In the modified algorithm, the backbone step and its analysis is the same as before. The main difference, as we mentioned, is in the hair step. The following lemma (analogous to Lemma 3.16) gives the guarantee of a hair step when the current set is intersected with the neighborhood of a cluster as opposed to the neighborhood of a single vertex.

Lemma 3.21. *Let $S \subseteq V$, and let $0 < w < k/2$. Then for any $C \geq 1$, either (a modified version of) DkS-Local(S, k) outputs a C-dense subgraph, or there exists a cluster of $W \subseteq V(H)$ of w vertices and a factor $f \geq 1/2$ s.t.*

$$\begin{aligned} |S \cap \Gamma(W)| &\leq f \cdot \frac{Cw(|S| + k)}{k}, \quad \textit{and} \\ |S \cap \Gamma(W) \cap H| &\geq f \cdot \frac{wd|S \cap H|}{Ck}. \end{aligned} \tag{3.4}$$

Proof. The proof follows precisely the same lines as that of Lemma 3.16, so we will only sketch it. We need the observation that (since the graph is bipartite)

$$\mathbb{E}_{W \subseteq H, |W| = w} |E(W, T)| = \frac{w}{k} \cdot \sum_{u \in H} |\Gamma(u) \cap T|,$$

for any set of vertices T.

Using this, and the same averaging lemma as before, we obtain the bounds stated. Note the extra C factor in the RHS of (3.4) (in the denominator). This loss is

[4]This is a trivial point, we should think of the original 'k' as $2k$.

because we need to move from $E(W,T)$ to $|\Gamma(W) \cap T|$ in order to obtain the desired bounds. If additionally we perform a max-density subgraph subroutine on the graph $(W, \Gamma(W) \cap T)$, we can ensure that we either find a C-dense subgraph, or $|\Gamma(W) \cap T| \geq \frac{|E(W,T)|}{C}$.

This completes the proof. (The above discussion also determines the change in the procedure DkS-Local which is required in this case). □

We now present the modified algorithm. The parameters are now as follows. Let $k = n^{1-\beta}$, and suppose $r/s = \beta + \varepsilon$ (as we outlined in the beginning of the section, we will use caterpillars corresponding to a slightly higher log-density). The main theorem is an analogue of Theorem 3.12. Using the same proof as the corollary to this, we obtain Theorem 3.19.

Theorem 3.22. *Suppose G, k are as above, and suppose $r/s = \beta + \varepsilon$. Further, suppose G has a (k,d) subgraph, with $d \geq C^3 k^{\beta(1-\varepsilon)}$. Then there exists an algorithm which runs in time $O\big(2^{n^{O(\varepsilon)}}\big)$ and finds a k-subgraph of density $\Omega(C)$.*

Note that as before, the pre-processing will ensure that we can assume a bound on the maximum degree, of $D = Cn/k = Cn^{\beta}$.

```
procedure DkS-Cat($G, k, r, s$)      // graph $G$, parameters $k, r, s$
begin
1       Start with $i = 0$ and $S_0 = V$.
2       for $i = 1, \dots, s$ do
3           Consider iteration $i$ in the construction of an $(r,s)$-caterpillar (Def. 3.1).
4           for each choice of $W_1, \dots, W_{L_i} \subseteq V$ of size $|W_j| = n^{\varepsilon}$ do
5               Compute the set of candidates $S_i$ for the rightmost internal vertex
                after fixing $W$'s as the 'leaf-clusters'.
6               If DkS-Local $(S, k)$ (the slight modification we mentioned) returns a
                $C$-dense subgraph, return it and exit.
```

A note on the running time. Note that guessing r clusters of size w correctly takes time at most $n^{O(wr)} = 2^{n^{O(\varepsilon)}}$, which is what we claimed.

We will now prove an inductive claim analogous to the one before. The main difference is that we will not explicitly keep track of an upper bound on $|S_i|$ at every step.

Theorem 3.23. *Suppose Algorithm 2 did not output a C-dense subgraph up to (and including) step i. Then there exist a set of leaves L_i, and a factor $f_i > 1$ for which the set of candidates for the last back-bone vertex S_i satisfies the two conditions*

$$|S_i \cap H| > f_i \cdot k^{\frac{ir}{s}+1-L_i} \tag{3.5}$$

$$\frac{|S_i \cap H|}{|S_i|} \geq f_i \cdot \left(\frac{k}{n}\right)^{\frac{ir}{s}+1-L_i} \tag{3.6}$$

Not maintaining an upper bound on $|S_i|$ turns out to be a technical issue. The place it comes up in the proof of Theorem 3.17 is in noting that the upper bound we maintain for $|S_i|$ is $> k$, so we can use that to to also bound $|S_i| + k$. In carrying out a proof with just a bound on the ratio, we need to ensure $|S_i|$ and $|S_i| + k$ are roughly equal. To see this, note that if $|S_i| < k$ before a leaf step, and the leaf step does not find a C-dense subgraph, then we end up with the contradiction that $|S \cap W(H) \cap H| > |S \cap W(H)|$! (using Lemma 3.21).

The rest of the proof follows along the lines of Theorem 3.17. We will sketch the part in which the choice of w being large is crucial.

Proof. The proof of the backbone step is exactly as in the proof of Theorem 3.17, so let us consider a hair step. Here the choice of w is important in claiming $|S_{i+1} \cap H|$ is large enough. Thus we need to check that

$$\frac{wd|S_i \cap H|}{Ck} \geq k^{\frac{(i+1)r}{s}+1-L_{i+1}} \iff \frac{wd}{C} \geq k^{r/s},$$

which is true, from our choice of parameters. This completes the inductive proof. □

Note that Theorem 3.22 now follows, because if we did not find a C-dense subgraph in any of the steps, we have a contradiction in the very last step ($i = s$).

Thus we could obtain a tradeoff between the running time and the approximation ratio. Such a phenomenon is quite interesting, and it would be nice to see it for other well-studied optimization problems. For instance, can we obtain an approximation ratio (significantly) better than $O(\sqrt{\log n})$ for sparsest cut if we allow, for example, $2^{n^{\varepsilon}}$ running time?

3.6 Spectral algorithms

Our algorithms for the DkS problem had a natural *barrier*, in terms of the log-density, as we saw earlier. Even in the Dense vs. Random question, algorithms based on "counting structures" cannot distinguish between a random graph with log-density ρ, and one which has a planted k-subgraph with denstiy $< k^{\rho}$. However we show in this section a surprisingly simple test based on the second eigenvalue which can overcome this seeming barrier for a certain range of the parameters.

We begin by recalling a well-known result of Füredi and Kómlos [41] about the second eigenvalue of random graphs.

Lemma 3.24 ([41])**.** *Let $G \sim G(n,p)$, for some $p = \omega(\log n/n)$. Let $\lambda_i(G)$ denote the ith largest eigenvalue of the adjacency matrix of G. Then we have*

$$\lambda_2(G) \le 2(1+o(1))(np(1-p))^{1/2} \quad w.h.p.$$

In the above, by "w.h.p." we mean with probability at least $1 - \frac{1}{\mathrm{poly}(n)}$ for any polynomial in n. Next we show that if there exists a k subgraph with sufficient density, we have a higher λ_2 for some range of parameters. In particular,

Lemma 3.25. *Let G be a graph with every vertex having a degree in the range $[n^\rho/2, 2n^\rho]$. Further, suppose H is an induced subgraph on k vertices with average degree $d > 8(n^{\rho/2} + \frac{kn^\rho}{n})$, and $k < n/4$. Then we have $\lambda_2(G) > 4n^{\rho/2}$.*

Proof. We will exhibit two orthogonal vectors x s.t. the value of $\frac{x^T Ax}{x^T x} > 4n^{\rho/2}$, where A is the adjacency matrix of G. From the variational definition of eigenvalues, the desired bound then follows.

The first vector is $x = \mathbf{1}$, i.e., the vector with all entries being 1. The value of the above ratio for this vector is at least $n^\rho/2$, because every vertex has degree at least this quantity. This is much larger than $4n^{\rho/2}$ for any constant $\rho < 1$ and large enough n.

The next vector we consider is x defined by (assume the coordinates are indexed by the vertices):

$$x_i = \begin{cases} 1, \text{ if } i \in V(H) \\ -\frac{k}{n-k} \text{ otherwise} \end{cases}$$

Note that x is orthogonal to the vector $\mathbf{1}$. Further, if A is the adjacency matrix of G, we have that

$$\begin{aligned} \frac{x^T Ax}{x^T x} &= \frac{\sum_{(i,j)\in E(G)} x_i x_j}{k + \frac{nk^2}{(n-k)^2}} \\ &\geq \frac{kd - 2kn^\rho \cdot \frac{k}{n-k}}{2k} \\ &\geq \frac{d}{2} - \frac{2kn^\rho}{n} \geq 4n^{\rho/2}. \end{aligned}$$

We used the fact that every vertex has degree at most $2n^\rho$ and the lower bound on d from the hypothesis. □

Thus for the Dense vs. Random problem, for the case of paramaters $\rho \leq 1/2$, $k = n^{1-\rho}$, the above lemmas show that if $d > 10n^{\rho/2}$, we can solve the distinguishing problem by computing the second eigenvalue. On the other hand, if we just used counting based algorithms, we need $d > k^\rho = n^{(1-\rho)\rho}$, which could be much larger (since we are dealing with $\rho < 1/2$). Thus the eigenvalue approach does better than counting based ones for certain ranges of parameters. An interesting open question is if this can be done for arbitrary graphs. More precisely,

Open Problem 3.26. *Can spectral/SDP based methods be used to obtain an $n^{\rho/2}$ approximation for arbitrary graphs? (at least in the case $\rho < 1/2$ for the entire graph)*

Chapter 4

Finding ‘Structure’ in Matrices

The study of matrices and their properties has a rich history, having applications in various branches of mathematics and science. We will consider a tiny speck in this vast subject – a class of problems related to the spectra of graphs.

As outlined in the introduction, these questions have applications in various branches of Computer Science, Engineering and Mathematics, from image segmentation [69] and clustering [53], to machine learning, and the theory of Markov chains [52] and graph partitioning [1]. We will now study some of these questions, and motivate the questions we will consider the chapters that follow.

4.1 Some questions

Let us now give a flavor of the kind of questions we will be studying. They will mainly be continuous optimization problems. They are generally not convex (apart from singular values), and they help reveal some “hidden structure” in the matrix (or some times in a natural graph which can be associated with the matrix).

In what follows, it is convenient to think of a rectangular matrix $A \in \Re^{m \times n}$ (in some of the problems we will choose $m = n$).

4.1.1 Singular values

Eigenvalues and Singular values are a basic tool in matrix analysis. They also have enormous important in practice, for instance in low rank approximations and Principal Component Analysis (PCA). The largest right singular value of a matrix A (as above) is defined to be

$$\max_{x \in \Re^n} \frac{\|Ax\|_2}{\|x\|_2},$$

over all nonzero x (the vector v which optimizes this quantity is called the largest singular vector). The singular vector denotes the direction which is the *most stretched* by the matrix A. It can also be shown that the outer product uv^T (where u is the largest *left* singular vector) is the best rank-one approximation to the matrix A, in terms of the Frobenius norm of the difference.

The largest singular value of A is also the maximum *eigenvalue* of the matrix A^TA, and computed quickly by *power iteration*. I.e., we start with a random vector u_0, and at time step i, set $u_i = A^TAu_{i-1}$. This procedure can be shown to converge *quickly* to the singular vector u (the rate turns out to depend on the gap between the largest and second largest eigenvalues of A^TA).

This simple procedure is often used in practice, since it is highly parallelizable and is very simple to implement in a distributed setting (for instance, in the computation of the so-called *personalized page-rank* [50]).

4.1.2 Cut norm and the Gröthendieck problem

The maximum singular vectors give the best approximation of a given matrix using a rank-one matrix. A somewhat related question is that of the *Maximum sum subrectangle*, i.e., given A, find sets $I \subseteq [m], J \subseteq [n]$ so as to maximize $|\sum_{i\in I, j\in J} a_{ij}|$. This quantity is also called the *cut-norm* of a matrix.

The cut norm plays a role in obtaining good approximations for *dense instances* of CSPs [39, 4] and is also closely related to low rank approximations. Alon and Naor gave the first constant factor approximation algorithm for the cut-norm of a matrix [6], and their main idea was to first compute the quantity

$$\|A\|_{\infty\mapsto 1} := \max_{x_i, y_j \in \{-1,1\}} \sum_{i,j} a_{ij} x_i y_j,$$

and then show that this quantity is within a constant factor of the cut norm.

The quantity $\|A\|_{\infty\mapsto 1}$ is then approximated to a constant factor by rounding a semidefinite programming relaxation (in which x_i, y_j are relaxed to be unit vectors). The fact that the relaxation has a small gap is called the *Gröthendieck's inequality.* One of the main directions in the thesis will be to systematically study such *mixed norms* of operators (i.e., matrices). We will come to these in a moment.

4.1.3 Quadratic programming

A generalization of the quantity $\|A\|_{\infty\mapsto 1}$ is what is called *quadratic programming.* Suppose we are given a matrix A (think of it as the adjacency matrix of a graph, with potentially negative weights). The objective is to find

$$QP(A) := \max_{x_i \in \{-1,1\}} \sum_{i,j} a_{ij} x_i x_j.$$

In the case when the graph describing A is bipartite, it corresponds to the question of computing $\|B\|_{\infty\mapsto 1}$ for an appropriate matrix B. The problem of computing $QP(A)$ is very well-studied in the optimization community, and an $O(\log n)$ approximation algorithm was obtained independently by [60, 59, 29].

This is one of the simplest *Gaussian rounding* schemes for semidefinite programs, and was also used by Charikar and Wirth [29] to approximate the maximum Cutgain in a graph. The algorithm also illustrates that semidefinite programs can *capture* $x_i = \{-1, 1\}$ constraints in a fairly general class of optimization problems, up to an $O(\log n)$ factor loss in approximation.

4.2 A common umbrella – operator norms

Let us take a slightly different standpoint and view the matrix A as an operator mapping $\Re^n \mapsto \Re^m$. The norm of an operator from one normed space to the other is defined to be the maximum *stretch* caused by the operator to a unit vector (in the domain). Suppose we consider the ℓ_q norm on $\Re^n$ and ℓ_p norm on $\Re^m$ in the above (recall that $\|x\|_q = (|x_1|^q + \cdots + |x_n|^q)^{1/q}$), where $p, q \geq 1$. The $q \mapsto p$ operator norm is now by definition,

$$\|A\|_{q\mapsto p} := \max_{x \in \Re^n, x \neq 0} \frac{\|Ax\|_p}{\|x\|_q}.$$

Operator norms have been studied in different contexts in both Computer Science and Operations Research. They can also *capture* (for different values of p, q) problems of different flavors. For instance, $p = q = 3$ is the well-studied notion of the spectral norm (or the maximum singular value), while $p = 1, q = \infty$ arises in approximation algorithms for the cut norm, as we have seen.

In this chapter and the next, we attempt a systematic study of the complexity of approximating the value of $\|A\|_{q\mapsto p}$ for different values of p, q. This will illustrate the different ideas which are needed for the different regimes of parameters.

Before we begin, we mention some applications. Since the problem is a natural one, it appears under different guises in the scientific computing community. For instance, $p \mapsto p$ norms are used in matrix condition number estimates [51]. $q \mapsto p$ norms have also been studied in connection to a recent paradigm, "robust optimization" [72]. We will later see some applications in computer science in detail. The first is a recent result of Englert and Räcke [34], in which they use the computation of matrix

$p \mapsto p$ norms as a subroutine for an ℓ_p version of the oblivious routing problem. We will also see applications of computing the $q \mapsto p$ norms in the *hypercontractive* regime to problems such as small set expansion and certifying restricted isometry (see Sections 5.4 and 5.6).

4.2.1 Known results and related work.

Very few provable guarantees are known about approximating $\|A\|_{q \mapsto p}$ in general. For computing $p \mapsto p$ norms (which are somewhat more studied), polynomial time algorithms for arbitrary A are known to exist only for $p = 1, 2$, and ∞. In these cases, the algorithms are quite simple (for $p = 1, \infty$, the maximizing vector will have only one non-zero coordinate). For the general problem, when $p \leq 2$, $q > 2$, Nesterov[61] shows that the problem can be approximated to a constant factor (around 2.3), using a semidefinite programming relaxation. When the matrix has only non-negative entries, this relaxation can be shown to be exact [72].

For other ranges of p, q, the best known bounds are polynomial factor approximations, obtained by 'interpolation'. For instance, to compute $\|A\|_{p \mapsto p}$ for some p, one computes the vectors which are maximizers for $p = 1, 2, \infty$, and pick the one with the best value for the $p \mapsto p$ objective. This can be shown to give an $O(n^{1/4})$ approximation for all p (see [51]). For the general problem of computing $\|A\|_{q \mapsto p}$, Steinberg [72] studies algorithms of this nature, and gives an algorithm with an overall approximation guarantee of $O(n^{25/128})$ (slightly better than $n^{1/5}$). This works by taking into account the approximation algorithms of Nesterov along with the $1, 2, \infty$ maximizers.

The guarantees of these algorithms are shown using a combination of Hölder's inequality, and the following simple fact about norms which will also be useful to us.

Observation 4.1. *Let A be an $m \times n$ real matrix, and p, q be parameters, and let p', q' be the* dual *norms, i.e., $1/p + 1/p' = 1$, and similarly for q. Then we have*

$$\|A\|_{q \mapsto p} = \|A^T\|_{p' \mapsto q'}. \tag{4.1}$$

Proof. The proof is by a simple application of the *dual* definition of the ℓ_p norm, i.e.,

$$\|x\|_p = \min_{y \,:\, \|y\|_{p'} \le 1} y^T x.$$

Plugging this into the definition of the norm gives the desired equality. □

The lemma will allow us to *move* from one range of parameters to another in our algorithms and hardness proofs.

A related problem in the same realm is the so-called "L_p Grothendieck problem". This has been studied by [56], and the aim is to compute (given an $n \times n$ matrix B and a parameter p),

$$\max_{x \neq 0} \frac{x^T B x}{\|x\|_p^2}.$$

For $p \ge 2$, it is possible to obtain an $O(p)$ factor approximation algorithm using the following relaxation (which is convex since $p \ge 2$):

$$\text{maximize} \sum_{i,j} B_{ij} X_{ij}, \quad \text{subject to}$$

$$X \succeq 0$$

$$\sum_{i,j} X_{ii}^{p/2} \le 1.$$

For $p < 2$, the last constraint is not convex. This is an "SDP" relaxation for the problem, and it can be rounded to obtain an $O(p)$ approximation algorithm for any B. The approximation ratio is also shown in [56] to be tight assuming the Unique

Games Conjecture. Finally, note that the case $p = \infty$ is precisely the Quadratic Programming problem.

The question of computing $\|A\|_{p\mapsto 2}$ is a special case of the L_p Grothendieck problem (one in which $B \succeq 0$). In this case (i.e., $B \succeq 0$), constant factor approximation algorithms are known, due to [61].

Recently, [32] study the problem of finding the best k-dimensional subspace approximation to a set of points, where one wishes to minimize the ℓ_p distances to the subspace. When $k = n - 1$ this can be shown to reduce to the L_p Grothendieck problem for the matrix A^{-1}.

On the hardness front, the problem of computing $q \mapsto p$ norms is known to be NP-hard (to compute exactly) in the range $q \geq p \geq 1$ [72], but no inapproximability results were known prior to our work. Subsequently,

4.2.2 Our contributions

We will give both algorithmic and hardness results.

Non-negative matrices. Let us first consider the case of matrices A with non-negative entries. Here we prove that if $1 \leq p \leq q$, then $\|A\|_{q\mapsto p}$ can be computed in polynomial time. More precisely we give an algorithm which gives a $(1+\delta)$ approximation in time polynomial in n, m, and $(1/\delta)$.

This is the first polynomial time guarantee (to the best of our knowledge) for computing the matrix $q \mapsto p$-norms for any 'general enough' p, q. The algorithm is a very natural one – it is based on a natural fixed point iteration, and was first proposed by [24]. We give an analysis of the running time of this algorithm and prove convergence to the optimum.

The fixed point iteration is a generalization of the familiar *power method* to compute the eigenvectors of a matrix. In fact, heuristic approaches to many optimization problems involve finding solutions via fixed point computations. It would be interesting to see if ideas from our analysis could potentially be useful in such other settings.

Note that computing the matrix $q \mapsto p$ norm is a problem of maximizing a convex function over a convex domain. While a convex function can be minimized efficiently over convex domains using gradient descent based algorithms, it is in general hard to maximize them. So what makes the problem of computing $\|A\|_{q \mapsto p}$ (with $p \leq q$ and positive entries) easy?

We show that in this case, the problem still *weakly* satisfies properties which hold for convex optimization. For instance, if we are trying to minimize a convex function over a convex domain, the *lower* level sets (set of points for which the function is lower than a threshold) are convex. In our case, it turns out that the *upper* level sets of the objective function are *simply connected* on the unit sphere (i.e., the set of points on the sphere with value more than a threshold looks like a "connected patch", though it need not be convex – see the analysis for a precise statement). Furthermore, it turns out that there is a unique local and global maximum, which is also a useful property of convex optimization.

We will discuss an application of the algorithm to the problem of *oblivious routing* in the ℓ_p norm. Englert and Räcke [34] recently showed that there exists an oblivious routing scheme which attains a competitive ratio of $O(\log n)$ when the objective function is the ℓ_p-norm of the flow vector ($|E|$ dimensional vector). However, they can efficiently compute this oblivious routing scheme only for $p = 2$. Using our algorithm, and some structural properties that we establish about the maxima (Section 5.3), we can make the result of [34] constructive. Here matrix p-norm computation is used as a 'separation oracle' in a multiplicative weights style update, and this gives an $O(\log n)$-competitive oblivious routing scheme for all ℓ_p-norms ($p \geq 1$).

Hardness of approximation. For general matrices (with negative entries allowed), we show the hardness of approximation up to an *almost polynomial factor* for computing the $q \mapsto p$ norm of general matrices when $q \geq p$ and both p, q are > 2. By duality, this implies the same hardness when both p, q are < 2 and $q \geq p$.[1]

For these ranges, we first prove that approximating $\|A\|_{q \mapsto p}$ upto any constant factor is NP-hard. Under the stronger assumption that $\mathsf{NP} \notin \mathsf{DTIME}(2^{\text{polylog}(n)})$, we prove that the problem is hard to approximate to a factor of $\Omega(2^{(\log n)^{1-\varepsilon}})$, for any constant $\varepsilon > 0$.

Techniques. We first consider the case $q = p$, for which we show constant factor hardness by a gadget reduction from the gap version of MaxCut. Then we show that the $p \mapsto p$ norm multiplies upon tensoring, and thus we get the desired hardness amplification.

While the proof of the small constant factor hardness carries over to the $q \mapsto p$ norm case with $q > p > 2$, in general these norms do not multiply under tensoring. We handle this by giving a way of starting with a hard instance of $p \mapsto p$ norm computation (with additional structure, as will be important), and convert it to one of $q \mapsto p$ norm computation.

The hardness results are interesting because they show the inapproximability of an optimization problem over the reals by reduction from combinatorial problems like MaxCut.

4.2.3 Hypercontractive norms and open problems

All our algorithms and hardness results apply to the case $p \leq q$, but we do not know either of these (even for non-negative matrices) for $p > q$ (which is rather surprising!). In this case, the quantity $\|A\|_{q \mapsto p}$ is called a *hypercontractive norm.*

[1]When $p \leq 2$ and $q \geq 2$, Nesterov's algorithm gives a constant factor approximation.

Our algorithmic results do not apply here. Indeed, the property which we saw for the $p \le q$ (and non-negative A) case, that the level sets are *connected* does not hold for $p > q$. Further we could have multiple optima and we do not know how to obtain efficient approximation algorithms for the norm.

On the hardness front, the $q < p$ case seems more related to questions like the Densest k-subgraph problem (informally, when the matrix is positive and $p < q$, if there is a 'dense enough' submatrix, the optimum vector would have most of its support on it). We will describe connections to problems in compressed sensing and some questions related to graph expansion in Section 5.6.

Making progress towards (either algorithmically or to prove hardness of approximation of) computing $q \mapsto p$ matrices for $p > q$ is a challenging problem which we leave open. We mention that some recent works [14] show inapproximability results for special p, q.

4.3 Maximum density subgraph in matrices

Finally, we will propose and study a problem which is a variant of two problems which we understand quite well: maximum density subgraph (Section 1.2.1) and quadratic programming (Section 4.1.3). We will call this the QP-Ratio problem, and it is formally stated as follows: given an $n \times n$ matrix A with zero diagonal,

$$\textsf{QP-Ratio}: \quad \max_{\{-1,0,1\}^n} \frac{\sum_{i \ne j} a_{ij} x_i x_j}{\sum x_i^2} \tag{4.2}$$

An alternate phrasing of this question is as follows: select a subset of non-zero variables S and assign them values in $\{\pm 1\}$ so as to maximize the ratio of the quadratic programming objective $\sum_{i<j \in S} a_{i,j} x_i x_j$ to the (normalized) size of S. Thus it is like quadratic programming, with the additional flexibility of being to throw away some *outliers* and restrict to a sub-matrix – the variables set to 0 are considered outliers,

and the goal is to maximize the solution quality on the rest of the solution. Note that the numerator itself is the quadratic programming objective $\sum_{i<j} a_{i,j}x_ix_j$, and can be maximized by setting all variables to be ± 1. However, the denominator term in the objective makes it worthwhile to set variables to 0.

If all the a_{ij} are positive, it is precisely the maximum density subgraph problem (because in this case there is no need to set any $x_i = -1$). A *normalized* version of the problem also comes up in Trevisan's spectral algorithm for Max Cutgain [74]. In some sense, the difficulty of this problem restricts the approximation ratio he can obtain for Max Cutgain.

Our interest in this problem however, is primarily from a *relaxation* point of view: semidefinite programming techniques have proved very useful for quadratic optimization problems (i.e. problems with a quadratic objective) over $\{0,1\}$ variables or $\{\pm 1\}$ variables (starting with the seminal work of Goemans and Williamson [45], and later work on constraint satisfaction problems). However it seems very difficult to capture the constraint $x_i \in \{-1,0,1\}$ using semidefinite programs (or more generally with convex relaxations). We introduce QP-Ratio as a problem we hope will capture and help understand this difficulty.

4.3.1 Our contributions

We first consider convex programming relaxations for QP-Ratio, their strengths and weaknesses (i.e., rounding algorithms and integrality gaps).

The most natural relaxation is based on the eigenvalue, which turns out to have an integrality gap of $\Omega(\sqrt{n})$. The main problem is due to contribution from non-zero variables which have very different magnitudes. We then consider a semidefinite programming (SDP) relaxation in which we can add constraints to try to *enforce* that this cannot happen. This turns out to perform somewhat better – in particular, we show how to round it to obtain an $\widetilde{O}(n^{1/3})$ approximation algorithm.

We can prove that the strengthened SDP has a better approximation ratio for special instances. For example, it is *exact* when A has only non-negative entries (which is precisely the maximum density subgraph question). Another special case is that of *bipartite* instances – i.e., instances of QP-Ratio where the support of a_{ij} is the adjacency matrix of a bipartite graph (akin to bipartite instances of quadratic programming, also known as the Grothendieck problem). For bipartite instances, we obtain an $\tilde{O}(n^{1/4})$ approximation and an almost matching the SDP integrality gap of $\Omega(n^{1/4})$.

Since the best algorithm gives only an $O(n^{1/3})$ factor approximation in general, it is natural to try proving inapproximability results. Alas, the situation here seems rather similar to that of the DkS problem. The best hardness result we can show assuming $P \neq NP$ is to rule out a PTAS. However, like in the DkS problem, we can give other forms of evidence for the problem's hardness.

We give evidence based on two conjectures, that it is hard to approximate QP-Ratio to within any constant factor. We remark that current techniques seem insufficient to prove such a result based on standard assumptions (such as $P \neq NP$). A similar situation exists for other problems with a ratio objective such as sparsest cut.

First, we rule out constant factor approximation algorithms for QP-Ratio assuming that random instances of k-AND are hard to distinguish from 'well-satisfiable' instances. This hypothesis was used as a basis to prove optimal hardness for the so called 2-Catalog problem (see [37]) and has proven fruitful in ruling out $O(1)$-approximations for the densest subgraph problem (see [2]). It is known that even very strong SDP relaxations (in particular, $\Omega(n)$ rounds of the Lasserre hierarchy) cannot refute this conjecture [75].

We also show a reduction from Ratio UG (a ratio version of the well studied unique games problem), to QP-Ratio. Ratio UG is an interesting problem worthy of study that could shed light on the complexity of other optimization questions. The technical challenge in our reduction is to develop Fourier-analytic machinery which lets us handle long-code reductions in the context of ratio objectives. We will see these in detail in Section 6.3.4.

There is a big gap in the approximation guarantee of our algorithm and our inapproximability results. We suspect that the problem is in fact hard to approximate to an n^{ε} factor for some $\varepsilon > 0$. In Section 6.3.1, we decribe a natural distribution over instances which we believe are hard to approximate up to polynomial factors. Like in the DkS problem, all existing techniques seem to fail to provide good approximations even for this (very simple) distribution over instances.

Chapter 5

Approximating Matrix Norms

In this chapter, we will study the approximability of $\|A\|_{q\mapsto p}$ for different values of p, q, as outlined in Section 4.2.2. On the algorithms side, for non-negative matrices and $q \geq p$, we analyze a simple algorithm based on power iteration which was first proposed by Boyd [24]. He showed that the method converges to a global maximum, but to the best of our knowledge, no bounds on the time of convergence were known.

We will show that the algorithm converges in polynomial time, and further establish additional properties, like the uniqueness of the optimum and structure of the level sets for this range of parameters. These properties turn out to be useful in applications (section 5.4) and also provide an *explanation* as to why we are able to solve a non-convex optimization problem!

The chapter is organized as follows. The algorithm for positive matrices is presented in Section 5.2.1, followed by its analysis, i.e., the correctness and bounds on the running time (Section 5.2.2). Then we discuss additional structural properties alluded to earlier, such as the uniqueness of the optimum, in Section 5.3. Then, we will see an application to the problem of oblivious routing in the ℓ_p norm (section 5.4). We then end the discussion of the $p \leq q$ case by discussing the inapproximability of the general problem (section 5.5): we first show a constant factor hardness for $\|A\|_{p\mapsto p}$

(Section 5.5.1), and show how to amplify it (Section 5.5.2). Then we use this to show hardness for $\|A\|_{q \mapsto p}$ in section 5.5.3, with $p \le q$. Finally, we will discuss a little about hypercontractive norms (the case $p > q$) and questions related to approximating it.

5.1 Notation and simplifications

Unless specified otherwise, A will denote an $m \times n$ matrix with real entries. Further, we will denote by A_i the ith row of A (which is an n dimensional vector). Also a_{ij} denotes the element in the ith row and jth column. Similarly for a vector $\mathbf{x}$, we denote the ith co-ordinate by x_i.

We write $\mathbb{R}_+$ for the set of non-negative reals. We say that a vector $\mathbf{x}$ is *positive* if the entries x_i are all > 0. Finally, for two vectors $\mathbf{x}, \mathbf{y}$, we write $\mathbf{x} \propto \mathbf{y}$ to mean that $\mathbf{x}$ is *proportional to* $\mathbf{y}$, i.e., $\mathbf{x} = \lambda \mathbf{y}$ for some λ (in all places we use it, λ will be > 0).

Our algorithmic results will focus on non-negative matrices A. However it will be much more convenient to work with matrices where we restrict the entries to be in $[1/N, 1]$, for some parameter N (i.e., zero entries can cause minor problems). Let us see how we can assume this without loss of generality, if we only wish to obtain a $(1+\delta)$ approximation to the norm in time roughly poly$(m, n, \frac{1}{\delta})$.

First off, by padding with zeroes appropriately, we can focus on the case $m = n$. Now suppose we are interested in a $(1+\delta)$ approximation. We can first scale A such that the largest entry is 1, pick $N \approx n^2/\delta$, and work with the matrix $A + \frac{1}{N}J$ (here J is the $n \times n$ matrix of ones). The following simple observation ensures that we obtain a good approximation.

Lemma 5.1. *Let A be an $n \times n$ matrix with maximum entry equal to 1. Let J be the $n \times n$ all ones matrix.*

$$\|A + \varepsilon J\|_{q \mapsto p} \le \|A\|_{q \mapsto p}\left(1 + \varepsilon n^{1+\frac{1}{p}-\frac{1}{q}}\right)$$

In what follows, we will thus assume all entries to lie in $[1/N, 1]$ for some parameter N, and we will refer to such A as a *positive matrix*.

5.2 An algorithm for positive matrices

In this section, we consider positive $n \times n$ matrices A, and prove that if $1 < p \leq q$, we can efficiently compute $\|A\|_{q \mapsto p}$. To start, let us define $f : \mathbb{R}^n \mapsto \mathbb{R}$ by

$$f(\mathbf{x}) = \frac{\|A\mathbf{x}\|_p}{\|\mathbf{x}\|_q} = \frac{(\sum_i |A_i \mathbf{x}|^p)^{1/p}}{(\sum_i |x_i|^q)^{1/q}}.$$

We present an algorithm due to Boyd [24], and prove that it converges quickly to the optimum vector. The idea is to consider ∇f, and rewrite the equation $\nabla f = 0$ as a fixed point equation (i.e., as $S\mathbf{x} = \mathbf{x}$, for an appropriate operator S). The iterative algorithm then starts with some vector $\mathbf{x}$, and applies S repeatedly. Note that in the case $p = 2$, this mimics the familiar power iteration (in this case S will turn out to be multiplication by the matrix $A^T A$).

5.2.1 Algorithm description

Let us start by looking at ∇f. In particular, $\frac{\partial f}{\partial x_i}$ is equal to

$$\frac{\|x\|_q \|Ax\|_p^{1-p} \cdot \sum_j a_{ij} |A_j x|^{p-1} - \|Ax\|_p \|x\|_q^{1-q} \cdot |x_i|^{q-1}}{\|x\|_q^2} \tag{5.1}$$

At a critical point, $\frac{\partial f}{\partial x_i} = 0$ for all i. Thus for all i,

$$|x_i|^{q-1} = \frac{\|x\|_q^q}{\|Ax\|_p^p} \cdot \sum_j a_{ij} |A_j x|^{p-1} \tag{5.2}$$

Define an operator $S : \mathbb{R}^n_+ \to \mathbb{R}^n_+$, with the ith co-ordinate of Sx being (note that all terms involved are positive)

$$(Sx)_i = \Big(\sum_j a_{ij}(A_j x)^{p-1}\Big)^{1/(q-1)}$$

Thus, at a critical point of f, we have $Sx \propto x$. Thus if we want to find a critical point of f, it is natural to consider the the following algorithm:

```
procedure FindNorm($A, p, q, \delta$)
 // $n \times n$ matrix $A$, parameters $p, q$, accuracy $\delta$
begin
1   Initialize $x = \frac{\mathbf{1}}{\|\mathbf{1}\|_p}$, and let $T := (Nn)\text{polylog}(N, n, 1/\delta)$.
2   for $i = 1 \ldots T$ do
3       set $x \leftarrow Sx$.
4       normalize $x$ to make $\|x\|_q = 1$.
5   return $x$.
```

We can then hope that the algorithm converges quickly to a critical point of f (i.e., it does not cycle, for instance). This will be the main focus of the remainder of the section. It turns out from our assumption on A, and since $p \le q$, that every *positive* fixed point of the iteration is also a critical point. Further, there will be a unique positive fixed point, which is also the unique maximum of f.

5.2.2 Analyzing the algorithm

We will treat $f(\mathbf{x})$ as defined over the domain $\mathbb{R}^n_+$. Since the matrix A is positive, the maximum must be attained in $\mathbb{R}^n_+$. Since f is invariant under scaling x, we restrict our attention to points in $\mathcal{S}^n_q = \{x \;:\; x \in \mathbb{R}^n_+, \|x\|_q = 1\}$. Thus the algorithm starts with a point in $\mathcal{S}^n_q$, and in each iteration moves to another point, until it converges.

First, we prove that the maximum of f over $\mathbb{R}^n_+$ occurs at an interior point (i.e., none of the co-ordinates are zero). It will then follow that if x^* denote a point at which maximum is attained, i.e., $f(x^*) = \|A\|_{q\mapsto p}$ (x^* need not be unique), we have $\nabla f = 0$ at x^*, and so x^* is a fixed point for the iteration.

Lemma 5.2. *Let $x^* \in \mathcal{S}_q^n$ be a point at which f attains maximum. Then each co-ordinate of x^* is at least $\frac{1}{(Nn)^2}$.*

Proof. Let x^* be the optimum vector, and suppose $\|x^*\|_q = 1$. Consider the quantity

$$f(x^*)^p = \frac{\sum_i (A_i x^*)^p}{\left(\sum_i (x^*)^q\right)^{p/q}}.$$

First, note that $x_i^* \neq 0$ for any i. Suppose there is such an i. If we set $x_i = \delta$, each term in the numerator above increases by at least $\frac{p\cdot\delta}{N^p}$ (because $A_i x^*$ is at least $\frac{1}{N}$, and $(\frac{1}{N} + \frac{\delta}{N})^p > \frac{1}{N^p}(1+p\delta)$), while the denominator increases from 1 to $(1+\delta^q)^{p/q} \approx 1 + (p/q)\delta^q$ for small enough δ. Thus since $q > 1$, we can set δ small enough and increase the objective. This implies that x^* is a positive vector.

Note that $A_j x^* \geq \frac{\mathbf{1}}{N} \cdot x^* \geq \frac{1}{N}$ (because the $\|x^*\|_1 \geq \|x^*\|_q = 1$). Thus for every i,

$$(Sx^*)_i^{q-1} = \sum_j a_{ij}(A_j x^*)^{p-1} \geq \frac{n}{N^p}.$$

Further, $\|A\|_p^p \leq n^{p+1}$, because each $a_{ij} \leq 1$ and so $A_j x \leq n x_{\max}$ (where $x_{\max}$ denotes the largest co-ordinate of x). Now since Eqn.(5.2) holds for x^*, we have

$$n^{p+1} \geq \|A\|_p^p = \frac{(Sx^*)_i^{q-1}}{(x^*)_i^{q-1}} \geq \frac{n}{N^p (x^*)_i^{q-1}}.$$

This implies that $x_i^* > \frac{1}{(Nn)^2}$, proving the lemma (we needed to use $q \geq p > 1$ to simplify). $\square$

The next lemma (shown in [24]) shows that with each iteration, the value of the function cannot decrease.

Lemma 5.3. *([24]) For any vector x, we have*

$$\frac{\|ASx\|_p}{\|Sx\|_q} \geq \frac{\|Ax\|_p}{\|x\|_q}$$

The analysis of the algorithm proceeds by maintaining two *potentials*, defined by

$$m(x) = \min_i \frac{(Sx)_i}{x_i} \qquad \text{and} \qquad M(x) = \max_i \frac{(Sx)_i}{x_i}.$$

Note that these are somewhat unusual potentials, tailored for positive vectors. The idea is that if x is a fixed point, then $Sx \propto x$, and thus $m(x) = M(x)$. In this case, from the computation in section 5.2.1, each of these quantities is equal to $\left(\frac{\|x\|_q^q}{\|Ax\|_p^p}\right)^{1/(q-1)}$. As observed in [24], these quantites can be used to 'sandwich' the norm – in particular,

Lemma 5.4. *For any positive vector x with $\|x\|_q = 1$, we have*

$$m(x)^{q-1} \leq \|A\|_{q \mapsto p}^p \leq M(x)^{q-1}$$

The lemma is powerful: it relates the norm (which we wish to compute) to certain quantities we can compute starting with *any* positive vector x. This is also the part in the analysis in which we crucially use $p \leq q$. Let us now give a proof of this lemma. Our proof has the additional advantage that it immediately implies the following

Corollary 5.5. *[Unique maximum] The maximum of f on $\mathcal{S}_q^n$ is attained at a unique point x^*. Further, this x^* is the unique critical point of f on $\mathcal{S}_q^n$ (which also means it is the unique fixed point for the iteration).*

We first prove the Lemma above.

Proof (of Lemma 5.4). Let $x \in \mathcal{S}_q^n$ be a positive vector. Let $x^* \in \mathcal{S}_q^n$ be a vector which maximizes $f(x)$.

The first inequality is a simple averaging argument:

$$\frac{\sum_i x_i \cdot (Sx)_i^{q-1}}{\sum_i x_i \cdot x_i^{q-1}} = \frac{\sum_i x_i \cdot \sum_j a_{ij}(A_j x)^{p-1}}{\sum_i x_i^q} \tag{5.3}$$

$$= \frac{\sum_j (A_j x)^{p-1} \sum_i a_{ij} x_i}{\sum_i x_i^q} \tag{5.4}$$

$$= \frac{\sum_j (A_j x)^p}{\sum_i x_i^q} = \frac{\|Ax\|_p^p}{\|x\|_q^q} \le \|A\|_{q\mapsto p}^p \tag{5.5}$$

The last inequality uses $\|x\|_q = 1$. Thus there exists an index i such that $(Sx)_i^{q-1}/x_i^{q-1} \le \|Ax\|_{q\mapsto p}$.

The latter inequality is more tricky – it gives an upper bound on $f(x^*)$, no matter which $x \in \mathcal{S}_q^n$ we start with. To prove this, we start by observing that x^* is a fixed point, and thus for all k,

$$m(x^*)^{q-1} = M(x^*)^{q-1} = \frac{(Sx^*)_k^{q-1}}{(x^*)_k^{q-1}}.$$

Call this quantity λ. Now, let $\theta > 0$ be the smallest real number such that $x - \theta x^*$ has a zero co-ordinate, i.e., $x_k = \theta x_k^*$, and $x_j \ge \theta x_j^*$ for $j \ne k$. Since $\|x\|_q = \|x^*\|_q$ and $x \ne x^*$, θ is well-defined, and $x_j > \theta x_j^*$ (strictly) for some index j. Because of these, and since each a_{ij} is strictly positive, we have $Sx > S(\theta x^*) = \theta^{(p-1)/(q-1)} S(x^*)$ (clear from the definition of S).

Now, for the index k, we have

$$\frac{(Sx)_k^{q-1}}{x_k^{q-1}} > \frac{\theta^{p-1}(Sx^*)_k^{q-1}}{(\theta x_k^*)^{q-1}} = \theta^{(p-q)} \cdot \lambda$$

Thus we have $M(x)^{q-1} > \lambda$ (since $q \ge p$, and $0 < \theta < 1$), which is what we wanted to prove. □

We can now prove the following.

Proof of Corollary 5.5. Let $x^* \in \mathcal{S}_q^n$ denote a vector which maximizes f over $\mathcal{S}_q^n$ (thus x^* is one fixed point of S). Suppose, if possible, that y is another fixed point. By the calculation in Eq.(5.5) (and since y is a fixed point and x^* maximizes f), we have

$$M(y)^{q-1} = m(y)^{q-1} = \frac{\|Ay\|_p^p}{\|y\|_q^q} = f(y)^q \leq f(x^*)^p$$

Now since $y \neq x^*$, the argument above (of considering the smallest θ such that $y - \theta x^*$ has a zero co-ordinate, and so on) will imply that $M(y)^{q-1} > \lambda = f(x^*)^p$, which is a contradiction.

This proves that there is no other fixed point. $\square$

The next few lemmas say that as the algorithm proceeds, the value of $m(x)$ increases, while $M(x)$ decreases. Further, it turns out we can quantify how much they change: if we start with an x such that $M(x)/m(x)$ is 'large', the ratio drops significantly in one iteration. The proofs of these are fairly straightforward.

Lemma 5.6. *Let x be a positive vector. Then $m(x) \leq m(Sx)$, and $M(x) \geq M(Sx)$.*

Proof. Suppose $m(x) = \lambda$. So for every i, we have $(Sx)_i \geq \lambda x_i$. Now fix some index i and consider the quantity

$$(SSx)_i^{q-1} = \sum_j a_{ij}(A_j Sx)^{q-1}.$$

Since A is a positive matrix and $(Sx)_i \geq \lambda x_i$, we must have $(A_j Sx) \geq \lambda \cdot (A_j x)$ for every j. Thus

$$(SSx)_i^{q-1} \geq \lambda^{q-1} \sum_j a_{ij}(A_j x)^{q-1} = \lambda^{q-1}(Sx)_i^{q-1}.$$

This shows that $m(Sx) \geq \lambda$. A similar argument shows that $M(Sx) \leq M(x)$. $\square$

Lemma 5.7. *Let* $\mathbf{x}$ *be a positive vector with* $\|\mathbf{x}\|_q = 1$*, and suppose* $M(\mathbf{x}) \geq (1 + \alpha)m(\mathbf{x})$*. Then* $m(S\mathbf{x}) \geq \big(1 + \frac{\alpha}{Nn}\big)m(\mathbf{x})$*.*

Proof. Let $m(\mathbf{x}) = \lambda$, and suppose k is an index such that $(S\mathbf{x})_k \geq (1+\alpha)\lambda \cdot x_k$ (such an index exists because $M(\mathbf{x}) > (1+\alpha)\lambda$. In particular, $(S\mathbf{x}) \geq \lambda\mathbf{x} + \alpha\lambda \cdot \mathbf{e}_k$, where $\mathbf{e}_k$ is the standard basis vector with the kth entry non-zero. Thus we can say that for every j,

$$A_j(S\mathbf{x}) \geq \lambda A_j\mathbf{x} + \alpha\lambda A_j\mathbf{e}_k.$$

The second term will allow us to quantify the improvement in $m(\mathbf{x})$. Note that $A_j\mathbf{e}_k = a_{jk} \geq \frac{1}{Nn}A_j\mathbf{1}$ (since A_{jk} is not too small). Now $\mathbf{1} \geq \mathbf{x}$ since $\mathbf{x}$ has q-norm 1, and thus we have

$$A_j(S\mathbf{x}) \geq \big(1 + \frac{\alpha}{Nn}\big)\lambda \cdot A_j\mathbf{x}$$

Thus $(SS\mathbf{x})_i^{q-1} \geq \big(1+\frac{\alpha}{Nn}\big)^{q-1}\lambda^{q-1}(S\mathbf{x})_i^{q-1}$, implying that $m(S\mathbf{x}) \geq \big(1+\frac{\alpha}{Nn}\big)\lambda$. □

This immediately implies that the value $\|A\|_{q\mapsto p}$ can be computed quickly. In particular,

Theorem 5.8. *For any* $\delta > 0$*, after* $O(Nn \cdot \text{polylog}(N, n, \frac{1}{\delta}))$ *iterations, the algorithm of Section 5.2.1 finds a vector* x *such that* $f(x) \geq (1-\delta)f(x^*)$

Proof. To start with, the ratio $\frac{M(x)}{m(x)}$ is at most Nn (since we start with $\mathbf{1}$, and the entries of the matrix lie in $[1/N, 1]$). Lemma 5.7 now implies that the ratio drops from $(1+\alpha)$ to $(1+\frac{\alpha}{2})$ in Nn iterations. Thus in $T = (Nn)\text{polylog}(N, n, 1/\delta)$ steps, the x we end up with has $\frac{M(x)}{m(x)}$ at most $\big(1 + \frac{\delta}{(Nn)^c}\big)$ for any constant c. This then implies that $f(x) \geq f(x^*)\big(1 - \frac{\delta}{(Nn)^c}\big)$, after T iterations. □

5.3 Proximity to the optimum

The argument above showed that the algorithm finds a point x such that $f(x)$ is close to $f(x^*)$. We proved that for positive matrices, x^* is unique, and thus it is natural to ask if the vector we obtain is 'close' to x^*.

We can prove that the x we obtain after $T = (Nn)\text{polylog}(N, n, 1/\delta)$ iterations is 'close' to x^*. The rough outline of the proof is the following: we first show that $f(\mathbf{x})$ is strictly concave 'around' the optimum.[1] Then we show that the 'level sets' of f are 'connected' (precise definitions follow). Then we use these to prove that if $f(x)$ is close to $f(x^*)$, then $x - x^*$ is 'small' (the choice of norm does not matter much).

Some of these results are of independent interest, and shed light into why the $q \mapsto p$ problem may be easier to solve when $p \le q$ (even for non-negative matrices).

Concavity around the optimum. We now show that the neighborhood of every critical point (where ∇f vanishes) is strictly concave. This is another way of proving that every critical point is a maximum (this was the way [34] prove this fact in the $p = q$ case).

Taking partial derivatives of $f(x) = \frac{\|Ax\|_p}{\|x\|_q}$, we observe that

$$\frac{\partial f}{\partial x_i} = f(x)\Big(\frac{\sum_k (A_k x)^{p-1} a_{ki}}{\|Ax\|_p^p} - \frac{x_i^{q-1}}{\|x\|_q^q}\Big) \tag{5.6}$$

where A_k refers to the k^{th} row of matrix A. Now, consider a critical point z, with $\|z\|_q = 1$ (w.l.o.g.). We can also always assume that w.l.o.g. the matrix A is such that $\|Az\|_p = 1$. Thus at a critical point z, as in Eq.(5.2), we have that for all i:

$$\sum_k (A_k z)^{p-1} a_{ki} = z_i^{q-1} \tag{5.7}$$

[1]It is easy to see that in general, f is not convex everywhere.

Computing the second derivative of f at z, and simplifying using $\|Az\|_p = \|z\|_q = 1$, we obtain

$$\frac{1}{p} \cdot \frac{\partial^2 f}{\partial x_i \partial x_j}\bigg|_z = (p-1)\sum_k (A_k z)^{p-2} a_{ki} a_{kj} + (q-p) z_i^{q-1} z_j^{q-1} \tag{5.8}$$

$$\frac{1}{p} \cdot \frac{\partial^2 f}{\partial x_i^2}\bigg|_z = (p-1)\sum_k (A_k z)^{p-2} a_{ki}^2 + (q-p) z_i^{2q-2} - (q-1) z_i^{q-2} \tag{5.9}$$

We will now show that the Hessian H_f is negative semi-definite, which proves that f is strictly concave at the critical point z. Let ε be any vector in $\mathbb{R}^n$. Then we have (the $(q-1)z_i^{q-2}$ in (5.9) is split as $(p-1)z_i^{q-2} + (q-p)z_i^{q-2}$, and $\sum_{i,j}$ includes the case $i = j$)

$$\begin{aligned}
&\varepsilon^T H_f \varepsilon = \\
&p(p-1)\Big(\sum_{i,j}\sum_k (A_k z)^{p-2} \cdot a_{ki} a_{kj} \cdot \varepsilon_i \varepsilon_j - \sum_i z_i^{q-2} \varepsilon_i^2\Big) \\
&\quad + p(q-p)\Big(\sum_{i,j} (z_i z_j)^{q-1} \varepsilon_i \varepsilon_j - \sum_i z_i^{q-2} \varepsilon_i^2\Big) \\
&\equiv \quad T_1 + T_2 \qquad \text{(say)}
\end{aligned}$$

We consider T_1 and T_2 individually and prove that they are negative. First consider T_2. Since $\sum_i z_i^q = 1$, we can consider z_i^q to be a probability distribution on integers $1, \ldots, n$. Cauchy-Schwartz now implies that $\mathbb{E}_i[(\varepsilon_i/z_i)^2] \geq \big(\mathbb{E}_i[(\varepsilon_i/z_i)]\big)^2$. This is equivalent to

$$\sum_i z_i^q \cdot \frac{\varepsilon_i^2}{z_i^2} \geq \Big(\sum_i z_i^q \cdot \frac{\varepsilon_i}{z_i}\Big)^2 = \sum_{i,j} z_i^q z_j^q \cdot \frac{\varepsilon_i \varepsilon_j}{z_i z_j} \tag{5.10}$$

Noting that $q \geq p$, we can conclude that $T_2 \leq 0$. Now consider T_1. Since z is a fixed point, it satisfies Eq. (5.7), thus we can substitute for x_i^{q-1} in the second term of T_1.

Expanding out $(A_k z)$ once and simplifying, we get

$$\begin{aligned}
\frac{T_1}{p(p-1)} &= \sum_{i,j}\sum_k (A_k z)^{p-2} a_{ki} a_{kj} \Big(\varepsilon_i \varepsilon_j - \frac{z_j}{z_i} \cdot \varepsilon_i^2 \Big) \\
&= -\sum_k (A_k z)^{p-2} \sum_{i,j} a_{ki} a_{kj} \cdot z_i z_j \cdot \Big(\frac{\varepsilon_i}{z_i} - \frac{\varepsilon_j}{z_j} \Big)^2 \\
&\le 0
\end{aligned}$$

This proves that f is concave around any critical point z.

Level sets of f. Let $\mathcal{S}_q^n$, as earlier, denote the (closed, compact) set $\{\mathbf{x} \in \mathbb{R}_+^n \ : \ \|\mathbf{x}\|_q = 1\}$. Let $\mathcal{N}_\tau$ denote $\{x \in \mathcal{S}_q^n \ : \ f(x) \ge \tau\}$, i.e., $\mathcal{N}_\tau$ is an 'upper level set'. (it is easy to see that since f is continuous and A is positive, $\mathcal{N}_\tau$ is closed).

Let $S \subseteq \mathcal{S}_q^n$. We say that two points x and y are *connected* in S, if there exists a path (a continuous curve) connecting x and y, entirely contained in S (and this is clearly an equivalence relation). We say that a set S is *connected* if every $x, y \in S$ are connected in S. Thus any subset of $\mathcal{S}_q^n$ can be divided into connected components. With this notation, we show ([34] proves the result when $p = q$).

Lemma 5.9. *The set $\mathcal{N}_\tau$ is connected for every $\tau > 0$.*

This follows easily from techniques we developed so far.

Proof. Suppose if possible, that $\mathcal{N}_\tau$ has two disconnected components S_1 and S_2. Since there is a unique global optimum x^*, we may suppose S_1 does not contain x^*. Let y be the point in S_1 which attains maximum (of f) over S_1 (y is well defined since $\mathcal{N}$ is closed). Now if $\nabla f|_y = \vec{0}$, we get a contradiction since f has a unique critical point, namely x^* (Lemma 5.5). If $\nabla f|_y \neq \vec{0}$, it has to be normal to the surface $\mathcal{S}_q^n$ (else it cannot be that y attains maximum in the connected component S_1). Let $\mathbf{z}$ be the direction of the (outward) normal to $\mathcal{S}_q^n$ at the point y. Clearly, $\langle \mathbf{z}, y \rangle > 0$ (intuitively this is clear; it is also easy to check).

We argued that $\nabla f|_y$ must be parallel to $\mathbf{z}$, and thus it has a non-zero component along y – in particular if we scale y (equivalent to moving along y), the value of f changes, which is clearly false! Thus $\mathcal{N}_\tau$ has only one connected component. $\square$

In what follows, we bear in mind Lemma 5.2. We now show that if $x \in \mathcal{S}_q^n$ is 'far' from x^*, then $f(x)$ is bounded away from $f(x^*)$. This, along with the fact that $\mathcal{N}_\tau$ is connected for all τ, implies that if $f(x)$ is very close to $f(x^*)$, then $\|x - x^*\|_1$ must be small. For ease of calculation, we give the formal proof only for $p = q$ (this is also the case which is used in the oblivious routing application). It should be clear that as long as we have that the Hessian at x^* is negative semidefinite, and third derivatives are bounded, the proof goes through.

Lemma 5.10 (Stability). *Suppose $x \in \mathcal{S}_q^n$, with $\|x - x^*\|_1 = \delta \le \frac{1}{(Nn)^{12}}$. Then*

$$f(x) \le f(x^*)\Big(1 - \frac{\delta^2}{(Nn)^6}\Big) \tag{5.11}$$

Proof. Let ε denote the 'error vector' $\varepsilon = x - x^*$. We will use the Taylor expansion of f around x^*. H_f denotes the Hessian of f and g_f is a term involving the third derivatives, which we will get to later. Thus we have: (note that ∇f and H_f are evaluated at x^*)

$$f(x) = f(x^*) + \varepsilon \cdot \nabla f_{|x^*} + \frac{1}{2}\,\varepsilon^T H_{f|x^*}\varepsilon + g_f(\varepsilon') \tag{5.12}$$

At x^*, the ∇f term is 0. From the proof above that the Hessian is negative semidefinite, we have

$$\varepsilon^T H_f \varepsilon = \tag{5.13}$$
$$-\,p(p-1)\sum_s (A_s x^*)^{p-2}\Big(\sum_{i,j} a_{si}a_{sj}x_i^* x_j^* \big(\frac{\varepsilon_i}{x_i^*} - \frac{\varepsilon_j}{x_j^*}\big)^2\Big)$$

We want to say that if $\|\varepsilon\|_1$ is large enough, this quantity is sufficiently negative. We should crucially use the fact that $\|x^*\|_p = \|x^* + \varepsilon\|_p = 1$ (since x is a unit vector in p-norm). This is the same as

$$\sum_i |x_i^* + \varepsilon_i|^p = \sum_i |x_i^*|^p.$$

Thus not all ε_i are of the same sign. Now since $\|\varepsilon\|_1 > \delta$, at least one of the ε_i must have absolute value at least δ/n, and some other ε_j must have the opposite sign, by the above observation. Now consider the terms corresponding to these i, j in Eqn.(5.13). This gives

$$\begin{aligned} \varepsilon^T H_f \varepsilon &\le -p(p-1) \sum_s (A_s x^*)^{p-2} \cdot a_{si} a_{sj} \cdot \frac{x_j^*}{x_i^*} \cdot \frac{\delta^2}{n^2} \qquad (5.14) \\ &\le -p(p-1) \sum_s (A_s x^*)^{p-2} \frac{(\sum_i a_{si})^2}{(Nn)^2} \cdot \frac{1}{(Nn)^2} \cdot \frac{\delta^2}{n^2} \\ &\le -p(p-1) \cdot \frac{\delta^2}{N^4 n^6} \cdot \|Ax^*\|_p^p \end{aligned}$$

Note that we used the facts that entries a_{ij} lie in $[\frac{1}{N}, 1]$ and that $x_i^* \in [\frac{1}{(Nn)^2}, 1]$. Thus it only remains to bound the third order terms (g_f, in Eqn.(5.12)). This contribution equals

$$\begin{aligned} g_f(\varepsilon) = \frac{1}{3!} \sum_i \varepsilon_i^3 \frac{\partial^3 f}{\partial x_i^3} + \frac{1}{2!} \sum_{i,j} \varepsilon_i^2 \varepsilon_j \frac{\partial^3 f}{\partial x_i^2 \partial x_j} \qquad (5.15) \\ + \sum_{i<j<k} \varepsilon_i \varepsilon_j \varepsilon_k \frac{\partial^3 f}{\partial x_i \partial x_j \partial x_k} \end{aligned}$$

It can be shown by expanding out, and using the facts that $m_{si} \le N(M_s x^*)$ and $\frac{1}{x_i^*} \le (Nn)^2$, that for i, j, k,

$$\frac{\partial^3 f}{\partial x_i \partial x_j \partial x_k} \le 10 p^3 (Nn)^3 \|Ax^*\|_p^p.$$

Thus, the higher order terms can be bounded by

$$g_f(\varepsilon) \le 10p^3 \cdot n^6 N^3 \cdot \|Ax^*\|_p^p \cdot \delta^3$$

So, if $\delta < \frac{1}{10p^3} \cdot \frac{1}{(Nn)^{12}}$, the Hessian term dominates. Thus we have, as desired:

$$f(x) \le f(x^*)\Big(1 - \frac{\delta^2}{(Nn)^6}\Big)$$

□

This proves that the vector we obtain at the end of the T iterations (for T as specified) has an ℓ_1 distance at most $\frac{1}{(Nn)^c}$ to x^*. Thus we have a polynomial time algorithm to compute x^* to any accuracy.

5.4 An application to oblivious routing

While the algorithm of the earlier section is rather specialized, i.e., it applies only to matrices with non-negative entries, there are many situations in which this restriction naturally applies. We will consider one instance now – an application to oblivious routing in the ℓ_p norm, a problem studied by [34].

Background. Oblivious routing is a notion first introduced by Harald Räcke in the context of routing multicommodity flows in graphs to minimize congestion. The problem is the following: given a graph $G = (V, E)$, the aim is to come up with a "routing scheme", i.e., a means to route unit flow between every pair of vertices (*without* any pre-defined set of demands). Now given a set of demands, we can route these by scaling the flows in the routing scheme linearly by the demands and taking the union of these flows. Note that this is the reason it is called an "oblivious" routing scheme (the scheme itself was defined without knowledge of the demands). The aim is to remain "competitive" with the best flow in hindsight – namely the best flow for this set of demands. The measure used to compare, especially in older works, is often the maximum congestion of an edge.

A remarkable fact proved by Räcke in [64] is that for any graph, there exists an oblivious routing scheme which achieves a competitive ratio of $O(\log n)$ with respect to the maximum congestion objective. Note that the maximum congestion can be seen as the ℓ_∞ norm of the "vector of aggregate flow", if we assume the capacities are all unit.

Another natural objective, which measures the *energy* of a flow is the ℓ_2 norm of the vector of aggregate flow. (It is inspired by the physical interpretation of the flow of current through a wire). More generally Englert and Räcke studied the question of finding oblivious routing schemes which are competitive w.r.t. the ℓ_p norm of the aggregate flow vector. In [34], they proved that there *exist* routing schemes which achieve a competitive ratio of $O(\log n)$, by using a careful argument based on duality.

To make their algorithm constructive, a main bottleneck is the lack of an algorithm to compute the $p \mapsto p$ norm of a matrix that arises naturally. It turns out that the entries of the matrix are the flows on edges due to certain tree-based routing schemes, and hence are all non-negative. Thus we can use the algorithm of the earlier section, along with some additional ideas to construct an oblivious routing scheme with $O(\log n)$ competitive ratio! The stability of the maxima (Lemma 5.10) plays a crucial role in the process.

Why care about an ℓ_p objective? Gupta et al. [47] first considered this question, and the general motivation is as follows: in the case $p = \infty$, this is the problem of minimizing the maximum congestion, for which the celebrated result of [64] gave an $O(\log n)$ competitive scheme. For ℓ_1, the optimal solution routes along shortest paths for each demand pair. The ℓ_p version is a way to trade-off between these two extremes.

Further, cases such as $p = 2$ are of interest because of their physical interpretation, as we mentioned earlier.

Zero-sum game framework of [34]. Englert and Räcke showed that there exists a routing scheme which gives a competitive ratio of $O(\log n)$ for the ℓ_p objective. Their proof, roughly speaking, is as follows. The first observation is that given a routing scheme, we can *compute* the competitive ratio. In particular, it turns out that for any routing scheme, the "worst" set of demands is when all the demands are across edges in the graph (and further more, the optimum routing for these demands is to simply route along the corresponding edges).

Thus, given a routing scheme, we can consider an $m \times m$ matrix M, where m is the number of edges in the graph, defined as follows: M_{ij} is the flow on edge i due to a unit demand across edge j (according to the routing scheme). Thus the competitive ratio of the routing scheme is precisely the $p \mapsto p$ norm of M! Thus [34] aim to obtain a tree-based routing scheme which leads to a matrix of norm $O(\log n)$. To prove the existence of such a scheme, they consider the two player game: the row player is the set of tree-based routing matrices M, and the column player is the set of demand vectors d which satisfy $\|d\|_p = 1$. The loss for the row player (or the reward for the column player) is the quantiy $\|Md\|_p$.

The key observation in [34] is that the game has a *pure* Nash equilibrium. Using this, they obtain an upper bound on the value of the game, due to the following lemma:

Lemma 5.11. *[34, 64] For any given vector $u \in \mathbb{R}^m$, there exists an tree-based routing scheme (denoted by matrix F_u) such that $\|F_u u\|_p \leq O(\log n)\|u\|_p$.*

The proof of this uses the ℓ_∞ version of Räcke's result, thus in some sense the ℓ_p version is *reduced* to the ℓ_∞ case.

Now how do we make this constructive? We can follow a natural multiplicative weights style approach (this was also done in [34] for $p = 2$). The key thing to note is that Lemma 5.11 is constructive. Thus suppose we have a procedure which gives a *good* routing matrix F_u given a demand vector u. The algorithm is as follows.

```
procedure FindRouting($G, \vec{c}$)
 // graph $G$ with $m$ edges, capacities $\vec{c}$ on the edges
begin
1    Start with any tree based routing $M_0$.
2    for $i = 0, \dots, n^6$ do
3        If $\|M_i\|_{p\mapsto p} \le C \log n$ (large constant $C$), return $M_i$, else let $d$ be a
         vector satisfying $\|M_i d\|_p \ge (1-\theta)\|M_i\|_{p\mapsto p}$, for $\theta = 1/n^6$.
4        Compute the matrix $F_d$ using the lemma above.
5        Set $M_{i+1} = (1-\varepsilon)M_i + \varepsilon F_d$, for $\varepsilon = 1/n^3$.
6    return the current $M$.
```

To see why the algorithm works, the key is to treat the quantity $\|M_i\|_{p\mapsto p}$ as a potential, and claim that it drops by a factor at at least $1/\text{poly}(n)$ in each iteration (unless the algorithm terminated and returned an M_i with small norm).

The reason is intuitively the following: consider some step i. Let d be the vector with $\|d\|_p = 1$ and $\|Md\|_p$ is maximized. Let M_{i+1} be obtained as in our update. How large can $\|M_{i+1}\|_{p\mapsto p}$ be? For vectors d' which are *close* to d, we will have $\|M_{i+1}d'\|_p$ will be only around $(1-\varepsilon)\|M_i d\|_p$, because of the update, and because F_d is a good routing matrix for d. Now from our *stability* result (lemma 5.10), we know that for any d' which is *far* from d, the quantity $\|M_i d\|_p$ was small enough to begin with, and adding a small F_d does not change things much! This shows that the potential $\|M_i\|_{p\mapsto p}$ reduces by a $1/\text{poly}(n)$ factor each time.

Let us now formally prove all these claims. We start with a simple lemma (which follows from the continuity of an appropriate function).

Lemma 5.12. *Let $f = \frac{\|Ax\|_p^p}{\|x\|_p^p}$, where $A_{n\times n}$ has a minimum entry $\frac{1}{N}$ and let $\mathbf{y}$ be a vector (n dimensions) with minimum entry $\frac{1}{(Nn)^2}$. Let $\mathbf{x}$ be close to $\mathbf{y}$ i.e. $\|\mathbf{x}-\mathbf{y}\|_1 = \delta \le \frac{1}{(Nn)^{12}}$. Then, $f(x) \le f(y) + 1$.*

Theorem 5.13. *There exists a polynomial time algorithm that computes an oblivious routing scheme with competitive ratio $O(\log n)$ when the aggregation function is the ℓ_p norm with $p > 1$ and the load function on the edges is a norm.*

Proof. The one technical detail to ensure we can apply our results, is to add a small multiple of J (the all-ones matrix) to the matrices we work with (before computing the norm, for instance). We also note that a convex combination of tree based routings is also a tree based routing, thus we always work with matrices arising out of such schemes.

It suffices to prove that if the procedure FindRouting does not return a good matrix at step i, then $\|M_i\|_{p\mapsto p} \le \|M_{i-1}\|_{p\mapsto p} - 1/n^4$. Suppose that every time we find a routing matrix via Lemma 5.11, we have $\|F_d d\|_p \le C\log n/4$ (where C is the constant used in the algorithm). Then we have the following (we do not try to optimize for the best polynomial factors in this proof). Let us fix some step i, and let d be the vector which maximizes $\|M_i x\|_p$ over $\|x\|_p = 1$.

Now for all vectors y that are far enough from d, one has $\|M_i y\|_p \le \|M_i y\|_p - \frac{1}{\text{poly}(n)}$ from Lemma 5.10 (stability). Thus we have $\|M_{i+1} y\|_p \le \|M_i y\|_p - \frac{1}{\text{poly}(n)}$ for all such vectors y. On the other hand, consider y in the δ-neighborhood of d. Using Lemma 5.12,

$$\|F_d y\|_p \le \frac{C\log n}{2}$$

Hence,

$$\begin{aligned}\|M_{i+1}y\|_p &= (1-\varepsilon)\|M_i y\|_p + \frac{C\varepsilon}{2}\log n \\ &\le \|M_i y\|_p - \varepsilon \times \frac{c}{2}\log n \qquad (\text{since } \|M_i y\|_p \ge C\log n\ ) \\ &\le \|M_i y\|_p - \frac{1}{\mathrm{poly}(n)}\end{aligned}$$

Hence, it follows that the matrices M_i decrease in their p-norm by a small quantity $\Omega(\frac{1}{\mathrm{poly}(n)})$ in every step. It follows that this iterative algorithm finds the required tree-based oblivious routing scheme in poly(n) steps. □

This completes the proof of Theorem 5.13, and it shows a simple application in which the computation of the matrix p-norm is used as a *separation oracle* in a multiplicative weights style algorithm.

5.5 Inapproximability results

We will now prove that it is NP-hard to approximate $\|A\|_{q\mapsto p}$-norm of a matrix to any fixed constant, for any $q \ge p > 2$. We then show how this proof carries over to the hardness of computing the $\infty \mapsto p$ norm.

5.5.1 Inapproximability of $\|A\|_{p\mapsto p}$

Let us start with the question of approximating $\|A\|_{p\mapsto p}$. Our first aim will be to prove the following

Lemma 5.14. *For any $p > 2$, there exists a constant $\eta > 1$ s.t. it is NP-hard to approximate $\|A\|_{p\mapsto p}$ to a factor better than η.*

The proof will by a gadget reduction from the gap version of the MaxCut problem, whose hardness is well-known. We will then amplify the gap η by tensoring.

Theorem 5.15 ([49]). *There exist constants $1/2 \leq \rho < \rho' < 1$ such that given a regular graph $G = (V, E)$ on n vertices and degree d, it is hard to distinguish between:*
Yes *case: G has a cut containing at least $\rho'(nd/2)$ edges, and*
No *case: No cut in G cuts more that $\rho(nd/2)$ edges.*

Suppose we are given a graph $G = (V, E)$ which is regular and has degree d. The p-norm instance we consider will be that of maximizing $g(x_0, \ldots, x_n)$ ($x_i \in \mathbb{R}^n$), defined by

$$g(x_0, x_1, \ldots, x_n) = \frac{\sum_{i\sim j} |x_i - x_j|^p + Cd \cdot \big(\sum_i |x_0 + x_i|^p + |x_0 - x_i|^p\big)}{n|x_0|^p + \sum_i |x_i|^p}.$$

Here C will be chosen appropriately later. Writing $t(u) = |x_0 + u|^p + |x_0 - u|^p$ for convenience, we can see $g(\mathbf{x})$ as the ratio

$$\frac{\sum_{i\sim j} \big(|x_i - x_j|^p + C[t(x_i) + t(x_j)]\big)}{\sum_{i\sim j} 2|x_0|^p + |x_i|^p + |x_j|^p}. \tag{5.16}$$

The idea is to do the analysis on an edge-by-edge basis. Define

$$f(x, y) := \frac{|x - y|^p + C\big(|1 + x|^p + |1 - x|^p + |1 + y|^p + |1 - y|^p\big)}{2 + |x|^p + |y|^p}$$

The gadget. The function f is the key *gadget* in our proof. It is designed s.t. it takes a somewhat large value when x, y are both ± 1 and have opposite signs, and a smaller value otherwise. We will now make this formal.

Definition 5.16. *We will call a real number x "good" if $|x| \in [1 - \varepsilon, 1 + \varepsilon]$, and bad otherwise. A* pair *(x, y) of reals is called good if both x, y are good and they have opposite signs.*

We now show two technical lemmas (their proofs are straightforward computations). Note also that $p > 2$ is crucial in these.

Lemma 5.17. *For all $x \in \mathbb{R}$, we have*

$$\frac{|1+x|^p + |1-x|^p}{1+|x|^p} \le 2^{p-1}.$$

Further, for any $\varepsilon > 0$, there exists a $\delta > 0$ such that if $|x| \notin [1-\varepsilon, 1+\varepsilon]$, then

$$\frac{|1+x|^p + |1-x|^p}{1+|x|^p} \le 2^{p-1} - \delta.$$

Proof. We may assume $x > 0$. First consider $x > 1$. Write $x = 1 + 2\theta$, and thus the first inequality simplifies to

$$\big[(1+2\theta)^p - (1+\theta)^p\big] \ge (1+\theta)^p - 1 + 2\theta^p.$$

Now consider

$$I = \int_{x=1}^{1+\theta} \big((x+\theta)^{p-1} - x^{p-1}\big)dx.$$

For each x, the function being integrated is $\ge \theta^{p-1}$, since $p > 2$ and $x > 0$. Thus the integral is at least θ^p. Now evaluating the integral independently and simplifying, we get

$$(1+2\theta)^p - 2(1+\theta)^p + 1 \ge p \cdot \theta^p,$$

which gives the inequality since $p > 2$. Further there is a slack of $(p-2)\theta^p$. Now suppose $0 < x < 1$. Writing $x = 1 - 2\theta$ and simplifying similarly, the inequality follows. Further, since we always have a slack, the second inequality is also easy to see. $\square$

Lemma 5.18. *For any $\varepsilon > 0$, and large enough C, we have that*

$$f(x,y) \le \begin{cases} C \cdot 2^{p-1} + \frac{(1+\varepsilon)2^p}{2+|x|^p+|y|^p}, & \text{if } (x,y) \text{ is good} \\ C \cdot 2^{p-1} & \text{otherwise} \end{cases} \tag{5.17}$$

Proof. The proof is a fairly straight-forward case analysis. Let $b(x)$ denote a predicate which is 1 if x is bad and 0 otherwise.

Case 1. (x,y) is good. The upper bound in this case is clear (using Lemma 5.17).

Case 2. Neither of x, y are bad, but $xy > 0$. Using Lemma 5.17, we have $f(x,y) \le C \cdot 2^{p-1} + \varepsilon$, which is what we want.

Case 3. At least one of x, y are bad (i.e., one of $b(x), b(y)$ is 1). In this case Lemma 5.17 gives (writing $t(x) = 1 + |x|^p$, and $\Delta(x) = 2^{p-1} - \delta b(x)$)

$$\begin{aligned} f(x,y) &\le \frac{|x-y|^p + C\big(t(x)\Delta(x) + t(y)\Delta(y)\big)}{2 + |x|^p + |y|^p} \\ &= C \cdot 2^{p-1} + \frac{|x-y|^p - C\big(\delta b(x)t(x) + \delta b(y)t(y)\big)}{2 + |x|^p + |y|^p} \end{aligned}$$

Since $|x-y|^p \le 2^{p-1}(|x|^p + |y|^p)$, and one of $b(x), b(y) > 0$, we can choose C large enough (depending on δ), so that $f(x,y) \le C \cdot 2^{p-1}$. □

Soundness. Assuming the lemma, let us see why the analysis of the No case follows. Suppose the graph has a Max-Cut value at most ρ, i.e., every cut has at most $\rho \cdot nd/2$ edges. Now consider the vector x which maximizes $g(x_0, x_1, \dots, x_n)$. It is easy to see that we may assume $x_0 \neq 0$, thus we can scale the vector s.t. $x_0 = 1$. In the following lemma, let $S \subseteq V$ denote the set of 'good' vertices (i.e., vertices for which $|x_i| \in (1-\varepsilon, 1+\varepsilon)$).

Lemma 5.19. *The number of good edges is at most $\rho \cdot \frac{(|S|+n)d}{4}$.*

Proof. Recall that good edges have both end-points in S, and further the corresponding x values have opposite signs. Thus the lemma essentially says that there is no cut in S with $\rho \cdot \frac{(|S|+n)d}{4}$ edges.

Suppose there is such a cut. By greedily placing the vertices of $V \setminus S$ on one of the sides of this cut, we can extend it to a cut of the entire graph with at least

$$\begin{aligned}\rho \cdot & \frac{(|S|+n)d}{4} + \frac{(n-|S|)d}{4} \\ &= \frac{\rho n d}{2} + \frac{(1-\rho)(n-|S|)}{4} > \frac{\rho n d}{2}\end{aligned}$$

edges, which is a contradiction. This finishes the proof of the lemma. □

Let $\mathcal{N}$ denote the numerator of Eq.(5.16). We have

$$\begin{aligned}\mathcal{N} &= \sum_{i\sim j} f(x_i,x_j)(2+|x_i|^p+|x_j|^p) \\ &\le C \cdot 2^{p-1} \cdot \Big(nd + d\sum_i |x_i|^p\Big) + \sum_{i\sim j,\ \text{good}} (1+\varepsilon)2^p \\ &\le Cd \cdot 2^{p-1} \cdot \Big(n + \sum_i |x_i|^p\Big) + \frac{\rho d(n+|S|)}{4} \cdot 2^p(1+\varepsilon).\end{aligned}$$

Now observe that the denominator is $n+\sum_i |x_i|^p \ge n+|S|(1-\varepsilon)^p$, from the definition of S. Thus we obtain an upper bound on $g(x)$

$$g(x) \le Cd \cdot 2^{p-1} + \frac{\rho d}{4} \cdot 2^p(1+\varepsilon)(1-\varepsilon)^{-p}.$$

Completeness, and hardness factor. In the Yes case, there is clearly an assignment of ± 1 to x_i such that $g(\mathbf{x})$ is at least $Cd \cdot 2^{p-1} + \frac{\rho' d}{4} \cdot 2^p$. Thus if ε is small enough (C is chosen sufficiently large depending on ϵ), the gap between the optimum values in the Yes and No cases can be made $\big(1+\frac{\Omega(1)}{C}\big)$, where the $\Omega(1)$ term is determined by the difference $\rho' - \rho$. This proves that the p-norm is hard to approximate to some fixed constant factor. Note that in the analysis, ε was chosen to be a small constant depending on p and $\rho' - \rho$.

The instance. Let us now formally write out the instance of the $\|A\|_{p\mapsto p}$ problem which we used in the reduction. This will be useful when arguing about certain properties of the tensored instance, which we need for proving hardness of $\|A\|_{q\mapsto p}$ for $p < q$.

Let the instance of MaxCut we are reducing from be $G = (V, E)$. First we do a simple change of variable and let $z = n^{1/p}x_0$. Now, we construct the $5|E| \times (n+1)$ matrix M (we have 5 rows per edge $e = (u, v)$). This matrix takes the same value $\|M\|_p$ as g. Further in the YES case, there is a vector $x = (n^{1/p}, x_1, x_2, \ldots, x_n)$ with $x_i = \pm 1$, that attains a value $(C.d.2^{p-1} + \rho' d.2^{p-2})$.

5.5.2 Amplifying the gap by tensoring

We observe that the matrix $p \mapsto p$-norm is multiplicative under tensoring (it is well known to be true for eigenvalues i.e. for $p = 2$). The tensor product $M \otimes N$ is defined in the standard way – we think of it as an $m \times m$ matrix of blocks, with the i, jth block being a copy of N scaled by m_{ij}. More precisely,

Lemma 5.20. *Let M, N be square matrices with dimensions $m \times m$ and $n \times n$ respectively, and let $p \geq 1$. Then $\|M \otimes N\|_p = \|M\|_p \cdot \|N\|_p$.*

While Lemma 5.20 is stated for square matrices, it also works with rectangular matrices because we can pad 0s to make it square. We note that it is crucial that we consider $\|A\|_p$. Matrix norms $\|A\|_{q\mapsto p}$ for $p \neq q$ do not in general multiply upon tensoring. Let us now prove the lemma above.

Proof. Let $\lambda(A)$ denote the p-norm of a matrix A (resp., B). Let us first show the easy direction, that $\lambda(M \otimes N) \geq \lambda(M) \cdot \lambda(N)$. Suppose x, y are the vectors which 'realize' the p-norm for M, N respectively. Then

$$\begin{aligned}\|(M \otimes N)(x \otimes y)\|_p^p &= \sum_{i,j} |(M_i \cdot x)(N_j \cdot y)|^p \\ &= \Big(\sum_i (M_i \cdot x)^p\Big)\Big(\sum_j (N_j \cdot y)^p\Big) \\ &= \lambda(M)^p \cdot \lambda(N)^p\end{aligned}$$

Also $\|x \otimes y\|_p = \|x\|_p \cdot \|y\|_p$, thus the inequality follows.

Now for the other direction, i.e., we wish to show $\lambda(M \otimes N) \leq \lambda(M) \cdot \lambda(N)$. Consider an mn dimensional vector x, and let $z := (A \otimes B)x$. We will think of x, z as being divided into m blocks of size n each. Further by $x^{(i)}$ (and $z^{(i)}$), we denote the vector in $\mathbb{R}^n$ which is formed by the ith block of x (resp. z).

By definition, we have:

$$\begin{aligned}\|z\|_p^p &= \sum_k \|z^{(k)}\|_p^p, \quad \text{and} \\ z^{(k)} &= \sum_i a_{ki} B x^{(i)}.\end{aligned}$$

Now, let $u^{(i)} := Bx^{(i)}$ for $1 \leq i \leq m$, and define $v^{(j)}$ to be vectors in $\mathbb{R}^m$ defined for $1 \leq j \leq n$ as the vectors formed by collecting the jth entry in the vectors $u^{(i)}$ for $1 \leq i \leq m$. Thus by the above, we have

$$\|z\|_p^p = \sum_k \|\sum_i a_{ki} u^{(i)}\|_p^p \leq \sum_j \lambda(A)^p \|v^{(j)}\|_p^p.$$

The last inequality is the tricky bit – this is by noting that $u^{(i)}$ is an n-dimensional vector for each i, and we can expand out $\|u^{(i)}\|_p^p$ as sum over these n dimensions (if we call the summation variable j, we collect terms by j), and use the fact that for any vector x, $\|Ax\|_p^p \leq \lambda(A)^p \|x\|_p^p$.

We are now almost done. Consider the quantity $\sum_j \|v^{(j)}\|_p^p$. This is precisely equal to

$$\sum_i \|u^{(i)}\|_p^p = \sum_i \|Bx^{(i)}\|_p^p \leq \lambda(B)^p \|x\|_p^p.$$

Combining this with the above, we obtain $\|z\|_p^p \leq \lambda(A)^p \lambda(B)^p \|x\|_p^p$, which is what we set out to prove. □

Hence, given any constant $\gamma > 0$, we repeatedly tensor the instance of the matrix M from Proposition 5.14 $k = \log_\eta \gamma$ times ($M' = M^{\otimes k}$), to obtain the following:

Theorem 5.21. *For any $\gamma > 0$ and $p \geq 2$, it is NP-hard to approximate the p-norm of a matrix within a factor γ. Also, it is hard to approximate the matrix p-norm to a factor of $\Omega(2^{(\log n)^{1-\varepsilon}})$ for any constant $\varepsilon > 0$, unless $\mathsf{NP} \subseteq \mathsf{DTIME}(2^{polylog(n)})$.*

Further, in the Yes case, there is a vector $y' = (n^{1/p}, x_1, x_2, \ldots, x_n)^{\otimes k}$ where $x_i = \pm 1$ (for $i = 1, 2, \ldots n$) such that $\|M'y'\|_p \geq \tau_C$, where τ_C is the completeness in Theorem 5.21.

We now establish some structure about the tensored instance, which we will use for the hardness of $q \mapsto p$ norm. Let the entries in vector y' be indexed by k-tuple $I = (i_1, i_2, \ldots, i_k)$ where $i_k \in \{0, 1, \ldots, n\}$. It is easy to see that $y'_I = \pm n^{w(I)/p}$ where $w(I)$ is the number of 0s in tuple I.

Let us introduce variables $x_I = n^{-w(I)/p} y_I$ where $w(I) =$ number of 0s in tuple I. It is easy to observe that there is a matrix B such that

$$\frac{\|M'y\|_p}{\|y\|_p} = \frac{\|B\mathbf{x}\|_p}{\sum_I n^{w(I)} |x_I|^p} = g'(x)$$

Further, it can also be seen that in the Yes case, there is a ± 1 assignment for x_I which attains the value $g'(x) = \tau_C$.

5.5.3 Approximating $\|A\|_{q\mapsto p}$ when $p \neq q$.

Let us now consider the case $p \neq q$, more specifically, we have $2 < p < q$, and we wish to prove a hardness of approximation result similar to the theorem above. The idea is to use the same instance as in the case $p \mapsto p$. However as we mentioned earlier, the hardness amplification step using tensor products does not when $q \neq p$ (in particular, it is not true that $q \mapsto p$ norms multiply under tensor products).

However, we show that in our case, the instances are special – in particular, if the matrices we begin with have a certain structure, then the norm of the tensor product is indeed equal to the product of the norms. We show that the kind of instances we deal with indeed have this property, and thus can achieve hardness amplification.

Again, we will first prove that there exists a small constant beyond which we cannot approximate. Let us start with the following maximization problem (which is very similar to Eqn.(5.16))

$$g(x_0, x_1, \dots, x_n) = \frac{\Big(\sum_{i\sim j} |x_i - x_j|^p + Cd \cdot \big(\sum_i t(x_i)\big)\Big)^{1/p}}{\big(n|x_0|^q + \sum_i |x_i|^q\big)^{1/q}}, \tag{5.18}$$

where $t(x_i)$, as earlier, is $|x_0+x_i|^p+|x_0-x_i|^p$. Notice that x_0 is now 'scaled differently' than in Eq.(5.16). This is crucial. Now, in the YES case, we have

$$\max_{\mathbf{x}} \; g(\mathbf{x}) \geq \frac{\big(\rho'(nd/2) \cdot 2^p + Cnd \cdot 2^p\big)^{1/p}}{(2n)^{1/q}}.$$

Indeed, there exists a ± 1 solution which has value at least the RHS. Let us write $\mathcal{N}$ for the numerator of Eq.(5.18). Then

$$g(\mathbf{x}) = \frac{\mathcal{N}}{\big(n|x_0|^p + \sum_i |x_i|^p\big)^{1/p}} \times \frac{\big(n|x_0|^p + \sum_i |x_i|^p\big)^{1/p}}{\big(n|x_0|^q + \sum_i |x_i|^q\big)^{1/q}}.$$

Suppose we started with a No instance. The proof of the $q = p$ case implies that the first term in this product is at most (to a $(1+\varepsilon)$ factor) $\frac{\left(\rho(nd/2)\cdot 2^p + Cnd\cdot 2^p\right)^{1/p}}{(2n)^{1/p}}$.

Now, we note that the second term is at most $(2n)^{1/p}/(2n)^{1/q}$. This follows because for any vector $y \in \mathbb{R}^n$, we have $\|y\|_p/\|y\|_q \le n^{(1/p)-(1/q)}$. We can use this with the $2n$-dimensional vector $(x_0, \ldots, x_0, x_1, x_2, \ldots, x_n)$ to see the desired claim.

From this it follows that in the No case, the optimum is at most (upto a $(1+\varepsilon)$ factor), $\left(\rho(nd/2)\cdot 2^p + Cnd \cdot 2^p\right)^{1/p}(2n)^{-1/q}$. This proves that there exists an $\alpha > 1$ s.t. it is NP-hard to approximate $\|A\|_{q\mapsto p}$ to a factor better than α.

A key property we used in the above argument is that in the Yes case, there exists a ± 1 solution for the x_i $(i \ge 0)$ which has a large value. It turns out that this is the only property we need. More precisely, suppose A is an $n \times n$ matrix, let α_i be positive integers (we will actually use the fact that they are integers, though it is not critical). Now consider the optimization problem $\max_{\mathbf{y}\in\mathbb{R}^n} g(\mathbf{y})$, with

$$g(\mathbf{y}) = \frac{\|A\mathbf{y}\|_p}{(\sum_i \alpha_i |y_i|^p)^{1/p}} \tag{5.19}$$

In the previous section, we established the following claim from the proof of Theorem 5.21.

Lemma 5.22. *For any constant $\gamma > 1$, there exist thresholds τ_C and τ_S with $\tau_C/\tau_S > \gamma$, such that it is NP-hard to distinguish between:*

> Yes *case. There exists a ± 1 assignment to y_i in (5.19) with value at least τ_C, and*
>
> No *case. For all $\mathbf{y} \in \mathbb{R}^n$, $g(\mathbf{y}) \le \tau_S$.*

Proof. Follows from the structure of the product instance.

Using techniques outlined above, we can now show that Claim 5.22 implies the desired result.

Theorem 5.23. *It is NP-hard to approximate $\|A\|_{q\mapsto p}$ to any fixed constant γ for $q \geq p > 2$ and hard to approximate within a factor of $\Omega(2^{(\log n)^{1-\varepsilon}})$ for any constant $\varepsilon > 0$, assuming $\mathsf{NP} \notin \mathsf{DTIME}(2^{polylog(n)})$.*

Proof. As in previous proof (Eq.(5.18)), consider the optimization problem $\max_{\mathbf{y}\in\mathbb{R}^n} h(\mathbf{y})$, with

$$h(\mathbf{y}) = \frac{\|A\mathbf{y}\|_p}{(\sum_i \alpha_i |y_i|^q)^{1/q}} \tag{5.20}$$

By definition,

$$h(\mathbf{y}) = g(\mathbf{y}) \cdot \frac{(\sum_i \alpha_i |y_i|^p)^{1/p}}{(\sum_i \alpha_i |y_i|^q)^{1/q}}. \tag{5.21}$$

Completeness. Consider the value of $h(\mathbf{y})$ for A, α_i in the Yes case for Claim 5.22. Let $\mathbf{y}$ be a ± 1 solution with $g(\mathbf{y}) \geq \tau_C$. Because the y_i are ± 1, it follows that

$$h(\mathbf{y}) \geq \tau_C \cdot \big(\sum_i \alpha_i\big)^{(1/p)-(1/q)}.$$

Soundness. Now suppose we start with an A, α_i in the No case for Claim 5.22.

First, note that the second term in Eq.(5.21) is at most $\big(\sum_i \alpha_i\big)^{(1/p)-(1/q)}$. To see this, we note that α_i are positive integers. Thus by considering the vector $(y_1, \ldots, y_1, y_2, \ldots, y_2, \ldots)$, (where y_i is duplicated α_i times), and using $\|u\|_p/\|u\|_q \leq d^{(1/p)-(1/q)}$ for $u \in \mathbb{R}^d$, we get the desired inequality.

This gives that for all $\mathbf{y} \in \mathbb{R}^n$,

$$h(\mathbf{y}) \leq g(\mathbf{y}) \cdot \big(\sum_i \alpha_i\big)^{(1/p)-(1/q)} \leq \tau_S \cdot \big(\sum_i \alpha_i\big)^{(1/p)-(1/q)}.$$

This proves that we cannot approximate $h(\mathbf{y})$ to a factor better than τ_C/τ_S, which can be made an arbitrarily large constant by Claim 5.22. This finishes the proof, because the optimization problem $\max_{\mathbf{y}\in\mathbb{R}^n} h(\mathbf{y})$ can be formulated as a $q \mapsto p$ norm computation for an appropriate matrix as earlier. $\square$

Note that this hardness instance is not obtained by tensoring the $q \mapsto p$ norm hardness instance. It is instead obtained by considering the $\|A\|_p$ hardness instance and transforming it suitably.

Approximating $\|A\|_{\infty \mapsto p}$ The problem of computing the $\infty \mapsto p$ norm of a matrix A turns out to have a very simple alternative formulation in terms of column vectors of the matrix A: given vectors $\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}$, find the $\max_{\mathbf{x} \in \{-1,1\}^n} \|\sum_i x_i \mathbf{a_i}\|_p$ (longest vector in the ℓ_p norm [2]). As mentioned earlier there is a constant factor approximation for $1 \le p \le 2$ using [61]. However, for the other norms ($p > 2$), using similar techniques we can show

Theorem 5.24. *It is NP-hard to approximate $\|A\|_{\infty \mapsto p}$ to any constant γ for $p > 2$ and hard to approximate within a factor of $\Omega(2^{(\log n)^{1-\varepsilon}})$ for any constant $\varepsilon > 0$, assuming* $\mathsf{NP} \notin \mathsf{DTIME}(2^{polylog(n)})$.

5.6 Hypercontractivity

Both the algorithmic and hardness results of the earlier sections have applied to computing $\|A\|_{q \mapsto p}$ when $p \le q$. Somewhat surprisingly, neither of these extend to the case $p > q$, in which case the norm measures an important property of the matrix called *hypercontractivity.* This notion plays a crucial role in many applications in Mathematics and Computer Science. An operator (or a matrix, A) is said to be q, p hypercontractive if we have $\|Ax\|_p \le \|x\|_q$ for some $q \le p$.

Proving certain operators to be hypercontractive is a crucial step in applications as diverse as the theory of Markov Chains [22], measure concentration [31], hardness of approximation (the so-called Beckner-Bonami inequalities [16]), Gaussian pro-

[2]Note that despite being similar sounding, this is in no way related to the well-studied Shortest Vector Problem [54] for lattices, which has received a lot of attention in the cryptography community [68]. SVP asks for *minimizing* the same objective as defined here, but with $x_i \in \mathbb{Z}$.

cesses [46], and many more. A recent survey by Punyashloka Biswal [21] considers some Computer Science applications in detail.

We will mention and discuss a few applications which arose recently, and which help further motivate the study of the approximability of matrix norm questions.

5.6.1 Certifying "Restricted Isometry"

We have introduced $q \mapsto p$ norms of matrices as extensions of the singular value of a matrix. Apart from being a natural extension to ℓ_p spaces, are there applications in which being able to compute it for different q, p are important? In this section, we will see an application in which we will need to certify a certain matrix property at different "scales", and the choice of p, q we use will depend crucially on the scale. We note that this application is folklore in the Compressed Sensing community.[3]

A notion which was recently studied, particularly in compressed sensing is that of "RIP" matrices, or matrices which have the Restricted Isometry Property (RIP). A linear operator, given by a matrix A $(m \times n)$ is said to be an *isometry* for a vector x if $\|Ax\|_2 = \|x\|_2$. It is said to be an *almost isometry* if we have $\|x\|_2 \le \|Ax\|_2 \le 10\|x\|_2$ (the choice of constant here is arbitrary). Now, we say that a matrix has the RIP property if it is an almost isometry, when restricted to *sparse* vectors. More formally,

Definition 5.25. *We say a matrix A satisfies the Restricted Isometry Property w.r.t. the sparsity parameter k iff*

$$\|x\|_2 \le \|Ax\|_2 \le 10\|x\|_2 \quad \forall x \ : \ \|x\|_0 \le k.$$

Here $\|x\|_0$ denotes the size of the support of the vector x.

A well-studied problem in compressed sensing (see the blog post by Tao [73]) is to give explicit constructions (or algorithms which use very little randomness) of

[3] I would like to thank Edo Liberty for pointing out this connection.

matrices A which satisfy the RIP property for a certain sparsity parameter k (typically $\ll n$). It is desired to use a "number of measurements" i.e., the value of m as small as possible.

It is easy to prove that a *random* matrix A has the RIP property. In particular, using a standard Chernoff bound argument, we can show:

Lemma 5.26. *Let m, n, k be parameters, with $m \leq n$ and $m \geq Ck \log(n/k)$. Let A be an $m \times n$ matrix with each entry drawn i.i.d. from a standard Gaussian $N(0,1)$. Then with high probability (at least $1 - \frac{1}{n^2}$), we have:*

$$\frac{m}{10} \cdot \|x\|_2^2 \leq \|Ax\|_2^2 \leq 10m \cdot \|x\|_2^2 \;\; \forall x \text{ s.t. } \|x\|_0 \leq k. \tag{5.22}$$

Since in practice it suffices to find one matrix with the given property, it is useful to have an algorithm which "checks" if a given matrix has the RIP property. More weakly, we could ask for such a certification algorithm which works w.h.p. for random matrices.

We will now see that being able to compute hypercontractive norms can help certify the RIP property (at least in *one direction*). To be concrete, let us fix parameters. Let n be an integer large enough, and let k be a small power of n (i.e., $k = n^\gamma$ for some $0 < \gamma < 1$).

Suppose we have an algorithm to compute $\|A\|_{q \mapsto 2}$, for some $q \leq 2$. Denote by q' the *dual* of q, i.e., $1/q + 1/q' = 1$. Now suppose $\|A\|_{q \mapsto 2} = \lambda$. Then for all x s.t. $\|x\|_0 = k$, we have

$$\lambda \geq \frac{\|Ax\|_2}{\|x\|_q} \geq \frac{\|Ax\|_2}{\|x\|_2} \cdot \frac{\|x\|_2}{\|x\|_q} \geq \frac{\|Ax\|_2}{\|x\|_2} \cdot \frac{1}{k^{\frac{1}{q} - \frac{1}{2}}}.$$

(Note that in the last step, we used Hölder's inequality). Thus in terms of q', we can say that for all k-sparse x, we have

$$\frac{\|Ax\|_2^2}{\|x\|_2^2} \le \lambda^2 k^{1-\frac{2}{q'}}.$$

If we can compute λ efficiently for large enough q' (i.e., q close enough to 1), let us see how we can certify that a random A satisfies the upper bound in Eq.(5.22). For a random A, we have the following (this can again be verified by first noting that $\|A\|_{q\mapsto 2} = \|A^T\|_{2\mapsto q'}$, and a simple Chernoff bound):

Lemma 5.27. *Let A be an $m \times n$ matrix with $m < n$, and suppose the entries of A are picked i.i.d. from $N(0,1)$. Then we have*

$$\|A\|_{q\mapsto 2} = \|A^T\|_{2\mapsto q'} \le n^{1/q'},$$

w.p. at least $1 - 1/n^2$.

Now since we assumed we could compute λ for random A (even up to a constant, say), we can certify, from the above, that for any k-sparse x,

$$\frac{\|Ax\|_2^2}{\|x\|_2^2} \le n^{\frac{2}{q'}} k^{1-\frac{2}{q'}}.$$

For q' large enough, this is a fairly good bound, because it implies that for $m \ge k\left(\frac{n}{k}\right)^{2/q'}$, we can certify that an $m \times n$ random matrix satisfies the RIP property.

Special cases. Are there certain ranges of parameters in which we can compute $\|A\|_{q\mapsto 2}$ efficiently for random A? It turns out there are, and we will give one example. We consider the case of bounding the $4/3 \mapsto 2$ norm, for a random matrix A with m rows and n columns, with $m < \sqrt{n}$. Note that this unfortunately does not help us certify the RIP property for any interesting range of k.

In this case, we can proceed as follows: first note that by duality of norms (4.1), it suffices to consider the question of bounding the $2 \mapsto 4$ norm of a random matrix A with n rows and m columns and $m < \sqrt{n}$. More precisely, we wish to show that for such a matrix, we have

$$\|Ax\|_4^4 \leq O(n) \cdot \|x\|_2^4 \qquad \forall x \in \Re^m.$$

It turns out that for $m < \sqrt{n}$, we can do this by "relaxing" the question to one of computing the spectral norm (and we do not lose much in this process). More formally, we note that (recall A_i refers to the ith row of the matrix A)

$$\|Ax\|_4^4 = \sum_i \langle A_i, x\rangle^4 = \sum_i \langle A_i \otimes A_i, x \otimes x\rangle^2.$$

Now consider a matrix B which is $n \times m^2$, and has $B_i := A_i \otimes A_i$ (treating the tensor product as an m^2 dimensional vector). From the above, we have that

$$\max_{\|x\|_2^2=1} \|Ax\|_4^4 \leq \max_{\|x\|_2^2=1} \|B(x \otimes x)\|_2^2 \leq \max_{\|z\|_2^2=1} \|Bz\|_2^2 \leq \|B\|_{2\mapsto 2}^2.$$

However for $m < \sqrt{n}$, the matrix B is still rectangular with more rows than columns, and i.i.d. rows. Thus we can hope to use methods from random matrix theory to bound its singular values. The problem though is that the *entries* of the matrix are not i.i.d. anymore.

However, we can use the "non-isotropic rows" version (Theorem 5.44 of [76]) of the spectral norm bound to obtain that $\|B\|_{2\mapsto 2}^2 \leq O(n)$. The details of this are rather straightforward, so we will not get into them. This gives the desired bound on $\|A\|_{2\mapsto 4}$.

We note that in this case it is also possible to prove the lower bound of (5.22) using properties of the matrix B, since the number of rows is larger than the number of columns. It will be interesting to see if strengthening of these ideas can help cerfity the RIP property for some interesting values of k.

Finally, we note that this range of parameters is also considered in the recent work of Barak *et al.* [14]. They obtain a somewhat sharper bound (in particular, the precise constant in the $O(n)$ term above) for the norm using a relaxation they call the *tensor-SDP*. Their ideas are very similar in spirit to the above discussion.

5.6.2 Relations to the expansion of small sets

The recent work of Barak *et al.* [14] showed that computing $\|A\|_{2\mapsto 4}$ efficiently would imply an approximation algorithm for small-set expansion. In particular, such an algorithm could be used to find a *sparse* (defined slightly differently) vector in the span of a bunch of vectors, which turns out to be related to SSE. We will not go into the details.

5.6.3 Robust expansion and locality sensitive hashing

A final application we mention is a recent result of Panigrahy, Talwar and Wieder [63] on lower bounds for Nearest Neighbor Search (NNS). Their main idea is to relate (approximate) NNS in a metric space to a certain expansion parameter of the space, called "robust expansion". This is a more fine-grained notion of expansion (than the conductance, or even the spectral profile). Formally, it is defined by two parameters:

Definition 5.28. *A graph $G = (V, E)$ has (δ, ρ) robust expansion at least Φ, if for all sets $S \subseteq V$ of size at most δ, and sets $T \subseteq V$ s.t. $E(S,T) \geq \rho \cdot E(S,V)$, we have $|T|/|S| \geq \Phi$. That is, for sets S of size at most δn, no set smaller than $\Phi|S|$ can* capture *a ρ fraction of the edges out of S. (This can be seen as a* robust *version of vertex expansion for small sets)*

[63] then show space lower bounds for randomized algorithms for metric-NNS in terms of the robust expansion of a graph defined using the metric (for appropriate δ, ρ). The moral here is that good robust expansion implies good lower bounds on the size of the data structure. While it seems that approximating the robust expansion of a general graph is a very hard question (it is related, for instance, to DkS and small-set expansion), it is possible to obtain bounds for specific graphs (such as those obtained from trying to prove lower bounds for ℓ_1 and ℓ_∞ metrics in their framework).

The main tool used for this purpose is a hypercontractive inequality for the adjacency matrix of the graph. Roughly speaking, if we have a good upper bound on $\|A\|_{q\mapsto p}$ for appropriate q, p, it is possible to show that a small set T cannot *capture* a good fraction of edges out of a set S. This example illustrates that being able to approximate hypercontractive norms (or show hardness thereof) is an important question even for matrices with *all positive entries.*

Open Problem 5.29. *Can we compute $\|A\|_{q\mapsto p}$ for $p > q$, for non-negative matrices A?*

Chapter 6

Maximum Density Subgraph and Generalizations

In this chapter, will discuss the QP-Ratio problem introduced in Section 4.3. As we mentioned earlier, this is a generalization of the maximum density subgraph problem in graphs, to matrices which could potentially have negative entries.

We start by considering continuous relaxations for the problem, and obtain an $\widetilde{O}(n^{1/3})$ approximation. We will then see certain natural special cases in which we can obtain a better approximation ratio. Then, we will move to showing hardness of approximation results. As discussed in Section 4.3.1, we do not know how to prove strong inapproximability results based on "standard" assumptions such as $P \neq NP$ (the best we show is an APX-hardness). We thus give evidence for hardness based on the random k-AND conjecture and a ratio version of the unique games conjecture.

The analysis of the algorithm will make clear the difficulty in capturing the $x_i \in \{-1, 0, 1\}$ constraint using convex relaxations.

6.1 Algorithms for QP-Ratio

Let us first recall the definition of QP-Ratio. Given an $n \times n$ matrix A with zero diagonal, the QP-Ratio objective is defined as

$$\textsf{QP-Ratio}: \quad \max_{\{-1,0,1\}^n} \frac{\sum_{i \neq j} a_{ij} x_i x_j}{\sum x_i^2} \tag{6.1}$$

Our algorithms for the problem will involve trying to come up with convex relaxations for the problem.

6.1.1 A first cut: the eigenvalue relaxation

We start with the most natural relaxation for QP-Ratio (4.2):

$$\max \frac{\sum_{i,j} A_{ij} x_i x_j}{\sum_i x_i^2} \text{ subject to } x_i \in [-1, 1]$$

(instead of $\{0, \pm 1\}$). The solution to this is precisely the largest eigenvector of A, scaled such that entries are in $[-1, 1]$. Thus the optimum solution to the relaxation can be computed efficiently.

However it is easy to construct instances for which this relaxation is *bad*. Let A be the adjacency matrix of an $(n+1)$ vertex star (with v_0 as the center of the star). The optimum value of the QP-Ratio obective in this case is $O(1)$, because if we set k of the x_i non-zero, we cannot obtain a numerator value $> k$.

The relaxation however, can cheat by setting $x_0 = \frac{1}{2}$ and $x_i = \frac{1}{\sqrt{2n}}$ for $i \in [n]$. This solution achieves an objective value of $\Omega(\sqrt{n})$. Thus the relaxation has a gap of $\Omega(\sqrt{n})$.

Note that the main reason for the integrality gap is because the fractional solution involves x_i of very different magnitudes.

6.1.2 Adding SDP constraints and an improved algorithm

Thus the natural question is, can we write a Semidefinite Program (SDP) which can capture the problem better? We prove that the answer is yes, to a certain extent. Consider the following relaxation:

$$\max \sum_{i,j} A_{ij}\langle \mathbf{u}_i, \mathbf{u}_j\rangle \text{ subject to}$$
$$\sum_i \mathbf{u}_i^2 = 1, \text{ and}$$
$$|\langle \mathbf{u}_i, \mathbf{u}_j\rangle| \le \mathbf{u}_i^2 \text{ for all } i,j \tag{6.2}$$

It is easy to see that this is indeed a relaxation: start with an integer solution $\{x_i\}$ with k non-zero x_i, and set $\mathbf{u}_i = (x_i/\sqrt{k})\cdot \mathbf{u}_0$ for a fixed unit vector $\mathbf{u}_0$.

Without constraint (6.2), the SDP relaxation is equivalent to the eigenvalue relaxation given above. Roughly speaking, equation (6.2) tries to impose the constraint that non-zero vectors are of equal length. This is because if $\mathbf{u}_i \gg \mathbf{u}_j$, then $\langle \mathbf{u}_i, \mathbf{u}_j\rangle$ should be smaller than $\mathbf{u}_j^2$, which is $\ll \|\mathbf{u}_i\|\|\mathbf{u}_j\|$ (which is what Cauchy-Schwarz automatically gives).

Indeed, in the example of the $(n+1)$-vertex star, this relaxation has value equal to the true optimum. In fact, for any instance with $A_{ij} \ge 0$ for all i,j (This follows from observing that the relaxation is strictly stronger than an LP relaxation used in [26], which itself is exact).

There are other natural relaxations one can write by viewing the $\{0, \pm 1\}$ requirement like a 3-alphabet CSP. We consider one of these in section 6.2.1, and show an $\Omega(n^{1/2})$ integrality gap. It is interesting to see if lift and project methods starting with this relaxation can be useful.

An $\widetilde{O}(n^{1/3})$ rounding algorithm. We will now see how we can obtain an algorithm which shows that the SDP is indeed stronger than the eigenvalue relaxation we saw earlier. We consider an instance of QP-Ratio defined by A $(n \times n)$. Let $\mathbf{u}_i$ be an optimal solution to the SDP, and let the objective value be denoted sdp. The algorithm will round the $\mathbf{u}_i$ into $\{0, \pm 1\}$.

Outline of the algorithm. The algorithm can be thought of as having two *phases*. In the first, we will move to a solution in which all the vectors are either of equal length or 0 (this is the "vector equivalent" of variables being $\{0, \pm 1\}$). Then we will see how to round this solution to a $\{0, \pm 1\}$ solution. The lossy step (in terms of the approximation ratio) is the first – here we prove that the loss is at most $\widetilde{O}(n^{1/3})$, and in the second step we use a standard algorithm for quadratic programming ([61, 29]). This gives an approximation ratio of $\widetilde{O}(n^{1/3})$ overall.

We will sometimes be sloppy w.r.t. logarithmic factors in the analysis. Since the problem is the same up to scaling the A_{ij}, let us assume that $\max_{i,j} |A_{ij}| = 1$. There is a trivial solution which attains a value $1/2$ (if i, j are indices with $|A_{ij}| = 1$, set x_i, x_j to be ± 1 appropriately, and the rest of the x's to 0). Now, since we are aiming for an $\widetilde{O}(n^{1/3})$ approximation, we can assume that $\mathsf{sdp} > n^{1/3}$.

As we stated in the algorithm outline, the *difficulty* is when most of the contribution to sdp is from non-zero vectors with very different lengths. The idea of the algorithm will be to move to a situation in which this does not happen. First, we show that if the vectors indeed have roughly equal length, we can round well. Roughly speaking, the algorithm uses the lengths $\|\mathbf{v}_i\|_2$ to determine whether to pick i, and then uses the ideas of [29] (or the earlier works of [60, 59]) applied to the vectors $\frac{\mathbf{v}_i}{\|\mathbf{v}_i\|_2}$.

Lemma 6.1. *Given a vector solution $\{\mathbf{v}_i\}$, with $\mathbf{v}_i^2 \in [\tau/\Delta, \tau]$ for some $\tau > 0$ and $\Delta > 1$, we can round it to obtain an integer solution with cost at least* $\mathsf{sdp}/(\sqrt{\Delta} \log n)$.

Proof. Starting with $\mathbf{v}_i$, we produce vectors $\mathbf{w}_i$ each of which is either 0 or a unit vector, such that

$$\text{If} \quad \frac{\sum_{i,j} A_{ij}\langle \mathbf{v}_i, \mathbf{v}_j\rangle}{\sum_i \mathbf{v}_i^2} = \mathsf{sdp}, \text{ then } \frac{\sum_{i,j} A_{ij}\langle \mathbf{w}_i, \mathbf{w}_j\rangle}{\sum_i \mathbf{w}_i^2} \geq \frac{\mathsf{sdp}}{\sqrt{\Delta}}.$$

Stated this way, we are free to re-scale the $\mathbf{v}_i$, thus we may assume $\tau = 1$. Now note that once we have such $\mathbf{w}_i$, we can throw away the zero vectors and apply the rounding algorithm of [29] (with a loss of an $O(\log n)$ approximation factor), to obtain a $0, \pm 1$ solution with value at least $\mathsf{sdp}/(\sqrt{\Delta}\log n)$.

So it suffices to show how to obtain the $\mathbf{w}_i$. Let us set (recall we assumed $\tau = 1$)

$$\mathbf{w}_i = \begin{cases} \mathbf{v}_i/\|\mathbf{v}_i\|_2, \text{ with prob. } \|\mathbf{v}_i\|_2 \\ 0 \text{ otherwise} \end{cases}$$

(this is done independently for each i). Note that the probability of picking i is proportional to the length of $\mathbf{v}_i$ (as opposed to the typically used square lengths, [28] say). Since $A_{ii} = 0$, we have

$$\frac{\mathbb{E}\left[\sum_{i,j} A_{ij}\langle \mathbf{w}_i, \mathbf{w}_j\rangle\right]}{\mathbb{E}\left[\sum_i \mathbf{w}_i^2\right]} = \frac{\sum_{i,j} A_{ij}\langle \mathbf{v}_i, \mathbf{v}_j\rangle}{\sum_i |\mathbf{v}_i|} \geq \frac{\sum_{i,j} A_{ij}\langle \mathbf{v}_i, \mathbf{v}_j\rangle}{\sqrt{\Delta}\sum_i \mathbf{v}_i^2} = \frac{\mathsf{sdp}}{\sqrt{\Delta}}. \qquad (6.3)$$

The above proof only shows the existence of vectors $\mathbf{w}_i$ which satisfy the bound on the ratio. The proof can be made constructive using the method of conditional expectations. In particular, we set variables one by one, i.e. we first decide whether to make $\mathbf{w}_1$ to be a unit vector along it or the 0 vector, depending on which choice maintains the ratio to be $\geq \theta = \frac{\mathsf{sdp}}{\sqrt{\Delta}}$. Now, after fixing $\mathbf{w}_1$, we fix $\mathbf{w}_2$ similarly etc., while always maintaining the invariant that the ratio $\geq \theta$.

At step i, let us assume that $\mathbf{w}_1, \ldots, \mathbf{w}_{i-1}$ have already been set to either unit vectors or zero vectors. Consider $\mathbf{v}_i$ and let $\tilde{\mathbf{v}}_i = \mathbf{v}_i/\|\mathbf{v}_i\|_2$. $\mathbf{w}_i = \tilde{\mathbf{v}}_i$ w.p. $p_i = \|\mathbf{v}_i\|_2$ and 0 w.p $(1-p_i)$.

In the numerator, $B = \mathbb{E}[\sum_{j\neq i, k\neq i} a_{jk}\langle w_j, w_k\rangle]$ is contribution from terms not involving i. Also let $\mathbf{c}_i = \sum_{k\neq i} a_{ik}\mathbf{w}_k$ and let $\mathbf{c}'_i = \sum_{j\neq i} a_{ji}\mathbf{w}_j$. Then, from equation 6.3

$$\theta \leq \frac{\mathbb{E}[\sum_{j,k} a_{jk}\langle \mathbf{w}_j, \mathbf{w}_k\rangle]}{E[\sum_j |\mathbf{w}_j|^2]} = \frac{p_i\big(\langle \tilde{\mathbf{v}}_i, \mathbf{c}_i\rangle + \langle \mathbf{c}'_i, \tilde{\mathbf{v}}_i\rangle + B\big) + (1-p_i)B}{p_i\big(1 + \sum_{j\neq i}\|\mathbf{w}_j\|_2^2\big) + (1-p_i)\big(\sum_{j\neq i}\|\mathbf{w}_j\|_2^2\big)\big)}$$

Hence, by the simple fact that if c, d are positive and $\frac{a+b}{c+d} > \theta$, then either $\frac{a}{c} > \theta$ or $\frac{b}{d} > \theta$, we see that either by setting $\mathbf{w}_i = \tilde{\mathbf{v}}_i$ or $\mathbf{w}_i = 0$, we get value at least θ. □

Let us define the 'value' of a set of vectors $\{\mathbf{u}_i\}$ to be $\mathsf{val} := \frac{\sum A_{ij}\langle \mathbf{u}_i, \mathbf{u}_j\rangle}{\sum_i \mathbf{u}_i^2}$. The $\mathbf{v}_i$ we start will have $\mathsf{val} = \mathsf{sdp}$.

Claim 6.2. *We can move to a set of vectors such that (a)* val *is at least* $\mathsf{sdp}/2$*, (b) each non-zero vector* $\mathbf{v}_i$ *satisfies* $\mathbf{v}_i^2 \geq 1/n$*, (c) vectors satisfy* (6.2)*, and (d)* $\sum_i \mathbf{v}_i^2 \leq 2$*.*

The proof is by showing that very small vectors can either be enlarged or thrown away

Proof. Suppose $0 < \mathbf{v}_i^2 < 1/n$ for some i. If $S_i = \sum_j A_{ij}\mathbf{v}_i \cdot \mathbf{v}_j \leq 0$, we can set $\mathbf{v}_i = 0$ and improve the solution. Now if $S_i > 0$, replace $\mathbf{v}_i$ by $\frac{1}{\sqrt{n}} \cdot \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|_2}$ (this only increases the value of $\sum_{i,j} A_{ij}\langle \mathbf{v}_i, \mathbf{v}_j\rangle$), and repeat this operation as long as there are vectors with $\mathbf{v}_i^2 < 1/n$. Overall, we would only have increased the value of $\sum_{i,j} A_{ij}\mathbf{v}_i \cdot \mathbf{v}_j$, and we still have $\sum_i \mathbf{v}_i^2 \leq 2$. Further, it is easy to check that $|\langle \mathbf{v}_i, \mathbf{v}_j\rangle| \leq \mathbf{v}_i^2$ also holds in the new solution (though it might not hold in some intermediate step above). □

The next lemma also gives an upper bound on the lengths – this is where the constraints (6.2) are crucial. It uses equation 6.2 to upper bound the contribution from each vector – hence large vectors can not contribute much in total, since they are few in number.

Lemma 6.3. *Suppose we have a solution of value Bn^ρ and $\sum_i \mathbf{v}_i^2 \le 2$. We can move to a solution with value at least $Bn^\rho/2$, and $\mathbf{v}_i^2 < 16/n^\rho$ for all i.*

Proof. Let $\mathbf{v}_i^2 > 16/n^\rho$ for some index i. Since $|\langle \mathbf{v}_i, \mathbf{v}_j\rangle| \le \mathbf{v}_j^2$, we have that for each such i,

$$\sum_j A_{ij}\langle \mathbf{v}_i, \mathbf{v}_j\rangle \le B\sum_j \mathbf{v}_j^2 \le 2B$$

Thus the contribution of such i to the sum $\sum_{i,j} A_{ij}\langle \mathbf{v}_i, \mathbf{v}_j\rangle$ can be bounded by $m \times 4B$, where m is the number of indices i with $\mathbf{v}_i^2 > 16/n^\rho$. Since the sum of squares is ≤ 2, we must have $m \le n^\rho/8$, and thus the contribution above is at most $Bn^\rho/2$. Thus the rest of the vectors have a contribution at least $\mathsf{sdp}/2$ (and they have sum of squared-lengths ≤ 2 since we picked only a subset of the vectors) □

Theorem 6.4. *Suppose A is an $n \times n$ matrix with zero's on the diagonal. Then there exists a polynomial time $\widetilde{O}(n^{1/3})$ approximation algorithm for the* QP-Ratio *problem defined by A.*

Proof. As before, let us rescale and assume $\max i, j|A_{ij}| = 1$. Now if $\rho > 1/3$, Lemmas 6.2 and 6.3 allow us to restrict to vectors satisfying $1/n \le \mathbf{v}_i^2 \le 4/n^\rho$, and using Lemma 6.1 gives the desired $\widetilde{O}(n^{1/3})$ approximation; if $\rho < 1/3$, then the trivial solution of $1/2$ is an $\widetilde{O}(n^{1/3})$ approximation. □

6.1.3 Special case: A is bipartite

In this section, we prove the following theorem:

Theorem 6.5. *When A is bipartite (i.e. the adjacency matrix of a weighted bipartite graph), there is a (tight upto logarithmic factor) $O(n^{1/4}\log^2 n)$ approximation algorithm for* QP-Ratio *.*

Bipartite instances of QP-Ratio can be seen as the ratio analog of the Grothendieck problem [6]. The algorithm works by rounding the semidefinite program relaxation from section 6.1. As before, let us assume $\max_{i,j} a_{ij} = 1$ and consider a solution to the SDP (6.2). To simplify the notation, let u_i and v_j denote the vectors on the two sides of the bipartition. Suppose the solution satisfies:

$$(1) \sum_{(i,j)\in E} a_{ij}\langle u_i, v_j\rangle \geq n^\alpha, \qquad (2) \sum_i u_i^2 = \sum_j v_j^2 = 1.$$

If the second condition does not hold, we scale up the vectors on the smaller side, losing at most a factor 2. Further, we can assume from Lemma 6.2 that the squared lengths u_i^2, v_j^2 are between $\frac{1}{2n}$ and 1. Let us divide the vectors $\{u_i\}$ and $\{v_j\}$ into $\log n$ groups based on their squared length. There must exist two levels (for the u and v's respectively) whose contribution to the objective is at least $n^\alpha/\log^2 n$.[1] Let L denote the set of indices corresponding to these u_i, and R denote the same for v_j. Thus we have $\sum_{i\in L, j\in R} a_{ij}\langle u_i, v_j\rangle \geq n^\alpha/\log^2 n$. We may assume, by symmetry that $|L| \leq |R|$. Now since $\sum_j v_j^2 \leq 1$, we have that $v_j^2 \leq 1/|R|$ for all $j \in R$. Also, let us denote by A_j the $|L|$-dimensional vector consisting of the values a_{ij}, $i \in L$. Thus

$$\frac{n^\alpha}{\log^2 n} \leq \sum_{i\in L, j\in R} a_{ij}\langle u_i, v_j\rangle \leq \sum_{i\in L, j\in R} |a_{ij}| \cdot v_j^2 \leq \frac{1}{|R|}\sum_{j\in R}\|A_j\|_1. \tag{6.4}$$

We will construct an assignment $x_i \in \{+1, -1\}$ for $i \in L$ such that $\frac{1}{|R|} \cdot \sum_{j\in R} \big|\sum_{i\in L} a_{ij}x_i\big|$ is 'large'. This suffices, because we can set $y_j \in \{+1, -1\}$, $j \in R$ appropriately to obtain the value above for the objective (this is where it is crucial that the instance is bipartite – there is no contribution due to other y_j's while setting one of them).

Lemma 6.6. *There exists an assignment of $\{+1, -1\}$ to the x_i such that*

$$\sum_{j\in R} \big|\sum_{i\in L} a_{ij}x_i\big| \geq \frac{1}{24}\sum_{j\in R}\|A_j\|_2$$

[1]Such a clean division into levels can only be done in the bipartite case – in general there could be negative contribution from 'within' the level.

Furthermore, such an assignment can be found in polynomial time.

Proof. The intuition is the following: suppose $X_i, i \in L$ are i.i.d. $\{+1, -1\}$ random variables. For each j, we would expect (by random walk style argument) that $\mathbb{E}\big[\big|\sum_{i \in L} a_{ij} X_i\big|\big] \approx \|A_j\|_2$, and thus by linearity of expectation,

$$\mathbb{E}\Big[\sum_{j \in R} \Big|\sum_{i \in L} a_{ij} X_i\Big|\Big] \approx \sum_{j \in R} \|A_j\|_2.$$

Thus the existence of such x_i follows. This can in fact be formalized using the following lemma (please refer to full version for the proof):

$$\mathbb{E}\big[\big|\sum_{i \in L} a_{ij} X_i\big|\big] \geq \|A_j\|_2/12 \tag{6.5}$$

This equation is seen to be true from the following lemma

Lemma 6.7. *Let $b_1, \ldots, b_n \in \mathbb{R}$ with $\sum_i b_i^2 = 1$, and let $X_1, \ldots, X_n$ be i.i.d. $\{+1, -1\}$ r.v.s. Then*

$$\mathbb{E}[|\sum_i b_i X_i|] \geq 1/12.$$

Proof. Define the r.v. $Z := \sum_i b_i X_i$. Because the X_i are i.i.d. $\{+1, -1\}$, we have $\mathbb{E}[Z^2] = \sum_i b_i^2 = 1$. Further, $\mathbb{E}[Z^4] = \sum_i b_i^4 + 6\sum_{i<j} b_i^2 b_j^2 < 3(\sum_i b_i^2)^2 = 3$. Thus by Paley-Zygmund inequality,

$$\Pr[Z^2 \geq \frac{1}{4}] \geq \frac{9}{16} \cdot \frac{(\mathbb{E}[Z^2])^2}{\mathbb{E}[Z^4]} \geq \frac{3}{16}.$$

Thus $|Z| \geq 1/2$ with probability at least $3/16 > 1/6$, and hence $\mathbb{E}[|Z|] \geq 1/12$. □

□

We also make this lemma constructive as follows. Let r.v. $S := \sum_{j \in R} \big| \sum_{i \in L} a_{ij} X_i \big|$. It is a non-negative random variable, and for every choice of X_i, we have

$$S \le \sum_{j \in R} \sum_{i \in L} |a_{ij}| \le L^{1/2} \sum_{j \in R} \|A_j\|_2 \le n^{1/2} \mathbb{E}[S]$$

Let p denote $\Pr[S < \frac{\mathbb{E}[S]}{2}]$. Then from the above inequality, we have that $(1-p) \ge \frac{1}{2n^{1/2}}$. Thus if we sample the X_i say n times (independently), we *hit* an assignment with a large value of S with high probability. □

Proof of Theorem 6.5. By Lemma 6.6 and Eq (6.4), there exists an assignment to x_i, and a corresponding assignment of $\{+1, -1\}$ to y_j such that the value of the solution is at least

$$\frac{1}{|R|} \cdot \sum_{j \in R} \|A_j\|_2 \ge \frac{1}{|R| \; |L|^{1/2}} \sum_{j \in R} \|A_j\|_1 \ge \frac{n^{\alpha}}{|L|^{1/2} \log^2 n}. \qquad \text{[By Cauchy Schwarz]}$$

Now if $|L| \le n^{1/2}$, we are done because we obtain an approximation ratio of $O(n^{1/4} \log^2 n)$. On the other hand if $|L| > n^{1/2}$ then we must have $\|u_i\|_2^2 \le 1/n^{1/2}$. Since we started with u_i^2 and v_i^2 being at least $1/2n$ (Lemma 6.2) we have that all the squared lengths are within a factor $O(n^{1/2})$ of each other. Thus by Lemma 6.1 we obtain an approximation ratio of $O(n^{1/4} \log n)$. This completes the proof. □

6.1.4 Special case: A is positive semidefinite

The standard quadratic programming problem has a better approximation guarantee (of $2/\pi$, as opposed to $O(\log n)$) when the matrix A is p.s.d. We show that similarly for the QP-Ratio problem, there is a vast difference in the approximation ratios we can obtain. Indeed in this case, it is quite easy to obtain a polylog(n) approximation.

This proceeds as follows: start with a solution to the eigenvalue relaxation (call the value ρ). Since A is psd, the numerator can be seen as $\sum_i (B_i x)^2$, where B_i are linear forms. Now divide the x_i into $O(\log n)$ levels depending on their absolute value (need to show that x_i are not too small – poly in $1/n$, $1/|A|_\infty$). We can now see each term $B_i x_i$ a sum of $O(\log n)$ terms (grouping by level). Call these terms $C_i^1, \dots, C_i^\ell$, where ℓ is the number of levels. The numerator is upper bounded by $\ell(\sum_i \sum_j (C_i^j)^2)$, and thus there is some j such that $\sum_i (C_i^j)^2$ is at least $1/\log^2 n$ times the numerator. Now work with a solution y which sets $y_i = x_i$ if x_i is in the jth level and 0 otherwise. This is a solution to the ratio question with value at least ρ/ℓ^2. Further, each $|y_i|$ is either 0 or in $[\rho, 2\rho]$, for some ρ.

From this we can move to a solution with $|y_i|$ either 0 or 2ρ as follows: focus on the numerator, and consider some $x_i \neq 0$ with $|x_i| < 2\rho$ (strictly). Fixing the other variables, the numerator is a convex function of x_i in the interval $[-2\rho, 2\rho]$ (it is a quadratic function, with non-negative coefficient to the x_i^2 term, since A is psd). Thus there is a choice of $x_i = \pm 2\rho$ which only increases the numerator. Perform this operation until there are no $x_i \neq 0$ with $|x_i| < 2\rho$. This process increases each $|x_i|$ by a factor at most 2. Thus the new solution has a ratio at least half that of the original one. Combining these two steps, we obtain an $O(\log^2 n)$ approximation algorithm.

6.2 Integrality gaps

We will now show integrality gaps for two SDP relaxations. First, we will show a gap of $\Omega(n^{1/4})$ for the relaxation we introduced in Section 6.1.2. Next we will consider a *CSP-like* relaxation which we alluded to earlier, and show a gap of $\Omega(n^{1/2})$ for it.

We begin with the SDP defined in Section 6.1.2 Consider a complete bipartite graph on L, R, with $|L| = n^{1/2}$, and $|R| = n$. The edge weights are set to ± 1 uniformly at random. Denote by B the $n^{1/2} \times n$ matrix of edge weights (rows indexed by L and columns by R). A standard Chernoff bound argument shows

Lemma 6.8. *With high probability over the choice of B, we have* $\mathsf{opt} \leq \sqrt{\log n} \cdot n^{1/4}$.

Proof. Let $S_1 \subseteq L$, $S_2 \subseteq R$ be of sizes a, b respectively. Consider a solution in which these are the only variables assigned non-zero values (thus we fix some ± 1 values to these variables). Let val denote the value of the numerator. By the Chernoff bound, we have

$$\Pr[\mathsf{val} \geq c\sqrt{ab}] \leq e^{-c^2/3},$$

for any $c > 0$. Now choosing $c = 10\sqrt{(a+b)\log n}$, and taking union bound over all choices for S_1, S_2 and the assignment (there are $\binom{\sqrt{n}}{a}\binom{n}{b}2^{a+b}$ choices overall), we get that w.p. at least $1 - 1/n^3$, no assignment with this choice of a and b gives val bigger than $\sqrt{ab(a+b)\log n}$. The ratio in this case is at most $\sqrt{\log n \cdot \frac{ab}{a+b}} \leq \sqrt{\log n} \cdot n^{1/4}$. Now we can take union bound over all possible a and b, thus proving that $\mathsf{opt} \leq n^{1/4}$ w.p. at least $1 - 1/n$. $\square$

Let us now exhibit an SDP solution with value $n^{1/2}$. Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{\sqrt{n}}$ be mutually orthogonal vectors, with each $\mathbf{v}_i^2 = 1/2n^{1/2}$. We assign these vectors to vertices in L. Now to the jth vertex in R, assign the vector $\mathbf{u}_j$ defined by

$$\mathbf{u}_j = \sum_i B_{ij} \frac{\mathbf{v}_i}{\sqrt{n}}.$$

It is easy to check that $\mathbf{u}_j^2 = \sum_i \frac{\mathbf{v}_i^2}{n} = \frac{1}{2n}$. Further, note that for any i, j, we have (since all $\mathbf{v}_i$ are orthogonal) $B_{ij}\langle \mathbf{v}_i, \mathbf{u}_j \rangle = B_{ij}^2 \cdot \frac{\mathbf{v}_i^2}{\sqrt{n}} = \frac{1}{2n}$. This gives $\sum_{i,j} B_{ij}\langle \mathbf{v}_i, \mathbf{u}_j \rangle = n^{3/2} \cdot (1/2n) = n^{1/2}/2$.

From these calculations, we have $\forall i, j$, $|\mathbf{v}_i \cdot \mathbf{u}_j| \leq \mathbf{u}_j^2$ (thus satisfying (6.2); other inequalities of this type are trivially satisfied). Further we saw that $\sum_i \mathbf{v}_i^2 + \sum_j \mathbf{u}_j^2 = 1$.

This gives a feasible solution of value $\Omega(n^{1/2})$, and hence the SDP has an $\widetilde{\Omega}(n^{1/4})$ integrality gap.

Connection to the star example. This gap instance can also be seen as a collection of $n^{1/2}$ stars (vertices in L are the 'centers'). In each 'co-ordinate' (corresponding to the orthogonal $\mathbf{v}_i$), the assigment looks like a star. $O(\sqrt{n})$ different co-ordinates allow us to satisfy the constraints (6.2).

Note also that the gap instance is bipartite. This matches the improved rounding algorithm we saw before. Thus for bipartite instances, the analysis of the SDP is optimal up to logarithmic factors.

6.2.1 Other relaxations for QP-Ratio

For problems in which variables can take more than two values (e.g. CSPs with alphabet size $r > 2$), it is common to use a relaxation where for every vertex u (assume an underlying graph), we have variables $x_u^{(1)}, .., x_u^{(r)}$, and constraints such as $\langle x_u^{(i)}, x_u^{(j)} \rangle = 0$ and $\sum_i \langle x_u^{(i)}, x_u^{(i)} \rangle = 1$ (intended solution being one with precisely one of these variables being 1 and the rest 0).

We can use such a relaxation for our problem as well: for every x_i, we have three vectors a_i, b_i, and c_i, which are supposed to be 1 if $x_i = 0, 1$, and -1 respectively (and 0 otherwise). In these terms, the objective becomes

$$\sum_{i,j} A_{ij} \langle b_i, b_j \rangle - \langle b_i, c_j \rangle - \langle c_i, b_j \rangle + \langle c_i, c_j \rangle = \sum_{i,j} A_{ij} \langle b_i - c_i, b_j - c_j \rangle.$$

The following constraints can be added

$$\sum_i b_i^2 + c_i^2 = 1 \tag{6.6}$$

$$\langle a_i, b_j \rangle, \langle b_i, c_j \rangle, \langle a_i, c_j \rangle \geq 0 \text{ for all } i, j \tag{6.7}$$

$$\langle a_i, a_j \rangle, \langle b_i, b_j \rangle, \langle c_i, c_j \rangle \geq 0 \text{ for all } i, j \tag{6.8}$$

$$\langle a_i, b_i \rangle = \langle b_i, c_i \rangle = \langle a_i, c_i \rangle = 0 \tag{6.9}$$

$$a_i^2 + b_i^2 + c_i^2 = 1 \text{ for all } i \tag{6.10}$$

Let us now see why this relaxation does not perform better than the one in (6.2). Suppose we start with a vector solution $\mathbf{u}_i$ to the earlier program. Suppose these are vectors in $\Re^d$. We consider vectors in $\Re^{n+d+1}$, which we define using standard direct sum notation (to be understood as concatenating co-ordinates). Here e_i is a vector in $\Re^n$ with 1 in the ith position and 0 elsewhere. Let 0_n denote the 0 vector in $\Re^n$.

We set (the last term is just a one-dim vector)

$$\begin{aligned} b_i &= 0_n \oplus \frac{\mathbf{u}_i}{2} \oplus (\frac{|\mathbf{u}_i|}{2}) \\ c_i &= 0_n \oplus -\frac{\mathbf{u}_i}{2} \oplus (\frac{|\mathbf{u}_i|}{2}) \\ a_i &= \sqrt{1-\mathbf{u}_i^2} \cdot e_i \oplus 0_d \oplus (0) \end{aligned}$$

It is easy to check that $\langle a_i, b_j \rangle = \langle a_i, c_j \rangle = 0$, and $\langle b_i, c_j \rangle = \frac{1}{4} \cdot (-\langle \mathbf{u}_i, \mathbf{u}_j \rangle + |\mathbf{u}_i||\mathbf{u}_j|) \geq 0$ for all i, j (and for $i = j$, $\langle b_i, c_i \rangle = 0$). Also, $b_i^2 + c_i^2 = \mathbf{u}_i^2 = 1 - a_i^2$. Further, $\langle b_i, b_j \rangle = \frac{1}{4} \cdot (\langle \mathbf{u}_i, \mathbf{u}_j \rangle + |\mathbf{u}_i||\mathbf{u}_j|) \geq 0$. Last but not least, it can be seen that the objective value is

$$\sum_{i,j} A_{ij} \langle b_i - c_i, b_j - c_j \rangle = \sum_{i,j} A_{ij} \langle \mathbf{u}_i, \mathbf{u}_j \rangle,$$

as desired. Note that we never even used the inequalities (6.2), so it is only as strong as the eigenvalue relaxation (and weaker than the sdp relaxation we consider).

Additional valid constraints of the form $a_i + b_i + c_i = v_0$ (where v_0 is a designated fixed vector) can be introduced – however it it can be easily seen that these do not add any power to the relaxation.

6.3 Hardness of approximating QP-Ratio

Given that our algorithmic techniques give only an $n^{1/3}$ approximation in general, and the natural relaxations do not seem to help, it is natural to ask how hard we expect the problem to be. Our results in this direction are as follows: we show that the problem is APX-hard i.e., there is no PTAS unless $P = NP$. Next, we show that there cannot be a constant factor approximation assuming that Max k-AND is hard to approximate 'on average' (related assumptions are explored in [37]).

Let us however cut to the chase, and first give a natural distribution over instances in which approximating to a factor better than n^c for some small c seems beyond the reach of our algorithms. This is a 'candidate hard distribution' for the QP-Ratio problem, in the same vein as the planted version of DkS from Section 3.2.2.

6.3.1 A candidate hard distribution

To reconcile the large gap between our upper bounds and lower bounds, we describe a natural distribution on instances we do not know how to approximate to a factor better than n^δ (for some fixed $\delta > 0$).

Let $\mathcal{G}$ denote a bipartite random graph with vertex sets V_L of size n and V_R of size $n^{2/3}$, left degree n^δ for some small δ (say 1/10) [i.e., each edge between V_L and V_R is picked i.i.d. with prob. $n^{-(9/10)}$]. Next, we pick a random (planted) subset P_L of V_L of size $n^{2/3}$ and random assignments $\rho_L : P_L \mapsto \{+1, -1\}$ and $\rho_R : V_R \mapsto \{+1, -1\}$. For an edge between $i \in P_L$ and $j \in V_R$, $a_{ij} := \rho_L(i)\rho_R(j)$. For all other edges we assign $a_{ij} = \pm 1$ independently at random.

The optimum value of such a *planted* instance is roughly n^δ, because the assignment of ρ_L, ρ_R (and assigning 0 to $V_L \setminus P_L$) gives a solution of value n^δ. However, for $\delta < 1/6$, we do not know how to find such a planted assignment: simple counting and spectral approaches do not seem to help. Making progress on such instances would be the first step to obtaining better algorithms for the problem.

6.3.2 APX hardness of QP-Ratio

Let us first prove a very basic inapproximability result, namely that there is no PTAS unless $P = NP$.

We reduce Max-Cut to an instance of QP-Ratio. The following is well-known (we can also start with other QP problems instead of Max-Cut)

> There exist constants $1/2 < \rho' < \rho$ such that: given a graph $G = (V, E)$ which is regular with degree d, it is NP-hard to distinguish between
> YES. $\text{MaxCut}(G) \geq \rho \cdot \frac{nd}{2}$, and
> NO. $\text{MaxCut}(G) \leq \rho' \cdot \frac{nd}{2}$.

Given an instance $G = (V, E)$ of Max-Cut, we construct an instance of QP-Ratio which has V along with some other vertices, and such that in an OPT solution to this QP-Ratio instance, *all* vertices of V would be picked (and thus we can argue about how the best solution looks).

First, let us consider a simple instance: let $abcde$ be a 5-cycle, with a cost of $+1$ for edges ab, bc, cd, de and -1 for the edge ae. Now consider a QP-Ratio instance defined on this graph (with ± 1 weights). It is easy to check that the best ratio is obtained when precisely four of the vertices are given non-zero values, and then we can get a numerator cost of 3, thus the optimal ratio is 3/4.

Now consider n cycles, $a_i b_i c_i d_i e_i$, with weights as before, but scaled up by d. Let A denote the vertex set $\{a_i\}$ (similarly $B, C, ..$). Place a clique on the set of vertices A, with each edge having a cost $10d/n$. Similarly, place a clique of the same weight on E. Now let us place a copy of the graph G on the set of vertices C.

It turns out (it is actually easy to work out) that there is an optimal solution with the following structure: (a) all a_i are set to 1, (b) all e_i are set to -1 (this gives good values for the cliques, and good value for the $a_i b_i$ edge), (c) c_i are set to ± 1 depending on the structure of G, (d) If c_i were set to $+1$, $b_i = +1$, and $d_i = 0$; else $b_i = 0$ and $d_i = -1$ (Note that this is precisely where the 5-cycle with one negative sign helps!)

Let $x_1, ..., x_n \in \{-1, 1\}$ be the optimal assignment to the Max-Cut problem. Then as above, we would set $c_i = x_i$. Let the cost of the MaxCut solution be $\theta \cdot \frac{nd}{2}$. Then we set $4n$ of the $5n$ variables to ± 1, and the numerator is (up to lower order terms):

$$2 \cdot (10d/n)\frac{n^2}{2} + \theta \cdot \frac{nd}{2} + 3nd = (\Delta + \theta)nd,$$

where Δ is an absolute constant.

We skip the proof that there is an optimal solution with the above structure. Thus we have that it is hard to distinguish between a case with ratio $(\Delta + \rho')d/4$, and $(\Delta + \rho)d/4$, which rules out a PTAS for the problem.

6.3.3 Reduction from Random k-AND

We start out by quoting the assumption we use.

Conjecture 6.9 (Hypothesis 3 in [37]). *For some constant $c > 0$, for every k, $\exists \Delta_0$, such that for every $\Delta > \Delta_0$, there is no polynomial time algorithm that, on most k-AND formulas with n-variables and $m = \Delta n$ clauses, outputs* **'typical'***, but never outputs* **'typical'** *on instances with $m/2^{c\sqrt{k}}$ satisfiable clauses.*

The reduction to QP-Ratio is as follows: Given a k-AND instance on n variables $X = \{x_1, x_2, \ldots x_n\}$ with m clauses $C = \{C_1, C_2, \ldots C_m\}$, and a parameter $0 < \alpha < 1$, let $A = \{a_{ij}\}$ denote the $m \times n$ matrix such that a_{ij} is $1/m$ if variable x_j appears in clause C_i as is, a_{ij} is $-1/m$ if it appears negated and 0 otherwise.

Let $f : X \to \{-1, 0, 1\}, g : C \to \{-1, 0, 1\}$ denote functions which correspond to assignments. Let $\mu_f = \sum_{i \in [n]} |f(x_i)|/n$ and $\mu_g = \sum_{j \in m} |g(C_j)|/m$. Let

$$\vartheta(f, g) = \frac{\sum_{ij} a_{ij} f(x_i) g(C_j)}{\alpha \mu_f + \mu_g}. \tag{6.11}$$

Observe that if we treat $f(), g()$ as variables, we obtain an instance of QP-Ratio[2]. We pick $\alpha = 2^{-c\sqrt{k}}$ and Δ a large enough constant so that Conjecture 6.9 and Lemmas 6.11 and 6.12 hold. The completeness follows from the natural assignment

Lemma 6.10 (Completeness)**.** *If α fraction of the clauses in the k-AND instance can be satisfied, then there exists function f, g such that θ is at least $k/2$.*

Proof. Consider an assignment that satisfies an α fraction of the constraints. Let f be such that $f(x_i) = 1$ if x_i is true and -1 otherwise. Let g be the indicator of (the α fraction of the) constraints that are satisfied by the assignment. Since each such constraint contributes k to the sum in the numerator, the numerator is at least αk while the denominator 2α. □

Soundness: We will show that for a typical random k-AND instance (i.e., with high probability), the maximum value $\vartheta(f, g)$ can take is at most $o(k)$.

Let the maximum value of ϑ obtained be ϑ_{max}. We first note that there exists a solution f, g of value $\vartheta_{max}/2$ such that the equality $\alpha\mu_f = \mu_g$ holds[3] – so we only need consider such assignments.

Now, the soundness argument is two-fold: if only a few of the vertices (X) are picked ($\mu_f < \frac{\alpha}{400}$) then the expansion of small sets guarantees that the value is small (even if each picked edge contributes 1). On the other hand, if many vertices (and hence clauses) are picked, then we claim that for every assignment to the variables (every f), only a small fraction ($2^{-\omega(\sqrt{k})}$) of the clauses contribute more than $k^{7/8}$ to the numerator.

The following lemma handles the case when $\mu_f < \alpha/400$.

[2]Note that as described, the denominator is *weighted*; we need to replicate the variable set X roughly $\alpha\Delta$ times (each copy has same set of neighbors in C) in order to reduce to an unweighted instance. We skip this straightforward detail.

[3]if $\alpha\mu_f > \mu_g$, we can pick more constraints such that the numerator does not decrease (by setting $g(C_j) = \pm 1$ in a greedy way so as to not decrease the numerator) till $\mu_{g'} = \alpha\mu_f$, while losing a factor 2. Similarly for $\alpha\mu_f < \mu_g$, we pick more variables.

Lemma 6.11. *Let k be an integer, $0 < \delta < 1$, and Δ be large enough. If we choose a bipartite graph with vertex sets X, C of sizes $n, \Delta n$ respectively and degree k (on the C-side) uniformly at random, then w.h.p., for every $T \subset X, S \subset C$ with $|T| \le n\alpha/400$ and $|S| \le \alpha|T|$, we have $|E(S,T)| \le \sqrt{k}|S|$.*

Proof. Let $\mu := |T|/|X|$ (at most $\alpha/400$ by choice), and $m = \Delta n$. Fix a subset S of C of size $\alpha\mu m$ and a subset T of X of size μn. The expected number of edges between S and T in G is $\mathbb{E}[E(S,T)] = k\mu \cdot |S|$. Thus, by Chernoff-type bounds (we use only the *upper tail*, and we have negative correlation here),

$$\Pr[E(S,T) \ge \sqrt{k}|S|] \le \exp\big(-\frac{(\sqrt{k}|S|)^2}{k\mu \cdot |S|}\big) \le \exp\left(-\alpha m/10\right)$$

The number of such sets S, T is at most $2^n \times \sum_{i=1}^{\alpha^2 m/400} \binom{m}{i} \le 2^n 2^{H(\alpha^2/400)m} \le 2^{n+\alpha m/20}$. Union bounding and setting $m/n > 20/\alpha$ gives the result. $\square$

Now, we bound $\vartheta(f,g)$ for solutions such that $\alpha\mu_f = \mu_g \ge \alpha^2/400$ using the following lemma about random instances of k-AND.

Lemma 6.12. *For large enough k and Δ, a random k-AND instance with Δn clauses on n variables is such that: for any assignment, at most a $2^{\frac{-k^{3/4}}{100}}$ fraction of the clauses have more than $k/2+k^{7/8}$ variables 'satisfied' [i.e. the variable takes the value dictated by the AND clause] w.h.p.*

Proof. Fix an assignment to the variables X. For a single random clause C, the expected number of variables in the clause that are satisfied by the assignment is $k/2$. Thus, the probability that the assignment satisfies more than $k/2(1+\delta)$ of the clauses is at most $\exp(-\delta^2 k/20)$. Further, each k-AND clause is chosen independently at random. Hence, by setting $\delta = k^{-\frac{1}{8}}$ and taking a union bound over all the 2^n assignments gives the result (we again use the fact that $m \gg n/\alpha$). $\square$

Lemma 6.12 shows that for every $\{\pm 1\}^n$ assignment to the variables x, at most $2^{-\omega(\sqrt{k})}$ fraction of the clauses contribute more than $2k^{7/8}$ to the numerator of $\vartheta(f,g)$. We can now finish the proof of the soundness part above.

Proof of Soundness. Lemma 6.11 shows that when $\mu_f < \alpha/400$, $\vartheta(f,g) = O(\sqrt{k})$. For solutions such that $\mu_f > \alpha/400$, i.e., $\mu_g \geq \alpha^2/400 = 2^{-2\sqrt{k}}/400$, by Lemma 6.12 at most $2^{-\omega(\sqrt{k})}$ ($\ll \mu_g/k$) fraction of the constraints contribute more than $k^{7/8}$ to the numerator. Even if the contribution is k [the maximum possible] for this small fraction, the value $\vartheta(f,g) \leq O(k^{7/8})$. □

These lemmas shows together show a gap of k vs $k^{7/8}$ assuming Hypothesis 6.9. Since we can pick k to be arbitrarily large, we can conclude that QP-Ratio is hard to approximate to any constant factor.

6.3.4 Reductions from ratio versions of CSPs

Here we ask: is there a reduction from a *ratio version* of Label Cover to QP-Ratio? For this to be useful we must also ask: is the (appropriately defined) ratio version of Label Cover hard to approximate? The answer to the latter question turns out to be yes, but unfortunately, we do not know how to reduce from Ratio-LabelCover.

Here, we present a reduction starting from a ratio version of *Unique Games* to QP-Ratio (inspired by [9], who give a reduction from Label Cover to Quadratic Programming, without the ratio). However, we do not know whether it is hard to approximate for the parameters we need. While it seems related to *Partial Unique Games* introduced by [65], they have an additional size constraint, that at least α fraction of vertices should be labeled, which enables a reduction from *Unique Games with Small-set Expansion.* However, a key point to note is that we do not need 'near perfect' completeness, as in typical UG reductions.

We hope the Fourier analytic tools we use to analyze the ratio objective could find use in other PCP-based reductions to ratio problems. Let us now define a ratio version of Unique Games, and a useful *intermediate* QP-Ratio problem.

Definition 6.13 (Ratio UG). *Consider a unique label cover instance* $\mathcal{U}\big(G(V,E),[R],\{\pi_e|e\in E\}\big)$. *The value of a partial labeling* $L : V \to [R] \cup \{\perp\}$ *(where label* $\perp$ *represents it is unassigned) is*

$$val(L) = \frac{|\{(u,v)\in E|\pi_{u,v}(L(u)) = L(v)\}|}{|\{v \in V|L(v) \neq \perp\}|}$$

The (s,c)*-Ratio UG problem is defined as follows: given* $c > s > 0$ *(to be thought of as constants), and an instance* $\mathcal{U}$ *on a regular graph* G*, distinguish between the two cases:*

- **YES:** *There is a partial labeling* $L : V \to [R] \cup \{\perp\}$*, such that* $val(L) \geq c$.
- **NO:** *For every partial labeling* $L : V \to [R] \cup \{\perp\}$*,* $val(L) < s$.

The main result of this section is a reduction from (s,c)-Ratio UG to QP-ratio. We first introduce the following *intermediate* problem:

QP-Intermediate. Given $A_{(n\times n)}$ with $A_{ii} \leq 0$, maximize

$$\frac{x^T A x}{\sum_i |x_i|} \text{ s.t. } x_i \in [-1,1].$$

Note that A is allowed to have diagonal entries (albeit only negative ones), and that the variables are allowed to take values in the *interval* $[-1,1]$.

Lemma 6.14. *Let* A *define an instance of QP-Intermediate with optimum value* opt_1. *There exists an instance* B *of QP-Ratio on* $(n\cdot m)$ *variables, with* $m \leq \max\{\frac{2\|A\|_1}{\varepsilon}, 2n\} + 1$*, and the property that its optimum value* opt_2 *satisfies* $\mathsf{opt}_1 - \varepsilon \leq \mathsf{opt}_2 \leq \mathsf{opt}_1 + \varepsilon$. *[Here* $\|A\|_1 = \sum_{i,j}|a_{ij}|$*.]*

Proof. The idea is to view each variable as an average of a large number (in this case, m) of new variables: thus a fractional value for x_i is 'simulated' by setting some of the new variables to ± 1 and the others zero. This is analogous to the construction in [9], and we skip the details. $\square$

Thus from the point of view of approximability, it suffices to consider QP-Intermediate. We now give a reduction from Ratio UG to QP-Intermediate.

Input: An instance $\Upsilon = (V, E, \Pi)$ of Ratio UG, with alphabet $[R]$.

Output: A QP-Intermediate instance $\mathcal{Q}$ with number of variables $N = |V| \cdot 2^R$.

Parameters: $\eta := 10^6 n^7 2^{4R}$

Construction:

- For every vertex $u \in V$, we have 2^R variables, indexed by $x \in \{-1,1\}^R$. We will denote these by $f_u(x)$, and view f_u as a function on the hypercube $\{-1,1\}^R$.
- Fourier coefficients (denoted $\widehat{f_u}(S) = \mathbb{E}_x[\chi_S(x) f_u(x)]$) are linear forms in the variables $f_u(x)$.
- For $(u,v) \in E$, define $T_{uv} = \sum_i \widehat{f_u}(\{i\}) \widehat{f_v}(\{\pi_{uv}(i)\})$.
- For $u \in V$, define $L(u) = \sum_{S:|S| \neq 1} \widehat{f_u}(S)^2$.
- The instance of QP-Intermediate we consider is

$$\mathcal{Q} := \max \frac{\mathbb{E}_{(u,v)\in E} T_{uv} - \eta \mathbb{E}_u L(u)}{\mathbb{E}_u |f_u|_1},$$

  where $|f_u|_1$ denotes $\mathbb{E}_x[|f_u(x)|]$.

Lemma 6.15. *(Completeness) If the value of Υ is $\geq \alpha$, then the reduction gives an instance of QP-Intermediate with optimum value $\geq \alpha$.*

Proof. Consider an assignment to Υ of value α and for each u set f_u to be the corresponding dictator (or $f_u = 0$ if u is assigned $\perp$). This gives a ratio at least α (the $L(u)$ terms contribute zero for each u). $\square$

Lemma 6.16. *(Soundness) Suppose the* QP-Intermediate *instance obtained from a reduction (starting with Υ) has value τ, then there exists a solution to Υ of value $\geq \tau^2/C$, for an absolute constant C.*

Proof. Consider an optimal solution to the instance $\mathcal{Q}$ of QP-Intermediate, and suppose it has a value $\tau > 0$. Since the UG instance is regular, we have

$$val(\mathcal{Q}) = \frac{\sum_u \mathbb{E}_{v\in\Gamma(u)} T_{uv} - \eta \sum_u L(u)}{\sum_u \|f_u\|_1}. \tag{6.12}$$

First, we move to a solution such that the value is at least $\tau/2$, and for every u, $|f_u|_1$ is either zero, or is "not too small". The choice of η will then enable us to conclude that each f_u is 'almost linear' (there are no higher level Fourier coefficients).

Lemma 6.17. *There exists a solution to $\mathcal{Q}$ of value at least $\tau/2$ with the property that for every u, either $f_u = 0$ or $\|f_u\|_1 > \frac{\tau}{n2^{2R}}$.*

Proof. Let us start with the optimum solution to the instance. First, note that $\sum_u \|f_u\|_1 \geq 1/2^R$, because if not, $|f_u(x)| < 1$ for every u and $x \in \{-1,1\}^R$. Thus if we scale all the f_u's by a factor $z > 1$, the numerator increases by a z^2 factor, while the denominator only by z; this contradicts the optimality of the initial solution. Since the ratio is at least τ, we have that the numerator of (6.12) (denoted $\mathcal{N}$) is at least $\tau/2^R$.

Now since $|\widehat{f_u}(S)| \leq \|f_u\|_1$ for any S, we have that for all u, v, $T_{uv} \leq R\cdot \|f_u\|_1 \|f_v\|_1$. Thus $\mathbb{E}_{v\in\Gamma(u)} T_{uv} \leq R \cdot \|f_u\|_1$. Thus the contribution of u s.t. $\|f_u\|_1 < \tau/(n2^{2R})$ to $\mathcal{N}$ is at most $n \times R \cdot \frac{\tau}{n2^{2R}} < \frac{\tau}{2^{R+1}} < \mathcal{N}/2$. Now setting all such $f_u = 0$ will only decrease the denominator, and thus the ratio remains at least $\tau/2$. [We have ignored the $L(u)$ term because it is negative and only improves when we set $f_u = 0$.] $\square$

For a boolean function f, we define the 'linear' and the 'non-linear' parts to be

$$f^{=1} := \sum_i \widehat{f}(i)\chi(\{i\}) \quad \text{ and } f^{\neq 1} := f - f^{=1} = \sum_{|S|\neq 1} \widehat{f}(S)\chi(S).$$

We will now state a couple of basic lemmas we will use.

Lemma 6.18. *[9] Let $f_u : \{-1,1\}^R \to [-1,1]$ be a solution to $\mathcal{Q}$ of value $\tau > 0$. Then*

$$\forall u \in V \qquad \sum_{i=1}^{R} |\widehat{f}_u(\{i\})| \leq 2.$$

Proof. Assume for sake of contradiction that $\sum_i \widehat{f}_u(\{i\}) > 2$.
Since $f_u^{=1}$ is a linear function with co-efficients $\{\widehat{f}_u(\{i\})\}$, there exists some $y \in \{-1,1\}^R$ such that $f_u^{=1}(y) = \sum_i |\widehat{f}_i(\{i\})| > 2$. For this y, we have $f^{\neq 1}(y) = f(y) - f^{=1}(y) < -1$.

Hence $|f^{\neq 1}|_2^2 > 2^{-R}$, which gives a negative value for the objective, for our choice of η. □

The following is the well-known Berry-Esséen theorem (which gives a quantitative version of central limit theorem). The version below is from [62].

Lemma 6.19. *Let $\alpha_1, \ldots, \alpha_R$ be real numbers satisfying $\sum_i \alpha_i^2 = 1$, and $\alpha_i^2 \leq \tau$ for all $i \in [R]$. Let X_i be i.i.d. Bernoulli (± 1) random variables. Then for all $\theta > 0$, we have*

$$\big|\Pr[\sum_i \alpha_i X_i > \theta] - N(\theta)\big| \leq \tau,$$

where $N(\theta)$ denotes the probability that $g > \theta$, for g drawn from the univariate Gaussian $\mathcal{N}(0,1)$.

Getting back, our choice of η will be such that:

1. For all u with $f_u \neq 0$, $\|f_u^{\neq 1}\|_2^2 \leq \|f_u\|_1^2/10^6$. Using Lemma 6.17 (and the naïve bound $\tau \geq 1/n$), this will hold if $\eta > 10^6 n^7 2^{4R}$. [A simple fact used here is that $\sum_u \mathbb{E}[T_{uv}] \leq nR$.]

2. For each u, $\|f_u^{\neq 1}\|_2^2 < \frac{1}{2^{2R}}$. This will hold if $\eta > n2^{2R}$ and will allow us to use Lemma 6.18.

Also, since by Cauchy-Schwarz inequality, $|f_u|_2^2 \geq \delta_u^2$, we can conclude that 'most' of the Fourier weight of f_u is on the linear part for *every* u. We now show that the Cauchy-Schwarz inequality above must be tight up to a constant (again, for every u).

A key step in the analysis is the following: if a boolean function f is 'nearly linear', then it must also be *spread out* [i.e. $\|f\|_2 \approx \|f\|_1$]. This helps us deal with the main issue in a reduction with a ratio objective – showing we cannot have a large numerator along with a very small value of $\|f\|_1$ (the denominator). Morally, this is similar to a statement that a boolean function with a *small support* cannot have all its Fourier mass on the *linear* Fourier coefficients.

Lemma 6.20. *Let $f : \{-1,1\}^R \mapsto [-1,1]$ satisfy $\|f\|_1 = \delta$. Let $f^{=1}$ and $f^{\neq 1}$ be defined as above. Then if $\|f\|_2^2 > (10^4+1)\delta^2$, we have $\|f^{\neq 1}\|_2^2 \geq \delta^2$.*

Proof. Suppose that $\|f\|_2^2 > (10^4+1)\delta^2$, and for the sake of contradiction, that $\|f^{\neq 1}\|_2^2 < \delta^2$. Thus since $\|f\|_2^2 = \|f^{=1}\|_2^2 + \|f^{\neq 1}\|_2^2$, we have $\|f^{=1}\|_2^2 > (100\delta)^2$.

If we write $\alpha_i = \widehat{f}(\{i\})$, then $f^{=1}(x) = \sum_i \alpha_i x_i$, for every $x \in \{-1,1\}^R$. From the above, we have $\sum_i \alpha_i^2 > (100\delta)^2$. Now if $|\alpha_i| > 4\delta$ for some i, we have $\|f^{=1}\|_1 > (1/2)\cdot 4\delta$, because the value of $f^{=1}$ at one of $x, x \oplus e_i$ is at least 4δ for every x. Thus in this case we have $\|f^{=1}\|_1 > 2\delta$.

Now suppose $|\alpha_i| < 4\delta$ for all i, and so we can use Lemma 6.19 to conclude that $\Pr_x(f^{=1}(x) > 100\delta/10) \geq 1/4$, which in turn implies that $\|f^{=1}\|\|_1 > (100\delta/10) \cdot \Pr_x(f^{=1}(x) > 100\delta/10) > 2\delta$.

Thus in either case we have $\|f^{=1}\|_1 > 2\delta$. This gives $\|f - f^{=1}\|_1 > \|f^{=1}\|_1 - \|f\|_1 > \delta$, and hence $\|f - f^{=1}\|_2^2 > \delta^2$ (Cauchy-Schwarz), which implies $\|f^{\neq 1}\|_2^2 > \delta^2$, which is what we wanted. $\square$

Now, let us denote $\delta_u = |f_u|_1$. Since Υ is a unique game, we have for every edge (u, v) (by Cauchy-Schwarz),

$$T_{u,v} = \sum_i \widehat{f}_u(\{i\})\widehat{f}_v(\{\pi_{uv}(i)\}) \le \sqrt{\sum_i \widehat{f}_u(\{i\})^2}\sqrt{\sum_j \widehat{f}_u(\{j\})^2} \le |f_u|_2|f_v|_2 \quad (6.13)$$

Now we can use Lemma 6.20 to conclude that in fact, $T_{u,v} \le 10^4\delta_u\delta_v$. Now consider the following process: while there exists a u such that $\delta_u > 0$ and $\mathbb{E}_{v\in\Gamma(u)}\delta_v < \frac{\tau}{4\cdot 10^4}$, set $f_u = 0$. We claim that this process only increases the objective value. Suppose u is such a vertex. From the bound on T_{uv} above and the assumption on u, we have $\mathbb{E}_{v\in\Gamma(u)}T_{uv} < \delta_u \cdot \tau/4$. If we set $f_u = 0$, we remove at most twice this quantity from the numerator, because the UG instance is regular [again, the $L(u)$ term only acts in our favor]. Since the denominator reduces by δ_u, the ratio only improves (it is $\ge \tau/2$ to start with).

Thus the process above should terminate, and we must have a non-empty graph at the end. Let S be the set of vertices remaining. Now since the UG instance is regular, we have that $\sum_u \delta_u = \sum_u \mathbb{E}_{v\in\Gamma(u)}\delta_v$. The latter sum, by the above is at least $|S| \cdot \tau/(4 \cdot 10^4)$. Thus since the ratio is at least $\tau/2$, the numerator $\mathcal{N} \ge |S| \cdot \frac{\tau^2}{8\cdot 10^4}$.

Now let us consider the following natural randomized rounding: for vertex $u \in S$, assign label i with probability $|\widehat{f}_u(\{i\})|/(\sum_i |\widehat{f}_u(\{i\})|)$. Now observing that $\sum_i |\widehat{f}_u(\{i\})| < 2$ for all u (Lemma 6.18), we can obtain a solution to ratio-UG of value at least $\mathcal{N}/|S|$, which by the above is at least τ^2/C for a constant C.

This completes the proof of Lemma 6.16. $\square$

This completes the reduction from a ratio version of UG to QP-Ratio.

Chapter 7

Conclusions and Future Directions

In the thesis, we have studied questions related to extracting structure from graphs and matrices. We also saw applications in both theory and practice in which questions of this nature arise. In graphs, we studied in detail the so-called *densest k-subgraph* problem. Our algorithms suggest that the following average case problem is key to determining the approximation ratio: given a random graph with a certain average degree, how dense a k-subgraph should we *plant* in it so as to be able to *detect* the planting?

We saw that the notion of *log-density* is crucial in answering this question. In particular, if the planted subgraph has a higher log-density than the entire graph, certain counting based algorithms will detect the planting. Furthermore, we saw that these ideas of counting are general enough to carry over to the case of arbitrary graphs, in which they help recover approximate dense subgraphs.

While log-density is a *barrier* for counting based algorithms, we also saw that if we are willing to allow mildly subexponential time algorithms, we can extend our algorithms to give an n^{ε} improvement in approximation factor (over the original factor of $n^{1/4}$) with running time $2^{n^{\varepsilon}}$. This type of a smooth tradeoff between the approximation ratio and running time is very interesting, and desirable for other approximation problems as well!

Next, we studied the problem of approximating the $q \mapsto p$ operator norm of matrices for different values of p, q. Such norms generalize singular values, and help capture several crucial properties of the matrices (or the underlying graphs). For the case $p \leq q$, we developed an understanding of the approximability of the problem. When the matrix has all non-negative entries, we proved that the $q \mapsto p$ norm can be computed exactly in this case. We further saw that without this restriction, the problem is NP-hard to approximate to any constant factor.

The algorithmic result, though it is specific to positive matrices has some points of interest. Firstly, the problem is that of maximizing a convex function over a convex domain, which we are somehow able to solve. Further, the algorithm is extremely simple, one obtained by writing $\nabla f = 0$ (for a natural f) as a fixed point equation, and taking many iterates of this equation. In fact, this algorithm was proposed by Boyd over thirty years ago [24], and we prove that it converges in polynomial time for this setting of parameters. Finally, the case of positive matrices arises in certain applications – one we describe is that of constructing *oblivious* routing schemes for multicommodity flow under an ℓ_p objective, a problem studied by Englert and Räcke [34].

It is interesting to see if the techniques used to show polynomial time convergence can be used in other contexts: algorithms based on fixed point iteration are quite common in practice, but formal analyses are often plagued with issues of converging to local optima, cycling, etc. Further, we are able to prove that even though the problem we are solving is not *as is* a convex optimization question, it shares many properties which allow us to solve it efficiently (such as the uniqueness of maximum, connectedness of level sets, and so on).

We then obtained inapproximability results for computing $q \mapsto p$ by simple gadget reductions, but it seems crucial in these reductions to have $p \leq q$. For $p > q$, which is referred to as the *hypercontractive* case, both our algorithmic and inapproximability results fail to work.

Finally, we studied the QP-Ratio problem, which we could see as a ratio version of the familiar quadratic programming problem. The key difficulty in this problem is to capture $x_i \in \{-1, 0, 1\}$ constraints using the algorithmic techniques we know. Even though it is a simple modification of the maximum density subgraph probem (which can be solved exactly in polynomial time), the best we know to approximate the objective is to a factor of $O(n^{1/3})$, in general.

The main deterrent is the "ratio" objective. Furthermore, proving hardness results for the problem seem quite difficult, because of precisely the same reason. We can, however give evidence for inapproximability in terms of more 'average-case' assumptions such as *Random k-AND*. In this respect our knowledge of the problem (from the point of view of approximation) is very similar to that of the densest k-subgraph question.

7.1 Open problems and directions

We will collect below some of the open problems we stated implicitly or explicitly in the chapters preceeding.

Beating the log-density in polynomial time. This is the most natural question arising from our work on the densest subgraph problem. For simplicity, let us consider the random planted problem. Can we distinguish between the following distributions in polynomial time?

YES: G is a random graph drawn from $G(n, p)$, with $p = n^{\delta}/n$, for some parameter δ. In G, a subgraph on k-vertices and average degree $k^{\delta-\varepsilon}$ is *planted*, for some small parameter ε.

NO: G is simply a random graph drawn from $G(n,p)$, with $p = n^{\delta}/n$, for some parameter δ.

We do not know how to solve this problem, for instance, in the case $\delta = 1/2$ and $\varepsilon = 1/10$.

A simpler $n^{1/4}$ approximation algorithm? Our algorithm, though quite simple to describe, is based on carefully *counting* caterpillar structures in the graph. It is not clear if this is the only way to go about it – for instance, could certain random walk based algorithms *mimic* the process of trying to find a sub-graph with higher log-density?

Finding dense subgraphs is quite important in practice, so progress on making the algorithms simpler is quite valuable.

Fractional powers of graphs. For certain values of the parameters, such as $r/s = 1/k$ for integers k, caterpillars are simply paths of length k. Thus in this case, our algorithm can be viewed as a walk type argument, and is related to arguments about the kth power of the graph. In this sense, caterpillars for general r, s seem to achieve the effect of taking *fractional* powers of a graph. Can this notion be of value in other contexts?

Hardness of DkS. This again, was mentioned many times in the thesis. The best known inapproximability results for DkS are extremely weak – they give a hardness of approximation of a small constant factor. Is it hard to approximate DkS to say, an $O(\log n)$ factor?

Our results, and belief related conjectures seems to suggest that the answer is yes.

Computing hypercontractive norms. As we have seen, computing $\|A\|_{q\mapsto p}$ for $p > q$ is a problem with applications in different fields. However for many interesting ranges of parameters, the complexity of the problem is very poorly understood. It seems plausible that the problem is in fact hard to approximate to a constant factor. Such results have been recently obtained for $2 \mapsto 4$ norms, however the general problem remains open.

Another distinguishing problem. We recall now the candidate *hard distribution* for the QP-Ratio problem which we described in Section 6.3.1. We pose it as a problem of distinguishing between two distributions on matrices:

1. A is formed as follows. Let G denote a bipartite random graph with vertex sets V_L of size n and V_R of size $n^{2/3}$, left degree n^{δ} for some small δ (say 1/10) [i.e., each edge between V_L and V_R is picked i.i.d. with prob. $n^{-9/10}$]. Next, we pick a random (planted) subset P_L of V_L of size $n^{2/3}$ and random assignments $\rho_L : P_L \mapsto \{+1, -1\}$ and $\rho_R : V_R \mapsto \{+1, -1\}$. For an edge between $i \in P_L$ and $j \in V_R$, $a_{ij} := \rho_L(i)\rho_R(j)$. For all other edges we assign $a_{ij} = \pm 1$ independently at random.

2. A is formed by taking a bipartite random graph with vertex sets V_L of size n and V_R of size $n^{2/3}$, left degree n^{δ}, and ± 1 signs on each edge picked independently at random.

Bibliography

[1] N Alon and V.D Milman. 1, isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, Series B*, 38(1):73 – 88, 1985.

[2] Noga Alon, Sanjeev Arora, Rajsekar Manokaran, Dana Moshkovitz, and Omri Weinstein. Manuscript. 2011.

[3] Noga Alon, Sanjeev Arora, Rajsekar Manokaran, Dana Moshkovitz, and Omri Weinstein. Inapproximability of densest k-subgraph from average case hardness, 2012.

[4] Noga Alon, W. Fernandez de la Vega, Ravi Kannan, and Marek Karpinski. Random sampling and approximation of max-csp problems. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC '02, pages 232–239, New York, NY, USA, 2002. ACM.

[5] Noga Alon, Michael Krivelevich, and Benny Sudakov. Finding a large hidden clique in a random graph. pages 457–466, 1998.

[6] Noga Alon and Assaf Naor. Approximating the cut-norm via grothendieck's inequality. *SIAM J. Comput.*, 35:787–803, April 2006.

[7] Sanjeev Arora, Boaz Barak, Markus Brunnermeier, and Rong Ge. Computational complexity and information asymmetry in financial products (extended abstract). In Andrew Chi-Chih Yao, editor, *ICS*, pages 49–65. Tsinghua University Press, 2010.

[8] Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. In *FOCS*, pages 563–572. IEEE Computer Society, 2010.

[9] Sanjeev Arora, Eli Berger, Elad Hazan, Guy Kindler, and Muli Safra. On non-approximability for quadratic programs. In *FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 206–215, Washington, DC, USA, 2005. IEEE Computer Society.

[10] Sanjeev Arora, Rong Ge, Sushant Sachdeva, and Grant Schoenebeck. Finding overlapping communities in social networks: toward a rigorous approach. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, pages 37–54, New York, NY, USA, 2012. ACM.

[11] Sanjeev Arora, Subhash Khot, Alexandra Kolla, David Steurer, Madhur Tulsiani, and Nisheeth K. Vishnoi. Unique games on expanding constraint graphs are easy: extended abstract. In Cynthia Dwork, editor, *STOC*, pages 21–28. ACM, 2008.

[12] Yuichi Asahiro, Refael Hassin, and Kazuo Iwama. Complexity of finding dense subgraphs. *Discrete Appl. Math.*, 121(1-3):15–26, 2002.

[13] Boaz Barak. Truth vs proof: The unique games conjecture and feiges hypothesis, 2012.

[14] Boaz Barak, Fernando G.S.L. Brandao, Aram W. Harrow, Jonathan Kelner, David Steurer, and Yuan Zhou. Hypercontractivity, sum-of-squares proofs, and their applications. In *Proceedings of the 44th symposium on Theory of Computing*, STOC '12, pages 307–326, New York, NY, USA, 2012. ACM.

[15] Boaz Barak, Prasad Raghavendra, and David Steurer. Rounding semidefinite programming hierarchies via global correlation. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:65, 2011.

[16] William Beckner. Inequalities in fourier analysis. *The Annals of Mathematics*, 102(1):pp. 159–182, 1975.

[17] Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an $o(n^{1/4})$ approximation for densest k-subgraph. In *STOC '10: Proceedings of the 42nd ACM symposium on Theory of computing*, pages 201–210, New York, NY, USA, 2010. ACM.

[18] Aditya Bhaskara, Moses Charikar, Venkatesan Guruswami, Aravindan Vijayaraghavan, and Yuan Zhou. Polynomial integrality gaps for strong sdp relaxations of densest k-subgraph. In *ACM SIAM Symposium on Discrete Algorithms*, 2012.

[19] Aditya Bhaskara, Moses Charikar, Rajsekar Manokaran, and Aravindan Vijayaraghavan. On quadratic programming with a ratio objective. In *International Colloquium on Automata and Language Processing (ICALP) 2012*, pages 187–198, 2012.

[20] Aditya Bhaskara and Aravindan Vijayaraghavan. Approximating matrix p-norms. *CoRR*, abs/1001.2613, 2010.

[21] Punyashloka Biswal. Hypercontractivity and its applications. *CoRR*, abs/1101.2913, 2011.

[22] Sergey Bobkov and Prasad Tetali. Modified log-sobolev inequalities, mixing and hypercontractivity. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, STOC '03, pages 287–296, New York, NY, USA, 2003. ACM.

[23] Bla Bollobs. Cambridge University Press, 2001.

[24] David W. Boyd. The power method for p norms. *Linear Algebra and its Applications*, 9:95 – 101, 1974.

[25] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, pages 107–117, Amsterdam, The Netherlands, The Netherlands, 1998. Elsevier Science Publishers B. V.

[26] Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX '00: Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 84–95, London, UK, 2000. Springer-Verlag.

[27] Moses Charikar, MohammadTaghi Hajiaghayi, and Howard J. Karloff. Improved approximation algorithms for label cover problems. In Amos Fiat and Peter Sanders, editors, *ESA*, volume 5757 of *Lecture Notes in Computer Science*, pages 23–34. Springer, 2009.

[28] Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Near-optimal algorithms for unique games. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, STOC '06, pages 205–214, New York, NY, USA, 2006. ACM.

[29] Moses Charikar and Anthony Wirth. Maximizing quadratic programs: Extending grothendieck's inequality. In *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 54–60, Washington, DC, USA, 2004. IEEE Computer Society.

[30] Eden Chlamtac and Madhur Tulsiani. Convex relaxations and integrality gaps. *Handbook on Semidefinite, Cone and Polynomial Optimization*, 2010.

[31] Dario Cordero-Erausquin and Michel Ledoux. Hypercontractive measures, talagrands inequality, and influences. In Boaz Klartag, Shahar Mendelson, and Vitali Milman, editors, *Geometric Aspects of Functional Analysis*, volume 2050 of *Lecture Notes in Mathematics*, pages 169–189. Springer Berlin / Heidelberg, 2012.

[32] Amit Deshpande, Kasturi R. Varadarajan, Madhur Tulsiani, and Nisheeth K. Vishnoi. Algorithms and hardness for subspace approximation. *CoRR*, abs/0912.1403, 2009.

[33] Yon Dourisboure, Filippo Geraci, and Marco Pellegrini. Extraction and classification of dense communities in the web. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 461–470, New York, NY, USA, 2007. ACM.

[34] Matthias Englert and Harald Räcke. Oblivious routing in the l_p-norm. In *Proc. of the 50th FOCS*, 2009.

[35] U. Feige, G. Kortsarz, and D. Peleg. Personal communication - the dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[36] U. Feige and M. Seltser. On the densest k-subgraph problems. Technical report, Jerusalem, Israel, Israel, 1997.

[37] Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the 34th annual ACM Symposium on Theory of Computing (STOC'02)*, pages 534–543. ACM Press, 2002.

[38] Uriel Feige and Robert Krauthgamer. The probable value of the lovász–schrijver relaxations for maximum independent set. *SIAM J. Comput.*, 32(2):345–370, 2003.

[39] Alan M. Frieze and Ravi Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999.

[40] Alan M. Frieze and Ravi Kannan. A new approach to the planted clique problem. In *FSTTCS'08*, pages 187–198, 2008.

[41] Z. Füredi and J. Komlós. The eigenvalues of random symmetric matrices. *Combinatorica*, 1:233–241, 1981.

[42] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.*, 18(1):30–55, 1989.

[43] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Co., San Francisco, Calif., 1979.

[44] David Gibson, Ravi Kumar, and Andrew Tomkins. Discovering large dense subgraphs in massive graphs. In *Proceedings of the 31st international conference on Very large data bases*, VLDB '05, pages 721–732. VLDB Endowment, 2005.

[45] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

[46] Leonard Gross. Logarithmic sobolev inequalities. *American Journal of Mathematics*, 97(4):pp. 1061–1083, 1975.

[47] Anupam Gupta, Mohammad T. Hajiaghayi, and Harald Räcke. Oblivious network design. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 970–979, New York, NY, USA, 2006. ACM.

[48] J. Hastad. Clique is hard to approximate within n̂(1-eps). In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS '96, pages 627–, Washington, DC, USA, 1996. IEEE Computer Society.

[49] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[50] Taher Haveliwala, Sepandar Kamvar, Dan Klein, Chris Manning, and Gene Golub. Computing pagerank using power extrapolation. Technical Report 2003-45, Stanford InfoLab, 2003.

[51] Nicholas J. Higham. Estimating the matrix p-norm. *Numer. Math*, 62:511–538, 1992.

[52] Mark Jerrum and Alistair Sinclair. Conductance and the rapid mixing property for markov chains: the approximation of permanent resolved. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 235–244, New York, NY, USA, 1988. ACM.

[53] Ravindran Kannan and Santosh Vempala. Spectral algorithms. *Foundations and Trends in Theoretical Computer Science*, 4:157288, 1974.

[54] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:126–135, 2004.

[55] Subhash Khot. Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In *Proceedings of the 44th Annual IEEE Symposium on the Foundations of Computer Science (FOCS'04)*, pages 136–145, 2004.

[56] Guy Kindler, Assaf Naor, and Gideon Schechtman. The ugc hardness threshold of the ℓ_p grothendieck problem. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 64–73, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[57] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999.

[58] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the Web for emerging cyber-communities. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1481–1493, 1999.

[59] Alexandre Megretski. Relaxation of quadratic programs in operator theory and system analysis. In *In Systems, Approximation, Singular Integral Operators, and Related Topics*, pages 365–392, 2001.

[60] A. Nemirovski, C. Roos, and T. Terlaky. On maximization of quadratic form over intersection of ellipsoids with common center. *Mathematical Programming*, 86:463–473, 1999. 10.1007/s101070050100.

[61] Yurii Nesterov. Semidefinite relaxation and nonconvex quadratic optimization. *Optimization Methods and Software*, 9:141–160, 1998.

[62] Ryan O'Donnel. Analysis of boolean functions - lecture 21. In *http://www.cs.cmu.edu/ odonnell/boolean-analysis/*.

[63] Rina Panigrahy, Kunal Talwar, and Udi Wieder. Lower bounds on near neighbor search via metric expansion. *CoRR*, abs/1005.0418, 2010.

[64] Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 255–264, New York, NY, USA, 2008. ACM.

[65] Prasad Raghavendra and David Steurer. Integrality gaps for strong sdp relaxations of unique games. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:575–585, 2009.

[66] Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In Leonard J. Schulman, editor, *STOC*, pages 755–764. ACM, 2010.

[67] Prasad Raghavendra, David Steurer, and Madhur Tulsiani. Reductions between expansion problems. In *Manuscript*, 2010.

[68] Oded Regev. Lattice-based cryptography. In *In Proc. of the 26th Annual International Cryptology Conference (CRYPTO*, pages 131–141, 2006.

[69] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 1997.

[70] Joel Spencer. The probabilistic method. In *SODA '92: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 41–47, Philadelphia, PA, USA, 1992. Society for Industrial and Applied Mathematics.

[71] Anand Srivastav and Katja Wolf. Finding dense subgraphs with mathematical programming, 1999.

[72] Daureen Steinberg. Computation of matrix norms with applications to robust optimization. Research thesis, Technion - Israel University of Technology, 2005.

[73] Terence Tao. Open question: deterministic uup matrices, 2012.

[74] Luca Trevisan. Max cut and the smallest eigenvalue. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 263–272, New York, NY, USA, 2009. ACM.

[75] Madhur Tulsiani. Csp gaps and reductions in the lasserre hierarchy. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 303–312, New York, NY, USA, 2009. ACM.

[76] R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. *ArXiv e-prints*, November 2010.