Beyond Worst-Case Analysis in Approximation Algorithms

Aravindan Vijayaraghavan

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 Charikar

September 2012

Abstract

Worst-case analysis has had many successes over the last few decades, and has been the tool of choice in analyzing approximation guarantees of algorithms. Yet, for several important optimization problems, approximation algorithms with good guarantees have been elusive.

In this dissertation, we look beyond worst-case analysis to obtain improved approximation guarantees for fundamental graph optimization problems, given that real-world instances are unlikely to behave like worst-case instances. We study average-case models and other structural assumptions that seem to capture properties of real-world instances, and design improved approximation algorithms in these settings. In some cases, this average-case study also gives us surprising insights into the worst-case approximability.

In the first part of the thesis, we design better algorithms for realistic average-case instances of graph partitioning. We propose and study a semi-random model for graph partitioning problems, that is more general than previously studied random models, and that we believe captures many real-world instances well. We design new $O(1)$ approximation algorithms for classical graph partitioning problems like Balanced Separator, Multicut, Small set expansion and Sparsest cut for these semi-random instances. We also explore how other assumptions about the stability of a planted solution can lead to improved approximation guarantees.

In the second part of the thesis, we consider the Densest k-Subgraph problem, which is an important, yet poorly understood problem, from both average-case and worst-case perspectives. We first study a natural average-case version of the problem and design new counting-based algorithms with improved guarantees. These average-case algorithms directly inspire new worst-case algorithms, which surprisingly match our guarantees from the average-case: our algorithms use linear-programming hierarchies to give a $n^{1/4+\epsilon}$ factor approximation while running in time $n^{O(1/\epsilon)}$, for

every $\epsilon > 0$. This natural average-case model also identifies a concrete barrier for progress on Densest k-subgraph, and seems to captures exactly the extent of current approaches. These results suggest that the approximability of the Densest k-subgraph problem may be similar from both worst-case and average-case perspectives, in contrast to graph partitioning.

Acknowledgements

I am greatly indebted to my advisor Moses Charikar, for his constant support and guidance throughout my PhD. I vividly remember going into his office as an first-year graduate student, who was very excited about the prospect of research, but had little idea on how to go about it. He patiently taught me how to do research. He has been the perfect role-model to aspire towards — his brilliance, his clarity of thought, his style of approaching problems and his ability to captivate the audience were always truly inspiring. He has been unwavering in his encouragement and support, especially during the lean patches, and it always seemed like he knew exactly what I needed at each point in my career. I could not have hoped for anyone better as my advisor.

I am very thankful to Konstantin Makarychev and Yury Makarychev, who have not only been great collaborators, but have also been my mentors and good friends too. A good fraction of the contents of this thesis is based on my work with them.

I would like to thank the members on my PhD committee: Moses Charikar, Sanjeev Arora, Mark Braverman, David Blei and Konstantin Makarychev for their time, and their useful comments and suggestions. I would like to thank the faculty members of the theory group at Princeton, particularly Sanjeev Arora and Boaz Barak who have been very encouraging and helpful in their feedbacks and discussions. I also thank the friendly staff at Princeton, especially Melissa Lawson and Mitra Kelly for their administrative help. I gratefully acknowledge my funding through Moses Charikar's grants: NSF grants MSPA-MCS 0528414, CCF 0832797, and AF 0916218.

Special thanks are due to my primary collaborator, and friend Aditya Bhaskara. I have been very lucky to have him around, whenever I have felt the need for a good sounding-board to discuss my thoughts, whether research or personal.

I have been very fortunate in having some terrific researchers as my collaborators including Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Julia Chuzhoy, Uriel Feige, Venkatesan Guruswami, Konstantin Makarychev, Yury Makarychev, Rajsekar

Manokaran and Yuan Zhou. I am very grateful for all the fun and learning I have had during these interactions. I am also very grateful to Ravi Sundaram for introducing me to the field of approximation algorithms during my undegraduate days.

My five years at Princeton have been an exciting journey, and for this I have mainly my friends to thank. Many thanks to my friends in the theory group including MohammadHossein Bateni, Eden Chlamtac, Seshadri Commandur, Rong Ge, Moritz Hardt, Shi Li, Rajsekar Manokaran, Indraneel Mukherjee, Yonathan Namaad, Huy Nguyen, Prasad Raghavendra, Sushant Sachdeva, Sid Sen, Srikanth Srinivasan, David Steurer, Madhur Tulsiani and Anuradha Venugopalan for the useful coffee-room discussions and creating such a great research atmosphere at Princeton. My thanks also to Ravishankar for the many long hours on the phone discussing research and life, right from high school days. My good friends comprising Aditya, Badam, Ketra, Rajsekar, Chris, Ana, CJ, Sid, Karthik, NG, Easwaran, Arthi and many others, have been my family away from home, and made my stay at Princeton tremendous fun.

I am very thankful to my family, especially my brother Koushik and Shalini for all their affection and encouragement.

I thank my fiancée Arthi for her unwavering love and support during my PhD. I have been very fortunate to have always had her by my side to share all the ups-and-downs of graduate school life.

Last but not the least, I am deeply indebted to my parents Geetha and Vijayaraghavan, without whose encouragement, love and hard work, this thesis would not have been possible. I dedicate this dissertation to them.

Dedicated to my dearest Appa and Amma.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Over the last five decades, there has been an extensive study on understanding and designing algorithms for the plethora of combinatorial optimization problems that we encounter in computer science like scheduling, clustering and graph partitioning. A canonical example is the (minimum) Balanced Cut problem, a fundamental graph optimization problem that is particularly useful for divide-and-conquer approaches and clustering:

Definition 1.1 (Balanced Cut)**.** *Given a graph G, find a partition of the vertices $V(G)$ into two roughly equal pieces,*[1] *that minimizes the number of edges between the pieces.*

To measure the performance of a candidate algorithm, worst-case analysis has traditionally been the tool of choice. Here, we are concerned with how well the candidate algorithm performs for every instance (in particular, the worst instance for this algorithm). The paradigm of worst-case analysis has been very successful in understanding computational complexity – even for those optimization problems which do not seem to have polynomial time algorithms (assuming $P \neq NP$), we have had success in designing algorithms which compute approximately optimal solutions

[1] the pieces S and $V(G) \setminus S$ should have at most size $2n/3$

efficiently. As an example, for the Balanced Cut problem Arora et al. [13] design an algorithm, that always computes a cut which is at most a $O(\sqrt{\log n})$ factor larger than the optimum. Formally, an algorithm $\mathcal{A}$ has worst-case approximation ratio $\alpha > 1$ if for every instance of the problem, $\mathcal{A}$ computes a solution which is within a multiplicative factor of α from the optimal solution [2].

An algorithm with good worst-case guarantees is the best possible scenario, as in the case of knapsack (FPTAS : $1 + \epsilon$ factor for any constant $\epsilon > 0$ [84]), makespan scheduling (2-factor approximation [67]), network design (2-factor approximation [51]) etc. Moreover, worst-case analysis is more suited to a mathematical treatment, and it comes with a rich history of mathematical tools and techniques to rely on.

On a contrasting note, for many important problems like graph partitioning, routing and coloring, the best known worst-case algorithms only give poly-logarithmic, or even polynomial factor approximations (for example, Balanced Cut). Morever, progress on better approximation algorithms for these problems seems to have stalled. In fact, some of these problems are provably hard to approximate upto large factors in the worst-case (assuming $P \neq NP$).

However, our primary interest in these optimization problems comes from the different scenarios that they arise in. Hence, we are chiefly concerned with those instances that come up from different applications in the real world. Yet, instances that we encounter in the real world are not worst case instances. Further, instances that are produced in these hardness reductions are contrived, and we are unlikely to encounter them in practice. In the absence of good approximations in the worst case, a compelling research direction is

Question 1. *Can we design approximation algorithms with good provable guarantees, if we focus on instances that tend to come up in practice?*

[2] Unless otherwise stated, we say that an algorithm $\mathcal{A}$ is a α approximation if $\mathcal{A}$ has worst-case approximation ratio α

Besides, worst-case complexity is too harsh of a metric for potential algorithms. It may be too hard, if not impossible (for problems with provable inapproximabililty results) to design algorithms which give good guarantees for every instance of the problem. In such cases, we would be very satisfied to have algorithms which work well in most cases. For example, some heuristics for graph partitioning problems (e.g. [54]) seem to work well empirically for many instances that come up in practice. To have a more rigorous understanding of how good these approaches are, we need a metric which captures algorithms which give good guarantees for *most instances* of the problem. This notion is captured by *average-case* analysis.

Moreover, from a more theoretical standpoint, worst case complexity exhibits only one facet of the algorithm or the problem in question. For instance, it can not distinguish between trivial algorithms and clever heuristics, which have the same worst-case complexity. Besides, even among problems with bad worst-case complexity, some of these problems seem to be easy for most instances, while some other problems seem to be very hard even on average. In fact, such problems are extremely useful for cryptographers to base their secure protocols on. A more complete understanding can come out of a study of average-case complexity.

The average-case analysis of a problem requires a probability distribution over the instances of the problem to start with. Since we need an algorithm with good guarantees for "most" instances of the problem, we require that the algorithm gives good approximation guarantees with high probability over the given distribution i.e. for all but a negligible fraction of the input, (see Section 2.2 for a more precise definition). The average-case complexity for a problem can differ widely depending on the distribution.

Over the last two decades, many works have studied approximation algorithms for average-case instances of important combinatorial optimization problems. Bui et al. [26] initiated a study of average-case algorithms for graph partitioning with

their random model for Balanced Cut. A series of works [36, 25, 53, 35, 34, 33] studied spectral approaches and various other heuristics for this problem. Later [70] extended this random model to a general class of graph partitioning problems and designed spectral algorithms which recover the partitions for a large range of parameters. Such random models have also been studied in the context of k-coloring [24], minimum Steiner tree [62] etc.

Typically, the distribution over instances corresponds to the standard Erdös-Renyi random graph model, or simple variants of it with a "planted" solution. Such random models assume that every edge is chosen independently at random. While the complete independence in these random models is very useful in algorithm design, it also renders them very unrealistic. This excess independence seems to be too strong an assumption to make about real-world instances — for instance, the clustering coefficients seen in graphs and networks from practice are much higher than that of a random graph (see [71] for more details).

Semi-random Models: The Realistic Average-case

Since the average-case approximability can differ widely depending on the distribution, the choice of the distribution over instances is crucial. The ideal distribution is one that is both natural and realistic i.e. a distribution that captures most of the instances that come up in practice. Semi-random models are a class of average-case models, whose generality lie in between the random model and worst-case. They allow more structure in the instances, by incorporating adversarial choices while generating the instances.

Blum and Spencer [24] first introduced a semi-random model for k-coloring, where random noise corrupted the choices of an adversary who is designing the instance. Feige and Kilian [39] considered a semi-random model for Balanced Cut, that made the previous random models more robust, by allowing an adversary to modify a

random instance, in such a way that would only make the planted solution even better. Yet, these semi-random models were more in the spirit of the random models, with added robustness. Kolla, Makarychev and Makarychev [61] recently studied semi-random models for the Unique Games Conjecture[56], and showed that the four most natural ways of introducing randomness to generate instances of Unique Games render it easy.

The extent of the adversarial choices dictates the generality of the model and how effective the semi-random model is in capturing real-world instances. On the one hand, we want to design algorithms with better guarantees than the worst-case. But on the other hand, we want the semi-random model to be general enough so that it can capture most instances we see in practice. Hence, the main challenge is

Question 2. *Can we design algorithms with better provable guarantees, for a realistic average-case model of the problem?*

The Approximation Stability Assumption

In a very exciting recent work, Balcan, Blum and Gupta [17] considered a different kind of a condition, called Approximation Stability, in modeling instances from practice. For many applications of clustering problems, there is a ground-truth clustering which we hope to get at. While the aim is to get as close to this target as possible, the objective function is used as a proxy to capture the closeness to the target. Designing approximation algorithms for such applications is useful only when we know that even approximately optimal solutions are close to the target clustering (which we assume is exactly the optimal solution for now). In such cases, they ask if we can design better algorithms, if we assume that even approximately optimal solutions are close to the target.

Blum et al. [17] applied the stability assumption to metric clustering problems like k-means and k-median, and designed polynomial time approximation schemes

(PTAS) for these problems, when only $O(1)$ factor approximations were known in the worst-case. One of the main questions that arise out of their work is whether such a stability assumption could lead to better approximations for combinatorial optimization problems like graph partitioning (a similar question was also posed by [22] recently). In other words,

Question 3. *Can we design better approximation algorithms, if we assume that the instance has a unique optimal solution with stability?*

In this thesis, we address these questions by studying the approximability of important graph optimization problems both on realistic average-case (semi-random) models, and under additional conditions like the stability assumption. This study also provides us surprising insights into the worst-case approximability in some cases. Before we elaborate on our contributions, we first introduce the problems which we study in the thesis.

1.1 The Case Studies

We will primarily be concerned with two basic graph optimization problems: Graph Partitioning and the Densest k-subgraph problem.

1.1.1 Graph Partitioning

Graph Partitioning is a class of problems, where the goal is to divide the vertices of the graph into a disjoint set of clusters, while (typically) minimizing the number of edges across the clusters, subject to some problem-specific constraints. An important problem that falls in this class is Balanced Cut, which was defined earlier (Definition 1.1). A variant which aims at finding lopsided partitions is Small Set Expansion

Definition 1.2 (Small Set Expansion)**.** *Given a graph G, find a partition of the vertices $V(G)$ into two pieces S and $V(G) \setminus S$ with $|S| = k$, that minimizes the number of edges between the pieces.*

Other problems which fall in this class include Minimum Multicut and Small Set Expansion (definitions in Chapter 2).

Graph partitioning problems are among the most fundamental problems in combinatorial optimization. The interest in Graph Partitioning is two-fold. Firstly, they are ubiquitous in computer science, with numerous applications in image segmentation, machine learning and more broadly, science and engineering. They are also used as basic building blocks for clustering or divide-and-conquer approaches in many combinatorial algorithms. Secondly, they are crucial in understanding important algorithmic barriers like the Unique Games Conjecture [56, 76][3].

There has been extensive research on graph partitioning problems, which has been mostly focused on analyzing the worst case performance of optimization algorithms. Over the last two decades, poly-logarithmic approximation algorithms were developed for such fundamental graph partitioning problems as Minimum Bisection [73], Balanced Cut [66, 13], Multicut [44]. Yet, there has been little success in obtaining constant factor approximation algorithms for these problems, and some recent results [57, 59, 74, 77] suggest that this may even be hard, assuming the Unique Games conjecture [56] and its variants.

Average-case. There has been considerable interest in the past, in studying average-case models for graph partitioning. These models typically assume that every edge is chosen independently at random, with the edge probability inside clusters being more than that between clusters.

[3]A conjecture about the hardness of Small Set Expansion implies the Unique Games conjecture

This random model attracted a lot of attention and was studied in a series of papers by [36, 25, 53, 35, 34, 33]. These papers explored several techniques for solving the problem — flow-based, combinatorial, spectral techniques, simulated annealing and go-with-the-leader technique. The algorithm of Boppana [25] finds the planted balanced cut $(S, V \setminus S)$ w.h.p. if $\varepsilon_2 - \varepsilon_1 > C\sqrt{\varepsilon_2 \log n/n}$. Later McSherry [70] obtained similar results for a more general class of graph partitioning problems. Coja-Oghlan [33] extended the result of Boppana to the case when $\varepsilon_2 - \varepsilon_1 > C(\frac{1}{n} + \sqrt{\varepsilon_2 \log(n\varepsilon_2)/n})$. Note that if $\varepsilon_2 - \varepsilon_1 = o(\sqrt{\varepsilon_2 \log n/n})$ then the random graph has exponentially many minimum bisections and with high probability, the planted bisection is not a minimum bisection [33]. The algorithm of Coja-Oghlan [33] finds w.h.p. a minimum balanced cut rather than the planted balanced cut. Feige and Killian [39] later made this model more robust, by allowing the adversary to remove edges between the clusters, and adding edges inside the two clusters : their algorithms make the spectral techniques of [25, 70] more robust. However, the main criticism against these models is that they typically assume too much independence (especially inside the clusters). Please see Section 2.6 for more details.

1.1.2 Densest k-subgraph

The other problem that we focus on, Densest k-subgraph (abbreviated as DkS) is the complement of Small Set Expansion. This problem may be seen as a natural optimization version of the classical NP-complete decision problem k-CLIQUE.

Definition 1.3 (Densest k-subgraph)**.** *Given a graph G and a parameter k, find the subgraph of G on at most k vertices with the maximum number of edges inside it.*

The approximability of DkS belies its extremely simple definition: despite much work on this important problem, it has been notorious in resisting progress on both the algorithmic and inapproximability fronts.

While the problem is believed to fairly hard to approximate, the best known inapproximability results just show that DkS does not admit a PTAS under various complexity theoretic assumptions. Feige [38] has shown this assuming random 3-SAT formulas are hard to refute, while more recently this was shown by Khot [56] assuming that NP does not have randomized algorithms that run in sub-exponential time (i.e. that $NP \not\subseteq \cap_{\epsilon>0} BPTIME(2^{n^\epsilon})$). Of course, DkS is NP-hard to compute exactly (as seen by the connection to k-CLIQUE.

The current best approximation ratio of $n^{1/3-\epsilon}$ for some small $\epsilon > 0$ was achieved by Feige, Kortsarz and Peleg [40]. Recently, and independently of our work, Goldstein and Langberg [47] presented an algorithm for which they computed the approximation ratio to be roughly $n^{0.3159}$. Other known approximation algorithms have approximation guarantees that depend on the parameter k. The greedy heuristic of Asahiro et al. [15] obtains an $O(n/k)$ approximation. Linear and semidefinite programming (SDP) relaxations were studied by Srivastav and Wolf [82] where they show how they can be used to get approximation ratios somewhat better than n/k.

Finding small dense subgraphs come up algorithmically in several settings (like detecting emerging communities [63], finding protein complexes and functional discovery in biological networks [50]). Further, the apparent inapproximability of this problem is a source of intractability in many problems with a strict budget, like [65, 6, 32]: the planted variants are particularly useful (see [8, 7, 60]) as an average-case hardness assumption like the Random 3-SAT assumption [38].

However, even for basic average-case models, the worst-case algorithms of [40] remain the best known. Thus, understanding the approximability of the problem, from both worst-case and average-case perspectives is an important challenge.

1.2 Our Contributions

In this thesis, we shed new light on the approximability of some fundamental graph optimizations problems from both average-case and worst-case perspectives.

In the first part of the thesis we design improved algorithms for graph partitioning in planted models which lie between the basic average-case and worst-case models, in their generality. First, we design new algorithms with improved approximation guarantees for realistic average-case models of graph partitioning problems. We first introduce a new semi-random model for graph partitioning that captures properties of real-world instances better than previous models. We then design constant factor approximations in these models for important graph partitioning problems like Balanced Cut, Multicut and Small Set Expansion, for which only poly-logarithmic approximations are known in the worst-case. Second, we study graph partitioning under a stability assumption, and design new algorithms for Balanced Cut and Small Set Expansion, which give constant factor approximations under this assumption.

In the second part of the thesis we focus on the Densest k-subgraph problem (DkS), and study it from both average-case and worst-case perspectives. We first study a natural average-case model of the problem, and design counting-based combinatorial algorithms for it. We then show how we can systematically translate these algorithms using linear programming hierarchies, to obtain the state-of-the-art worst-case approximation algorithms for Densest k-subgraph, with the same guarantees. Further, the natural average-case model identifies a concrete barrier for progress, and seems to captures exactly the extent of current approaches. In particular, this distribution of instances presents a barrier to current techniques involving Sherali-Adams hierarchy based SDP relaxations[4], which are among the most powerful algorithmic techniques, and capture most classes of known approximation algorithms. In contrast

[4]Semi-definite programming hierarchies for $\{0,1\}$ integer programming problems give a systematic way to generate successively stronger relaxations, which converge to the solution set of the problem.

to our results on graph partitioning, these results indicate that the worst-case and average-case approximabilities of Densest k-subgraph are suprisingly identical, and seem to go hand in hand. We now elaborate on these results (which are summarized in Table 1.1) in this section.

Problem	Worst Case	Semi-random	Random
Balanced Cut	$O(\sqrt{\log n})$ [13]	$O(1)$ (this thesis)	FPTAS[a] [25]
Multicut	$O(\log n)$ [44]	$O(1)$ (this thesis)	–
Small Set Expansion (size k)	$O(\sqrt{\log n \log(\frac{n}{k})})$ [18]	$O(1)$ (this thesis)	FPTAS [a][70]
Sparsest Cut	$O(\sqrt{\log n})$ [13]	$O(1)$ (this thesis)	FPTAS [a][70]
Densest k-subgraph	$n^{1/4}$ (this thesis)	$n^{1/4}$ (this thesis)	$n^{1/4}$ (this thesis)

[a]recovers the planted partition for appropriate parameters.

Table 1.1: Comparison of the approximation ratios in the worst-case, semi-random and random models.

1.2.1 Graph Partitioning in Realistic Average-case Models

Semi-random models for Graph Partitioning

The ideal distribution of instances is specific to the problem and on which use-cases of the problem we hope to capture. A good model for average-case model for graph partitioning tries to capture its chief application in graph clustering. When a practioner tries to solve his or her problem through graph partitioning, there is a belief that the instance has a good partitioning solutions. In other words, for the clustering corresponding to the optimal clustering, we expect much fewer edges between the different clusters, than inside the various clusters. The different average-case models we will encounter try to capture this property, to varying degrees of generality.

Over the last couple of decades, average-case models of graph partitioning problems like Balanced Cut have been studied in a series of works [26, 25, 35, 39, 70, 33]. However, the main criticism against these models is that they typically assume too much independence — for instance, every edge is chosen independently at random (even inside the clusters of the partition).

In Chapter 3, we define a new semi-random model for graph partitioning, which is more general than previous models, and which we believe capture real-world instances better. Before we proceed with the formal presentation of our model, let us discuss what we can reasonably assume about real-world instances. A real-world process that "generates" the graph partitioning (or clustering instance) adds an edge between clusters only when some random unexpected event happens. Therefore, in our opinion, it is reasonable to assume that edges between clusters are added at random. However, we cannot in general assume anything about edges within clusters (since their absence or presence does not affect the size of the cut between clusters). One could also view these edges between the clusters as random noise in an otherwise perfect clustering (partitioning).

This discussion leads to the following informal definition of semi-random instances: consider a set of vertices V and some clustering of V. A semi-random graph G on V is a graph with *arbitrary (adversarial)* edges inside clusters and *random* edges E_R between clusters (more generally, the set of edges between clusters might be a subset of a random set of edges). Please refer to Chapter 3 for more details on the model.

The random edges across clusters reflect the underlying belief of the practitioner — correlations between different clusters are unexpected (can be viewed as random noise). Since the correlations between items in a cluster can be arbitrary, our model is more realistic than previous models, and has good potential of capturing real-world instances.

The random edges E_R correspond to the "planted solution" in the semi-random graph partitioning instance. However, this planted solution E_R is not necessarily the optimal partitioning solution. Hence, we need to measure the approximation guarantees of algorithms for this semi-random model a little differently.

Measuring performance in semi-random models.

In typical average-case models, there is a clear planted solution, which turns out to be the optimum solution as well, with high probability. In contrast, semi-random models have adversarial steps — hence, the planted solution may no longer remain the optimal solution. For example, in the case of Small Set Expansion, the adversarial graph inside the larger cluster (of size $n - k$) could have a much smaller cut of the correct size (size k) than the planted cut (given by the random edges). In such cases, obtaining a solution whose cost compares favorably to the optimal solution would entail a worst-case approximation guarantee for the adversarial graph inside the larger cluster. To ensure that our task remains easier than worst-case approximations, we measure the performance of the algorithm against the planted solution.

Definition (Performance Ratio)**.** *The performance of the algorithm is measured by comparing the cost of edges cut by the algorithm (to partition the graph) to the expected number of edges in planted solution (the random edges E_R).*

In the rest of this thesis, we will also use *approximation ratio* to refer to the performance ratio of algorithms in semi-random models.

Improved approximations in the semi-random model

We design $O(1)$ factor approximation algorithms for "classical" graph partitioning problems in this semi-random model.

Informal Theorem. *Given a semi-random instance, there is an algorithm that finds with high probability a balanced cut* $(S', V \setminus S')$ *with* $|S'|, |V \setminus S'| = \Omega(n)$ *of cost*

$$O\left(|E_R| + n\sqrt{\log n}(\log\log n)^2\right).$$

Particularly if $\varepsilon > \sqrt{\log n}(\log\log n)^2/n$*, the cost of the cut is* $O(|E_R|) = O(\varepsilon n^2)$*.*

Informal Theorem. *Given a semi-random instance, there is an algorithm that finds with high probability a solution to the Small Set Expansion problem i.e., a subset* $S \subset V$ *of size* ρn*, of cost*

$$O\left(|E_R| + n\sqrt{\log n \log(1/\rho)}(\log\log n)^2\right)$$

Particularly if $\varepsilon\rho > \sqrt{\log n \log(1/\rho)}(\log\log n)^2/n$*, the cost of the cut is* $O(|E_R|) = O(\varepsilon\rho n^2)$*.*

Such results also hold for other basic graph partitioning problems like the Minimum Multicut and Sparsest Cut. The algorithm for the Small Set Expansion is not only interesting on its own, but can also be used to almost recover the original balanced cut under certain conditions. See Section 4.2 for a formal statement of the results.

We remark that as ε decreases, the problem becomes more challenging since the amount of randomness in the instances decreases. The performance guarantees given above show that as long as we have sufficient randomness, we can obtain $O(1)$ approximations. Alternately, these results show that we can obtain $O(1)$ approximations for semi-random instances upto an additive factor of $\tilde{O}(n\sqrt{\log n})$.

Note that the algorithm does not necessarily find the planted cut $(S, V \setminus S)$ since in general this is impossible. Indeed the adversary can just delete all edges between S and $V \setminus S$ and obtain an empty graph, or she can add every edge within S and

within $V \setminus S$ with probability ε and obtain a random $G(n, \varepsilon)$ graph. In either case, our algorithm has no information about the planted cut $(S, V \setminus S)$. However, if we assume that graphs induced by S and by $V \setminus S$ are combinatorial expanders, we can almost recover sets S and $V \setminus S$. This condition is similar to a stability assumption, with the added assumption of semi-randomness. This assumption for planted partitioning can be justified in the implicit belief that approximately optimal solutions are close to the planted partition.

Informal Theorem. *There is a constant $C > 1$, such that for every constant $\eta > 0$, given a semi-random instance G with combinatorial expansion $h(G[S]), h(G[V \setminus S]) \geq C\varepsilon n$ and $\varepsilon\eta > \sqrt{\log n \log(1/\eta)}(\log\log n)^2/n$, there is an algorithm that finds with high probability the partition $(S, V \setminus S)$ up to an error of $\pm\eta n$ vertices.*

A similar result also holds for the Small Set Expansion problem.

Comparison to previous average-case models and algorithms

In our semi-random model, the adversary has more power than in the model of Feige and Kilian. As in their model, the adversary can remove edges between S and $V \setminus S$ but additionally she has absolute control over induced graphs $G[S]$ and $G[V \setminus S]$ (whereas in the model of Feige and Kilian, she could only add extra edges to random $G(\frac{n}{2}, \varepsilon_2)$ subgraphs inside $G[S]$ and $G[V \setminus S]$).

Our algorithm for Balanced Cut, while designed for a more general model, also works in a wider range of parameters than the algorithms of Boppana[25], and Feige and Kilian[37] (note that the objective of our algorithms is slightly different, particularly we do not aim to recover the original partition precisely). To compare the algorithms, let us assume that probabilities ε_1 and ε_2 are of the same order of magnitude, $\varepsilon_1 = \Theta(\varepsilon)$ and $\varepsilon_2 = \Theta(\varepsilon)$. While the algorithm of Bopanna[25] and Feige and Kilian [39] require that $\varepsilon > C \log n/n$, we require only that $\varepsilon > C\sqrt{\log n}(\log\log n)^2/n$.

Our Techniques

We develop a new algorithmic framework for average-case models capturing many graph partitioning problems, which is very different from previous approaches. The core algorithmic idea is the "hidden solution sparsification" procedure which does the following: if E_R is the set of random edges added between different clusters of the partition (which is unknown to us, of course), we cut $O(|E_R|)$ edges and ensure that the remaining graph has one large component with $|E_R|/(\log^{\Omega(1)} n)$ edges across the different clusters of the intended clustering: running our favorite poly-logarithmic factor approximation algorithm on this remaining instance gives a $O(1)$-factor approximation overall.

This core algorithm tries to identify the random edges E_R, by using a semi-definite programming relaxation that defines a metric over the vertices. The crucial insight comes from a structural theorem, which establishes that most random edges in a semi-random instance are long according to the metric given by a "well-behaved" high-dimensional SDP solution. In this case, cutting long edges succeeds in removing half of the current set of random edges. Hence, this core procedure proceeds over roughly $O(\log \log n)$ phases — in each phase, we solve the SDP relaxation and use the vector solution to identify the set of few edges to cut.

1.2.2 Graph Partitioning under Stability conditions

In Chapter 6, we study graph partitioning problems when we have additional conditions about the stability of the optimum solution. One of the main question left open in the recent exciting work of Blum, Balcan and Gupta [17, 16] is

Question 4. *[17, 16] Can we design $O(1)$ approximations for graph partitioning problems like Balanced Cut, Sparsest Cut under the stability assumption?*

An assumption about the stability of the optimal solution for graph partitioning problems is equivalent to a condition about the expansion of the subgraph (please see Section 2.3.2 for formal definitions of different notions of expansion) inside the clusters (see Lemma 2.15). With this in mind, we study graph partitioning problems on instances, where roughly speaking, the expansion inside the clusters of the partition (for just the smaller cluster, in fact) is a constant factor more than the expansion of the planted partition. However, we need this condition about a stronger form of expansion: we assume that the algebraic expansion inside the clusters is a constant factor more than the combinatorial expansion of the planted partition. We call this *Planted Spectral Expander model*.

More precisely, we are given a graph G with a (planted) balanced cut $(S, V \setminus S)$ such that the normalized algebraic expansion of the induced graph $G[S]$ is greater than the combinatorial expansion (conductance) of the cut $(S, V \setminus S)$ by some constant factor.

This condition is satisfied for instance, under much the stronger assumption that the edges of $E(S, S)$ or $E(V \setminus S, V \setminus S)$ are chosen independently at random (with a higher probability) while $E(S, V \setminus S)$ is completely adversarial (this is the opposite of the semi-random model described earlier). Note that the famous Cheeger's inequality (Lemma 2.10) relates these two notions of expansion: hence, when the combinatorial expansion of the clusters is large enough (enough stability), then they have sufficient algebraic expansion to fit into our setting.

Results for the Planted Spectral Model We design algorithms which achieve constant factor bicriteria approximations for Balanced Cut and Small Set Expansion in this model.

Informal Theorem. *Given a planted spectral expander instance G, there is an algorithm that finds a balanced cut $(S', V \setminus S')$ with $|S'|, |V \setminus S'| = \Omega(n)$ of cost $O(E(S, V \setminus S)$.*

Note that we do not impose any restrictions on the graph $G[V \setminus S]$ and on edges in the cut $(S, V \setminus S)$; this result also applies to the case when $G[S]$ is a random graph with the appropriate parameters. Our algorithms are inspired by the results of [11, 68], where they infer global correlations between the vectors from local correlations and algebraic expansion.

1.2.3 Densest k-subgraph: An Average-Case Study

We begin the study of Densest k-subgraph from an average case perspective. We first describe a natural average-case model for Densest k-subgraph, which will be central to understand densest k-subgraph even in the worst-case. The average-case model that we study is inspired by the following simple fact: random graphs do not have dense subgraphs. Any potential algorithm for DkS should be able to identify a dense k-subgraph planted inside such a random graph. This leads us to the following natural average-case model, which we call the *Planted DkS model*:

Definition 1.4 (Planted DkS model)**.** *We are given a set of n vertices V, and three parameters $p \in [0, 1]$, $k, d \in [n]$. The graph is generated as follows:*

1. *Generate a graph $G_1 = \mathcal{G}(n, p)$ i.e. $\forall u, v \in V$ has an edge between them with probability p independently.*

2. *Adversary picks an arbitrary set of k vertices $V_H \subseteq V$, and adds arbitrary graph $H(V_H, E_H)$ of density d. H is refered to as the planted k-subgraph.*

If the graph H generated in step 2 of the model was also chosen randomly, this would correspond to a random planted model for DkS (analogous to the random

models for graph partitioning). The Planted DkS model can be viewed as a *semi-random model* because of the adversarial choices incorporated in step 2.

Any algorithm, that identifies a planted dense k-subgraph should at the very least, be able to certify that random graphs have *no dense k-subgraphs*. To this end, we first look at solving the following distinguishing (promise) problem — distinguish between G drawn from $\mathcal{G}(n,p)$, and G containing a k-subgraph H of certain density planted in it (generated according to the *Planted DkS model*).

We first design combinatorial algorithms which can solve the distinguishing problem in polynomial time for a certain range of parameters. At a high level, 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 (please see Chapter 7 for details):

Definition 1.5. *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 $|V|^{\alpha}$.*

For the above average-case model, we show that 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 $n^{O(1/(\beta-\alpha))}$.

The distinguishing algorithm is based on the fact that in $G(n,p)$, instances of any fixed constant size structure appear more often when the graph has a higher log-density. Our distinguishing algorithms involve cleverly counting appropriately defined trees of constant size in the graph — large counts indicate the presence of a dense k-subgraph, and small counts certify the absence of any dense k-subgraph with high probability.

1.2.4 Densest k-subgraph in the worst-case: an $n^{1/4}$ approximation

While the above intuition in terms of log-density comes from the average-case, log-density turns out to be the crucial factor in determining approximations in the worst-case as well. We design new algorithms for the worst-case, that are directly inspired by the counting algorithms we design for the average-case problem. Surprisingly, the approximation guarantees we obtain for arbitrary graphs matches the distinguishing factor we obtain for the average-case, leading to (roughly) an $n^{1/4}$ factor approximation algorithm. To do this systematic translation, we use a linear programming relaxation given by a standard linear program hierarchy like Lovasz-Schrijver (or Sherali-Adams).

At an informal level, we give an algorithm that *detects* whether G has a subgraph H on k vertices which has higher log-density than G does. That is, if G has log-density α, and it contains a subgraph H on k vertices with log-density $\beta > \alpha$ (average degree k^β), our algorithm will detect this. The most important case we need to consider is when $k = n^{1-\alpha}$ (that is, $Dk = n$, where D is the average degree in G): here, the density of a random k-subgraph is $O(1)$ (log-density 0). In this case, our algorithm outputs a subgraph on k vertices with non-trivial log-density — in fact, it is roughly $\beta - \alpha$.

Informally, we give a *family* of algorithms, parametrized by a rational number r/s, can be stated as follows (see Theorem 8.16 for a more precise statement):

Informal Theorem. *Let $s > r > 0$ be relatively prime integers, let G be an undirected graph with maximum degree $D = n^{r/s}$ ($\leq n/k$ [5]), which contains a k-subgraph H with average degree $d = \rho \cdot k^{r/s}$. Then there is an algorithm running in time $n^{O(r)}$ that finds a k-subgraph of average degree $\Omega(\rho)$.*

[5] This assumption can be made upto a loss of $O(\log n)$ factor (see Section 8.1.2)

This leads to an $O(n^{1/4+\epsilon})$ factor approximation in time $n^{O(1/\epsilon)}$.

To do this translation to the worst case, we use counts of exactly the same constant sized trees as before, to identify the vertices of the dense subgraph. One can view our algorithm as an attempt to certify that the given graph has no dense k-subgraphs (random-like) — the presence of a large count can be used to identify a dense k-subgraph. To translate this intuition, we use an linear programming relaxation given by $O(1/\epsilon)$ levels of a standard linear program hierarchy like Lovasz-Schrijver (or Sherali-Adams). In fact, our algorithm is one of the few cases (along with [29, 30, 78]) where hierarchies have been used to obtain better approximation algorithms in the worst-case.

1.2.5 Integrality gaps and Evidence for Hardness of Densest k-subgraph

The approximation ratio we achieve for general instances of DkS matches the "distinguishing ratio" we are currently able to achieve for the average-case. This suggests the following concrete open problem which seems to be a barrier for obtaining an approximation ratio of $n^{1/4-\epsilon}$ for DkS— distinguish between the following two distributions:

$\mathcal{D}_1$: graph G picked from $G(n, n^{-1/2})$, and

$\mathcal{D}_2$: graph G picked from $G(n, n^{-1/2})$ with the induced subgraph on $\sqrt{n}$ vertices replaced with $G(\sqrt{n}, n^{-(1/4+\epsilon)})$.

The believed inapproximability of Densest k-subgraph and its variants also comes up in various other problems that look for solutions supported on *small* sets[6]. However, considering the absence of good inapproximability results, it is important to

[6]In fact a public-key cryptosystem was designed in [7] based on the hardness of such an assumption.

provide some evidence in support of these barriers — say, using a candidate distribution of instances that seems hard for current techniques (as for 3-SAT [38], Clique [3]). Such a distribution could also serve as a spot-test for potential algorithms.

In the last part of the thesis, we show polynomial integrality gaps for strong SDP relaxations of the Densest k-subgraph problem. We study lift-and-project relaxations for DkS obtained from the Sherali-Adams Semi-definite Programming hierarchy. We show that even $\Omega\left(\frac{\log n}{\log \log n}\right)$ levels of the Sherali-Adams SDP relaxation have an integrality gap of $\tilde{O}(n^{1/4})$.

We note that prior results exhibiting gap instances for lift-and-project relaxations do so for problems that are *already known* to be hard to approximate under some suitable assumption; based on this hardness result, one would *expect* lift-and-project relaxations to have an integrality gap that matches the inapproximability factor. Our gap constructions for Densest k-Subgraph in this thesis are a rare exception to this trend, as the integrality gaps we show are substantially stronger than the (very weak) hardness bounds known for the problem. In the absence of good inapproximability results for this problem, we arguably give the strongest evidence that a natural distribution of instances for the problem is a barrier for current techniques.

1.3 Organization of the thesis

First we discuss the conventions, preliminaries and some background to understand the rest of the thesis in Chapter 2. This chapter will also give a formal definition of all the problems we will encounter in the thesis, and then discuss the various average-case models for graph partitioning that have been studied in the past. The rest of the thesis is broadly divided into two parts.

Chapters 3,4,5 will deal with designing improved approximation algorithms for average-case models of graph partitioning. First, in Chapter 3 we motivate and

describe formally the semirandom (average-case) model for graph partitioning that we study in this thesis. In Chapter 4 we describe the basic framework behind our new algorithms, and show $O(1)$ factor approximations for Balanced Cut and Multicut. In Chapter 5, we will give our algorithm for semi-random Small Set Expansion and Sparsest Cut, and show how they also help in successfully recovering the partitions under some extra assumptions. Finally, in Chapter 6, we will how we can also obtain $O(1)$ approximations for Balanced Cut and Small Set Expansion under some stability assumptions. The results in these chapters (3,4,5,6) are based on joint work with Konstantin Makarychev and Yury Makarychev [69].

The second part of the thesis (Chapters 7, 8, 9) deals the Densest k-subgraph problem (DkS). In Chapter 7, we first describe an average case model for DkS and design algorithms for it. In Chapter 8, we show how to translate these average-case algorithms to an algorithm with good worst-case guanrantees. In Chapter 9, we show how the average case model also points to a concrete barrier for improved approximations by showing integrality gaps for the Sherali-Adams SDP hierarchy. The results in Chapters 7 and 8 are based on joint work with Aditya Bhaskara, Moses Charikar, Eden Chlamtac and Uriel Feige [20]. The result in Chapter 9 is based on joint work with Aditya Bhaskara, Moses Charikar, Venkatesan Guruswami and Yuan Zhou [21].

Finally we conclude the thesis with some open problems and avenues for future work.

Chapter 2

Background

In this chapter we first introduce some basic terminology and tools that we will use in the rest of the thesis. Later we will review some of the average-case models that have been studied traditionally, especially in the context of graph partitioning.

2.1 Basic Notation

The following is a list of conventions and basic notations that we will follow in the thesis unless otherwise stated:

- $G(V, E)$ refers to an input undirected graph on n vertices, with vertex set V and edge set E.
- $E(S, T)$ will refer to the subset edges in E which have one end-point in S, and the other end point in T. We will use $E(S)$ to denote $E(S, S)$.
- The density of a subgraph H of G, refers to the average degree of the subgraph H.
- For $v \in V$, $\Gamma(v)$ denotes the set of neighbors of v, and for a set of vertices $S \subseteq V$, $\Gamma(S)$ denotes the set of all neighbors of vertices in S.

- for any number $x \in \mathbb{R}$, will use the notation $\text{fr}(x) = x - \lfloor x \rfloor$ to denote the fractional part of x.
- $\mathcal{H}$ will denote the euclidean space $\mathbb{R}^n$ for some sufficiently large finite n.
- For vector $\bar{u} \in \mathbb{R}^n$, $\|\bar{u}\|$ will denote its Euclidean norm (ℓ_2 norm) i.e. $\|\bar{u}\| = \sqrt{\sum_i u_i^2}$.
- For vectors $\bar{u}$,$\bar{v} \in \mathbb{R}^n$, we let $\langle \bar{u}, \bar{v} \rangle = \sum_i u_i v_i$ to represent the inner product between u and v.
- For a real valued random variable X, $\mathbb{E}[X]$ and $\mu(X)$ will denote its mean.
- The phrase *with high probability* or *w.h.p* will refer to 'with probability at least $1 - \frac{1}{q(n)}$', where $q(n)$ is an arbitrary polynomial in n (sometimes there will be a constant depending on the polynomial).
- Unless stated otherwise C will denote a sufficiently large constant which is ≥ 1.

2.2 Average Case Approximability

Here, we will clarify the notion of average case approximation ratio (or approximability) that we will use throughout this thesis. First, we recollect the definition of worst-case approximation ratio:

Definition 2.1 (Worst-case Approximability)**.** *Given a minimization problem Π with set of instances $\mathcal{I}$ of size n, we say than a randomized algorithm $\mathcal{A}$ has worst-case approximation ratio at most $\alpha(n)$ if and only if*

$$\max_{I \in \mathcal{I}} \frac{\mathbb{E}_{\mathcal{A}}[\mathcal{A}(I)]}{OPT(I)} \leq \alpha(n)$$

where the expectation $\mathbb{E}_{\mathcal{A}}[\cdot]$ is over the randomness in the algorithm.

Given a distribution $\mathcal{D}$ over the instances of a problem Π, we will say an algorithm $\mathcal{A}$ has average approximation ratio α for $(\Pi, \mathcal{D})$ if $\mathcal{A}$ achieves an α approximation with high probability over the distribution $\mathcal{D}$. More formally,

Definition 2.2 (Average-case Approximability)**.** *Given a minimization problem* Π *and a distribution* $\mathcal{D}$ *over its instances of size* n*, we say than an algorithm* $\mathcal{A} \equiv \{\mathcal{A}_\gamma\}_\gamma$ *has average approximation ratio* $\alpha(n)$ *if and only if for every* $\gamma > 0$ *there exists a constant* $c(\gamma) > 0$ *s.t.*

$$\Pr_{I \leftarrow \mathcal{D}} \left(\frac{\mathcal{A}_\gamma(I)}{OPT(I)} > \alpha(n) \right) < \frac{c(\gamma)}{n^\gamma}.$$

In the above definition $\{\mathcal{A}_\gamma\}_\gamma$ is a sequence of algorithms parameterized by γ, we will hereby use a single algorithm $\mathcal{A}$ to refer to it (for an appropriately large γ). Since we will be concerned with randomized algorithms, we will take into account the expected performance of the algorithm over its own random bits:

Definition 2.3 (Average-case Approximability for Randomized Algorithms)**.** *Given a minimization problem* Π *and a distribution* $\mathcal{D}$ *over its instances of size* n*, we say than a randomized algorithm* $\mathcal{A}$ *has average approximation ratio* $\alpha(n)$ *if and only if for every* $\gamma > 0$ *there exists a constant* c_γ *s.t.*

$$\Pr_{I \leftarrow \mathcal{D}} \left(\frac{\mathbb{E}_{\mathcal{A}} [\mathcal{A}(I]}{OPT(I)} > \alpha(n) \right) < \frac{c_\gamma}{n^\gamma}$$

where the inner expectation $\mathbb{E}_{\mathcal{A}}[\cdot]$ *is over the randomness in the algorithm.*

Since for the problems we deal with in this thesis, the solution cost can be bounded at most a polynomial in the size of the input, this definition is strictly stronger than a bound on

$$\mathbb{E}_{I \leftarrow \mathcal{D}} [\mathcal{A}(I)/OPT(I)]$$

which is a non-robust quantity, that may not compose well under reductions.

2.3 Graph Partitioning

2.3.1 Partitions

Definition 2.4. *Let V be a set of vertices. We say that $\mathcal{P}$ is a partition of V into disjoint sets or simply partition, if $V = \bigcup_{P \in \mathcal{P}} P$ and every two $P', P'' \in \mathcal{P}$ are disjoint. For every vertex $u \in V$, denote by $\mathcal{P}(u)$ the unique set $P \in \mathcal{P}$ containing u.*

Denote by $I_S : V \to \{0, 1\}$ the indicator function of the set $S \subset V$:

$$I_S(u) = \begin{cases} 1, & \text{if } u \in S; \\ 0, & \text{otherwise.} \end{cases}$$

Definition 2.5. *Let $G = (V, E)$ be a graph and $\mathcal{P}$ be a partition of V. Define the set of edges cut by the partition as follows*

$$\operatorname{cut}(\mathcal{P}, E) \equiv \{(u, v) \in E : \mathcal{P}(u) \neq \mathcal{P}(v)\}.$$

The cost of the cut equals the size of the set $\operatorname{cut}(\mathcal{P}, E)$*:*

$$\operatorname{cost}(\mathcal{P}, E) \equiv |\operatorname{cut}(\mathcal{P}, E)|.$$

The cost of the cut restricted to a subset $\mathcal{O} \subseteq V$ only considers those edges which are incident on $\mathcal{O}$:

$$\begin{aligned} \operatorname{cost}_{|\mathcal{O}}(\mathcal{P}, E) &\equiv |\{(u, v) \in \operatorname{cut}(\mathcal{P}, E) : u \in \mathcal{O} \text{ or } v \in \mathcal{O}\}| \\ &\equiv \sum_{(u,v) \in \operatorname{cut}(\mathcal{P}, E)} \max\{I_{\mathcal{O}}(u), I_{\mathcal{O}}(v)\}. \end{aligned}$$

2.3.2 Graph Expansion

There are different, yet related notions of graph expansion which we will use throughout this thesis. The most commonly used notion is that of Combinatorial Expansion (simply referred to as expansion usually):

Definition 2.6 (Combinatorial Expansion)**.** *The expansion $h(G)$ of the graph $G = (V, E)$ is given by*

$$h(G) \equiv \min_{\substack{S \subset V \\ 0 < |S| \leq \frac{1}{2}|V|}} \frac{E(S, V \setminus S)}{|S|}.$$

A related quantity that is closely related to expansion is the conductance.

Definition 2.7 (Conductance)**.** *The conductance $\Phi(G)$ of the graph $G = (V, E)$ is given by*

$$\Phi(G) \equiv \min_{\substack{S \subset V \\ 0 < E(S,S) \leq \frac{1}{2}|E|}} \frac{E(S, V \setminus S)}{E(S, V)}.$$

For d-regular graphs, conductance and expansion are equivalent to each other, and are used interchangeably. We now define an algebraic quantity associated with graphs which will be crucial for our algorithmic guarantees: algebraic expansion.

Definition 2.8 (Normalized Laplacian)**.** *Let G be a graph on n vertices with adjacency matrix $A(G)$. Let D be the diagonal matrix composed of the degrees $d_1, d_2, \ldots, d_n$. Then the normalized Laplacian matrix $\mathcal{L}$ is defined by $\mathcal{L} = D^{-1/2}(D - A)D^{-1/2}$.*

Definition 2.9 (Algebraic Expansion)**.** *Let G be a graph on n vertices and let $\mathcal{L}$ be its corresponding normalized laplacian matrix. G is said to have algebraic expansion λ_G iff $\lambda_G = \lambda_2$ where $\lambda_1 = 0, \lambda_2, \ldots, \lambda_n$ are the eigenvalues of $\mathcal{L}$ in increasing order.*

For d-regular graphs with adjacency matrix A, $\lambda_G = \frac{\lambda_1(A) - \lambda_2(A)}{d}$.

Cheeger's inequality gives a means of relating algebraic expansion and conductance (combinatorial expansion):

Lemma 2.10 (Cheeger's Inequality)**.** *[2, 49] For a graph G on n vertices with conductance $\Phi_G \in [0,1]$ and algebraic expansion $\lambda_G \in [0,1]$, we have*

$$\frac{\lambda_G}{2} \leq \Phi_G \leq \sqrt{2\lambda_G}$$

2.3.3 Partitioning Problems

We now define three of the graph partitioning problems that we study in this thesis.

Definition 2.11. (Balanced Cut) *Given a graph $G = (V, E)$, the aim is to find a partition $\mathcal{P}(P_1, P_2)$ of V with $|P_1| = |P_2| = n/2$ which minimizes* $\mathrm{cut}(\mathcal{P}, E)$.

A constant factor approximation algorithm finds a partition $\mathcal{P}'(P_1', P_2')$ with $|P_1'|, |P_2'| \leq \beta n/2$ for some fixed constant $1 \leq \beta < 2$, such that $\mathrm{cost}(\mathcal{P}', E) \leq O(1)\,\mathrm{cost}(\mathcal{P}^*, E)$*, where $\mathcal{P}^*$ is an optimal balanced cut*[1].

Definition 2.12. (Small Set Expansion) *Given a graph $G = (V, E)$ and a parameter $\rho \in (0, 1/2]$, the aim is to find a partition $\mathcal{P}(P_1, P_2)$ of V with $|P_1| = \rho n$ that minimizes* $\mathrm{cost}(\mathcal{P}, E)$*. We will also be concerned with constant factor approximations (defined like in Balanced Cut).*

Definition 2.13. (Multicut) *Given a graph $G = (V, E)$ and a set of terminal pairs $\{(s_i, t_i)\}_{1 \leq i \leq r}$, the aim is to find a partition $\mathcal{P}$ of V that separates all terminal pairs s_i,t_i (i.e., for all i, $\mathcal{P}(s_i) \neq \mathcal{P}(t_i)$) and minimizes* $\mathrm{cost}(\mathcal{P}, E)$.

2.4 Approximation Stability and Relations to Combinatorial Expansion

Here, we give a formal definition of the notion Approximation Stability we use, and show how for graph partitioning, it reduces to a condition about expansion in the

[1] This is sometimes referred to as $O(1)$ pseudo-approximation [13].

planted instance. Since we will only be concerned about Approximation Stability for graph partitioning problems with two clusters, we will use $\mathcal{P} = (P_1, P_2)$ to represent the planted partition. Note that this is slightly different from the notion used in [17] for clustering problems (for instance, their target clustering could be different from the optimal solution).

Definition 2.14 (Approximation Stability for Graph Partitioning)**.** *An instance $\mathcal{I}$ is said to be c-approximation stable with planted partition $\mathcal{P} = (P_1, P_2)$ if and only if for every S such that $|S| \leq n/2$ and $\min_i |P_i|/2 \leq |S| \leq 2\min_i 2|P_i|$ we have that*

$$\text{cost}\left((S, V \setminus S), E\right) > \text{cost}(\mathcal{P}, E) + c \cdot \min_{i \in \{1,2\}} \{|S \triangle P_i|\}$$

where $\min_i\{|P_i \triangle S|\}$ represents how much the two partitions differ.

Lemma 2.15 (Stability implies Expansion)**.** *In the notation of the Def. 2.14, if G is c-approximation stable instance with partition $\mathcal{P} = (P_1, P_2)$ s.t. $|P_1| < |P_2|$, then the subgraphs induced by clusters $G_{|P_i}$ have combinatorial expansion at least c i.e. $h(G_{|P_1}) > c$*

Proof. Let $|P_1| < |P_2|$ without loss of generality. Consider a set $T \subseteq P_1$, with $T \leq |P_1|/2$. Consider the partition given by $S = P_1 \cup T$. By approximation stability,

$$\text{cost}(\mathcal{P}, E) + c|P_1 \setminus S| < E(S, V \setminus S) = \text{cost}(\mathcal{P}, E) + E(S, T)$$

But, since $T = P_1 \setminus S$, we have that $|E(T, P_1 \setminus T)| > c|T|$ as required. □

In fact, it is not hard to see that the converse is also true. Hence, this notion of approximation stability is equivalent to expansion of the subgraphs given by the clusters.

2.5 Densest k-subgraph

Definition 2.16 (Densest k-subgraph)**.** *Given a graph G and a parameter k, find the subgraph of G on at most k vertices with the maximum number of edges inside it.*

The following conventions will be used in particular for the chapters 7,8,9 that deal with the Densest k-subgraph problem:

- D refers to the maximum degree of G.
- k refers to the size of the subgraph we are required to output.
- H will denote the densest k-subgraph (breaking ties arbitrarily) in G, and d denotes the average degree of H.

2.6 Prior Average-case models for Graph Partitioning

A natural average-case model for graph partitioning should try to capture its chief application in graph clustering. When a practioner tries to solve his or her problem through graph partitioning, there is a belief that the instance has a good partitioning solution. Such a good partitioning solution would have much fewer edges between the different clusters, than inside the various clusters. The different average-case models we will encounter try to capture this property, to varying degrees of generality.

Over the last couple of decades, there has been considerable interest in studying average-case models of graph partitioning problems like Balanced Cut [26, 25, 35, 39, 70]. The average-case models they handle are called the (planted) Random model and the Semi-Random model.

2.6.1 Planted Random Models

The first random model, sometimes called the planted random model was introduced in 1984 by Bui, Chaudhuri, Leighton and Sipser [26]. Here, the graph is generated by picking every edge independently at random, in such a way there is an obvious solution – we call this the *planted solution*. This planted solution is the intended clustering (partitioning) and the edge probability inside clusters is more than the edges probability between different clusters. For the Balanced Cut problem, we generate a graph on a set V of size n as follows:

Definition 2.17 (Planted Random model)**.** *We are given a set of n vertices V, and an arbitrary bipartition of the vertices into two equal parts S and $V \setminus S$. The graph is generated as follows:*

- *We sample edges between S and $V \setminus S$ with probability ε_1 independently at random.*
- *We sample edges within S and every edge within $V \setminus S$ with probability $\varepsilon_2 > \varepsilon_1$, independently at random.*

Note that all choices in the planted random model are random (there are no adversarial choices), so the model describes a probability distribution on graphs.

Results in the Planted Random model The model attracted a lot of attention and was studied in a series of papers by Dyer and Frieze [36], Boppana [25], Jerrum and Sorkin [53], Dimitriou and Impagliazzo [35], Condon and Karp [34] and Coja-Oghlan [33]. These papers explored several techniques for solving the problem — flow-based, combinatorial, spectral techniques, simulated annealing and go-with-the-leader technique. The algorithm of Boppana [25] finds the planted bisection $(S, V \setminus S)$ w.h.p. if $\varepsilon_2 - \varepsilon_1 > C\sqrt{\varepsilon_2 \log n/n}$. Later McSherry [70] obtained similar results for a more general class of graph partitioning problems, where there could be a partition

of V into many clusters (instead of two), with (possibly) different edge probabilities between different pairs of clusters.

Coja-Oghlan [33] extended the result of Boppana to the case when $\varepsilon_2 - \varepsilon_1 > C(\frac{1}{n} + \sqrt{\varepsilon_2 \log(n\varepsilon_2)/n})$. Note that if $\varepsilon_2 - \varepsilon_1 = o(\sqrt{\varepsilon_2 \log n/n})$ then the random graph has exponentially many minimum bisections and the planted bisection is not a minimum bisection w.h.p. [33]. The algorithm of Coja-Oghlan finds a minimum bisection rather than the planted bisection w.h.p.

The techniques used in these algorithms typically use various statistical properties of $\mathcal{G}(n,p)$ random graphs. For instance, the spectral algorithms of Boppana[25] and [70] use the fact that all of the eigenvalues (except the top one) of $\mathcal{G}(n,p)$ graphs are bounded by $O(\sqrt{np})$ with high probability. These techniques are not very robust to noise, and are specific to this distribution. This constitutes the main criticism against these models — they typically assume too much independence. For instance, every edge is chosen independently at random (even inside the clusters of the partition).

2.6.2 Semi-random model of Feige-Kilian

In 2000, Feige and Kilian [39] proposed a more flexible *semi-random* model. The model adds an extra post-processing step to the random planted model: after a random graph is generated, the adversary may delete edges between S and $V \setminus S$ and add new edges within S and within $V \setminus S$.

Definition 2.18 (Semirandom model of Feige-Kilian[39])**.** *The graph G is generated by a process which involves both a random step, and an adversarial step:*

- *Generate G_1 according to the Planted Random model (Def. 2.17).*
- *Adversary deletes arbitrary edges from G_1 between S and $V \setminus S$.*
- *Adversary adds new edges to within S and within $V \setminus S$.*

Semi-random instances of Feige and Kilian can have much more structure than random planted instances. Therefore, the model arguably captures real–world instances much better than the random model. From an algorithmic point of view, an important difference is that algorithms for the semi-random model of Feige and Kilian cannot overly exploit statistical properties of random graphs. In particular, spectral algorithms do not work for this model. Feige and Kilian [39] developed an SDP algorithm that finds the planted bisection if $\varepsilon_2 - \varepsilon_1 > C\sqrt{\varepsilon_2 \log n/n}$ (matching the bound of Boppana [25]). The SDP-based algorithm of [39] can be seen as a more robust form of the spectral algorithm of [25]. More technically, the eigenvalue gap for random graphs is used to provide a dual certificate to show that the semi-definite program captures the value of the balanced cut.

Chapter 3

Our Planted Models for Graph Partitioning

In this thesis, we study two planted models for graph partitioning, that lie between the worst-case and the basic average-case models (planted random model) in their generality. The first one is a semi-random model that generalizes all previous random models for graph partitioning. It allows the adversary complete power inside the clusters of the planted partition, and hence, seems to better capture properties of real-world instances. The second model enforces a stability assumption about the optimality of the planted partition. This is implemented by placing a condition on the (spectral) expansion inside the clusters of the planted partition. We now proceed to the description of the two models.

3.1 Semi-random Models for Graph Partitioning

Before we proceed with the formal presentation of our model, let us discuss what we can reasonably assume about real-world instances. In a graph partitioning problem, the goal is to divide graph vertices into several parts, or clusters, so as to minimize the number of cut edges (subject to constraints that depend on a specific problem).

A natural average-case model for graph partitioning should try to capture its chief application in graph clustering. Since, there is a belief that the instance has a good partitioning solution, this solution would have much fewer edges between the different clusters, than inside the various clusters.

When a practitioner solves a graph partitioning problem, she usually expects that the problem has a good solution — she believes that there is some underlying reason why there should be very few edges between clusters. That is, a real-world process that "generates" the graph instance adds an edge between clusters only when some random unexpected event happens. Therefore, in our opinion, it is reasonable to assume that edges between clusters are added at random. However, we cannot in general assume anything about edges within clusters (since their absence or presence does not affect the size of the cut between clusters). One could also view these edges between the clusters as random noise in an otherwise perfect clustering (partitioning). Additionally, in our model we assume that some random edges between cluster might be removed by the adversary (this assumption makes the model more robust).

This discussion leads to the following informal definition of semi-random instances: consider a set of vertices V and some clustering of V. A semi-random graph G on V is a graph with *arbitrary (adversarial)* edges inside clusters and *random* edges between clusters (more generally, the set of edges between clusters might be a subset of a random set of edges).

Before, we formally define the semi-random model, we refresh some basic notations from Section 2.3.1.

Definition 3.1. *Let V be a set of vertices. We say that $\mathcal{P}$ is a partition of V into disjoint sets or simply partition, if $V = \bigcup_{P \in \mathcal{P}} P$ and every two $P', P'' \in \mathcal{P}$ are disjoint. For every vertex $u \in V$, denote by $\mathcal{P}(u)$ the unique set $P \in \mathcal{P}$ containing u.*

Denote by $I_S : V \to \{0,1\}$ *the indicator function of the set* $S \subset V$*:*

$$I_S(u) = \begin{cases} 1, & \text{if } u \in S; \\ 0, & \text{otherwise.} \end{cases}$$

Definition 3.2. *Let* $G = (V, E)$ *be a graph and* $\mathcal{P}$ *be a partition of* V*. Define the set of edges cut by the partition as follows*

$$\operatorname{cut}(\mathcal{P}, E) \equiv \{(u,v) \in E : \mathcal{P}(u) \neq \mathcal{P}(v)\}.$$

The cost of the cut equals the size of the set $\operatorname{cut}(\mathcal{P}, E)$*:*

$$\operatorname{cost}(\mathcal{P}, E) \equiv |\operatorname{cut}(\mathcal{P}, E)|.$$

The cost of the cut restricted to a subset $\mathcal{O} \subseteq V$ *only considers those edges which are incident on* $\mathcal{O}$*:*

$$\begin{aligned} \operatorname{cost}_{|\mathcal{O}}(\mathcal{P}, E) &\equiv |\{(u,v) \in \operatorname{cut}(\mathcal{P}, E) : u \in \mathcal{O} \text{ or } v \in \mathcal{O}\}| \\ &\equiv \sum_{(u,v) \in \operatorname{cut}(\mathcal{P}, E)} \max\{I_{\mathcal{O}}(u), I_{\mathcal{O}}(v)\}. \end{aligned}$$

Our semi-random model for graph partitioning problems is the following:

Definition 3.3. (Semi-random Model for Graph Partitioning)
Consider a set of vertices V *and a partition of vertices into disjoint sets* $\mathcal{P}$*. Let* $E_K = \{(u,v) : \mathcal{P}(u) \neq \mathcal{P}(v)\}$ *be the set containing all vertex-pairs crossing partition boundaries. Let* $\widetilde{E_K} = \{(u,v) : \mathcal{P}(u) = \mathcal{P}(v)\}$ *be the set containing all vertex-pairs not crossing partition boundaries. (Thus,* $(V, E_K \cup \widetilde{E_K})$ *is the complete graph on* V*.) Consider a random subset of edges* E_R *of the set* E_K*: each edge* $(u,v) \in E_K$ *belongs to* E_R *with probability* ε *and these choices are independent. We define a random set*

of graphs $SR(\mathcal{P}, \varepsilon)$ as follows:

$$SR(\mathcal{P}, \varepsilon) = \{G = (V, E) : E \subseteq E_R \cup \widetilde{E_K}\}.$$

The optimal cost of the semi-random partition is defined as

$$\text{sr-cost}(\mathcal{P}, \varepsilon) = \mathbb{E}|E_R| = \varepsilon|E_K|.$$

To give an example, the corresponding semi-random model for Balanced Cut is as follows:

Definition 3.4. (Semi-random Model for Balanced Cut)
We are given a set V of n vertices, and a parameter ε. In our model, a semi-random graph G is generated as follows.

1. *The adversary chooses a subset $S \subset V$ of $n/2$ vertices.*
2. *The nature chooses a set of random edges E_R between S and $V \setminus S$ and adds edges from E_R to G. For every $u \in S$ and $v \in V \setminus S$, the edge (u, v) belongs to E_R with probability ε; choices for all edges (u, v) are independent.*
3. *The adversary arbitrarily adds edges within S and within $V \setminus S$.*
4. *The adversary deletes some edges between S and $V \setminus S$.*

Aim: *The performance of the algorithm is measured by comparing the cost of edges cut to the expected number of edges in E_R (the set of edges chosen at step 2 above).*

Note that the guarantees are not w.r.t the size of the cut $(S, V \setminus S)$ after step 4 or with the size of the optimal balanced cut. This is essential, since for example for $\varepsilon = 1$, $E_R = S \times (V \setminus S)$ the adversary can choose any graph G; so if we compared the cost of the cut with the cost of the optimal cut, our model would be the worst case model.

Justification of our model The random edges across clusters reflect the underlying belief of the practioner — correlations between different clusters are unexpected (can be viewed as random noise). Since the correlations between items in a cluster can be arbitrary, our model is more realistic than previous models.

Consider a toy example that illustrates why we believe that real-world instances are well described by our model. Suppose that we run a wiki website (or online store, online catalog etc). We track what pages our visitors read and construct a graph G on the set of all wiki pages V (see e.g., [72, 52]). If a visitor goes from page A to page B, we connect A and B with an edge. What is the structure of this graph? We expect that a visitor will read one article, then read an article that explains some term mentioned in the first one, then read another article related to the second one and so on. Sometimes, of course, the visitor will move to a completely unrelated article on a different subject. Consequently, there will be two types of edges in our graph — edges between pages on the same subject, and edges between pages on different subjects. Edges of the first type are not random and show real connections between related articles. However, edges of the second type are essentially random. Say, an edge between articles "Ravioli" and "Register Allocation" is likely to be completely random and does not show any connection between articles; it just happened that the visitor first read an article about ravioli and then decided to read an article on register allocation; in contrast, an edge between articles "Ravioli" and "Dumplings" is not random and shows a real connection between these food items. To summarize, in our example

- edges between pages on one subject are not random (i.e. edges within a cluster);
- edges between pages on different subjects are random (i.e. edges between clusters).

So G is a semi-random graph according to our model.

3.2 Planted Spectral Expander Model.

In their recent work, Balcan, Blum and Gupta [17] argued that in many clustering applications, various objective functions are only used as a proxy for its closeness to a "ground-truth clustering", which is intended to be the optimal solution. With this motivation, they showed that various clustering problems like k-means and k-median have PTAS under some assumptions about the stability of the optimal solution. The main challenge left open by their work was whether such stability assumptions could be used to attain $O(1)$ approximations to graph partitioning problems. Note that such stability assumptions correspond to a condition about the expansion of the clusters inside the planted partition (see Lemma 2.15 for the exact correspondence).

With this motivation, in Chapter 6 we study graph partitioning problems on instances with a planted partition, where one of clusters of the planted partition is a spectral (algebraic) expander. Consider a graph G with a (planted) balanced cut $(S, V \setminus S)$. A graph satisfying this model satisfies that the normalized algebraic expansion of the induced graph $G[S]$ is greater than the combinatorial expansion $h_{(S,V\setminus S)}$ of the cut by some constant factor.

Definition 3.5 (Planted Spectral Expander)**.** *We are given a graph $G = (V, E)$ on n vertices with a "planted" bisection $\mathcal{P} = \{P_1, P_2\}$ with $|P_1| \leq |P_2|$ (not known to the algorithm) of cut value at most εm and a parameter $C' > 1$. G is a Planted Spectral Expander if it satisfies:*

- *$E_1 \subset E(G_{|P_1})$ such that $|E_i| = m$.*
- *the graph $G' = (P_1, E_1)$ is a regular expander with a (normalized) algebraic expansion[1] $\lambda(G') > C'\varepsilon$.*

Note that for Balanced Cut, having this condition for any one of the two pieces suffices. This is a much weaker condition than edges of $E(S, S)$ or $E(V \setminus S, V \setminus S)$

[1] eigenvalue gap of the normalized Laplacian (see Def. 2.9)

being chosen independently at random. More crucially, in this case, these edges can be arbitrary.

The famous Cheeger's inequality (Lemma 2.10) relates the algebraic expansion to combinatorial expansion: when the combinatorial expansion of the subgraphs $G_{|P_1}$ or $G_{|P_2}$ is large enough (stability), then they have sufficient algebraic expansion to fit into our setting.

Chapter 4

Algorithms for Semi-Random Graph Partitioning: Balanced Separator

In this chapter, we describe our main algorithmic framework for average-case instances of Graph Partitioning problems like Balanced Cut and Multicut. We show how these algorithms, which are based on new semi-definite programming (SDP) techniques can achieve a $O(1)$ bicriteria approximation for the semi-random instances we introduced in Section 3.1. The ideas that we develop in this chapter will be generalized further to achieve similar guarantees for the Small Set Expansion and Sparsest Cut problems.

We first give a very brief and informal outline of our approach. The core of our algorithms is a *Hidden Solution Sparsification* step (HSS). This step is the same in all our algorithms. Let E_R represent the edges between clusters, which have been generated by the random process. As described in Chapter 3, to achieve a $O(1)$ factor approximation in this semi-random model, we need to find a partition which cuts at most $O(|E_R|)$ edges. For sake of convenience, let us think of E_R (which is unknown to us, of course) as constituting the optimal solution OPT. Intuitively,

the goal of the Hidden Solution Sparsification step is to find and remove almost all edges from E_R (all but $|E_R|/polylog(n)$ of them) by removing at most $O(OPT)$ edges. On the remaining instance, we can run worst-case algorithms with poly-logarithmic approximation guarantees to get a $O(1)$ approximation overall.

This core algorithm tries to identify the random edges E_R, by using a semi-definite programming relaxation, that defines a metric over the vertices. The crucial insight comes from a structural theorem, which establishes that most random edges in a semi-random instance are long according to the metric given by a well-behaved high-dimensional SDP solution. This we refer to as *Geometric Expansion* of the edges E_R (see Def. 4.5 for a formal definition). Geometric Expansion of E_R guarantees that in a "well-behaved" SDP solution, at most an $O(\delta^2)$ fraction of edges from E_R are shorter than δ for length scales $\delta \in \{2^{-t} : t = 1, 2, \ldots, \log\log n\}$. This core procedure will run over $O(\log\log n)$ phases (corresponding to these $O(\log\log n)$ length scales) — in each phase, cutting long edges (corresponding to the scale) succeeds in removing a constant fraction of the remaining random edges E_R. This successfully "sparsifies" the edges E_R by a poly-logarithmic factor.

To understand the Geometric Expansion property, let us compare this to the local-global phenomenon in [11]. Cheeger's inequality relates global correlations to local correlations in algebraic expanders: this implies that the SDP assigns on average a length of 1/4 to the edges (we in fact use this property in Chapter 6). However, this only helps in removing a constant fraction of the random edges E_R (by cutting edges of length $\geq 1/4$). However, we need to remove *all but* $|E_R|/\text{poly}(\log n)$ random edges. While we might try to achieve this by recursing on the remaining instance and repeating this procedure $O(\log\log n)$ times, this recursion fails because we lose the randomness of the edges E_R after removing some edges in the first step. Geometric Expansion can thus be seen as a stronger property over multiple length-scales, which

includes the local-global property of [11] in random graphs at one scale (of length $\Omega(1)$).

To show this Geometric Expansion property for semi-random instances, we first show that the set of feasible solutions to an SDP relaxation that imposes a metric on the vertices (ℓ_2^2 metrics) can be "approximated" by a small family of representative SDP solutions. For a fixed well-behaved SDP solution (one where the geometric neighborhood of every vertex is not abnormally large for a high-dimensional solution), we show that geometric expansion is satisfied with very large probability. Finally, using the union bound and the fact that there are much fewer representative SDP solutions than semi-random instances, we prove that every well-behaved feasible SDP solution has the required geometric expansion property w.h.p. However, we can not expect that *every* feasible solution satisfies this property — for instance, the intended SDP solution is low-dimensional and assigns all vertices to one of two fixed vectors. Our algorithm will have a sub-routine (called Heavy Vertices Removal Procedure) which will take advantage of such "low-dimensional" solutions to find a cheap partitioning solution. We now describe our algorithmic framework in more details.

The algorithmic framework: an informal description.

The HSS procedure finds a set of edges E^- and divides the graph $G - E^-$ into a set M and a number of sets Z_i such that:

1. The cost of the optimal solution for the sub-instance on $G[M] - E^-$ is at most $OPT/\log^{O(1)} n$.

2. Roughly speaking, each Z_i does not have to be further partitioned. Formally, we call this condition Φ-feasibility. Say, for the Balanced Cut problem, this condition means that each set Z_i contains at most cn vertices (for $c < 1$); for Multicut, it means that each Z_i contains at most one terminal from each source terminal pair.

3. All edges between M and Z_i and between sets Z_i lie in E^-.

4. There are "few" edges in E^-. For Balanced Cut and Multicut, we require that $|E^-| < O(OPT)$; for Small Set Expansion we have a more involved condition.

Then we run an existing $\log^{O(1)} n$-approximation algorithm for the sub-instance on the graph $G[M] - E^-$ (e.g. run the algorithm of Arora, Rao and Vazirani [13] for Balanced Cut). The first condition guarantees that the algorithm finds a partition $\{M_i\}$ of M of cost $O(OPT)$. We consider the combined partition $\{M_i, Z_j\}$ of V. When we solve Balanced Cut or Multicut, the total number of edges cut by this partition is $O(OPT)$. We join together some sets in $\{M_i, Z_j\}$ and obtain a feasible solution of cost $O(OPT)$ (this step depends on the problem at hand).

Hidden Solution Sparsification. Let us now focus on the Balanced Cut problem, to illustrate our approach. We find the partition $M, \{Z_i\}$ as follows. We start with the trivial partition $M = V$ and then iteratively cut sets Z_i from M. Once we cut a set Z_i, we do not further subdivide it. We ensure that after t rounds the cost of the optimal solution for the sub-instance on $G[M]-E^-$ is $O(OPT/2^t)$ and that properties (2)–(4) hold. Then after $O(\log \log n)$ iterations, we get the desired partitioning.

At iteration t, we solve the SDP relaxation for the problem on $G[M] - E^-$ and obtain an SDP solution $\varphi : M \to \mathbb{R}^n$ (the solution assigns vector $\varphi(u)$ to each vertex u). Since the cost of the optimal solution for $G[M] - E^-$ is $O(OPT/2^t)$, the cost of the SDP solution is also $O(OPT/2^t)$. The solution defines a metric $d(u, v) = \|\varphi(u) - \varphi(v)\|^2$ on the set M. We analyze the metric at scale $\delta_t = \delta_0/2^t$ (where $\delta_0 > 0$ is an absolute constant). For every vertex u, consider the set $B_u = \{v : d(u, v) \le \delta_t\}$ of vertices at distance at most δ_t from u. Let us say that a vertex u is δ_t-*light* if $|B_u| < \delta_t^2 n$, and that u is δ_t-*heavy* if $|B_u| \ge \delta_t^2 n$. Denote the set of heavy vertices by H and light vertices by L. Broadly speaking, we first use a procedure to remove the heavy vertices H (and further process them to get Φ-feasible sets Z_i), while cutting

only a few edges (these cut edges are added to E^-). In the remaining graph $G[M]-E^-$ all vertices are light. We show that in such a solution, at most an $O(\delta_t^2)$ fraction of edges from E_R are shorter than $\delta_t/2$. Here we crucially use that E_R is a random set of edges (and, thus, the graph $G = (V, E_R)$ is "geometrically expanding" according to Def. 4.5). We cut all edges in $G[M] - E^-$ longer than $\delta_t/2$ and add them to E^-. In the obtained graph $G[M] - E^-$ all edges are shorter than $\delta_t/2$, hence it contains at most $O(\delta_t^2 OPT)$ edges from E_R. Thus in the next iteration, the cost of the optimal solution for the sub-instance on $G[M] - E^-$ is $O(\delta_{t+1}^2 OPT)$ (as we need).

The Heavy Vertex Removal procedure finds new sets Z_i that cover all heavy vertices H in several rounds. In each round, we define a few sets Z_i; each Z_i contains a subset of heavy vertices together with their r-neighborhoods (where $r \in (\delta_t, 2\delta_t)$). We cut sets Z_i away, add edges from Z_i to the rest of the graph to E^- and then process remaining heavy vertices. We ensure that all sets Z_i have a small diameter and this implies that sets Z_i are Φ-feasible. We cut sets Z_i so that sets Z_i cut in one round are far away from each other and the total number of rounds is small. This guarantees that the total number of cut edges by this procedure is small (here, we use that each set Z_i contains a ball B_u for some heavy vertex u, and hence Z_i is not very small).

To upper bound the number of edges cut by removing edges longer than $\delta_t/2$, we observe that the SDP value in iteration t is $O(\delta_t^2 OPT)$. The number of these cut edges is $O(\delta_t^2 OPT/\delta_t) = O(\delta_t OPT)$. Thus, the number of edges cut in all iterations is $O(\sum_i \delta_i OPT) = O(OPT)$.

In the rest of the chapter, after introducing some basic notations, we first describe the main technical ingredient — Hidden Solution Sparsification subroutine, and then prove the structural theorem for semi-random instances which is crucial towards proving the required guarantees. We finally show how to obtain $O(1)$ approximation guarantees for Balanced Cut, Multicut and Minimum Uncut.

4.1 Preliminaries : Φ-feasibility, Heavy vertices and Geometric Expansion

Definition 4.1. *Let V be a set of vertices. We say that a map $\varphi : V \to \mathcal{H}$ is an SDP solution if vectors in $\varphi(V)$ satisfy ℓ_2^2–triangle inequalities: for every $u, v, w \in V$, $\|\varphi(u) - \varphi(v)\|^2 + \|\varphi(v) - \varphi(w)\|^2 \geq \|\varphi(u) - \varphi(w)\|^2$.*

For instance, solutions to the SDP relaxations in Chapters 4,5 (Figures 4.4,4.5,5.2) satisfy the above definition. The SDP solution φ coming from these relaxations define a metric on the vertices V (given by $\|\varphi(u) - \varphi(v)\|^2$ for $u, v \in V$) because of their triangle inequality constraints. The cost of a solution φ corresponds to the total length of edges according to metric given by φ (note that there are no constraints corresponding to the edges in these relaxations).

Definition 4.2. *Let $G = (V, E)$ be a graph, $\mathcal{P}$ be a partition of V, and $\mathcal{O}$ be a subset of V. Define the cost of an SDP solution $\varphi : V \to \mathcal{H}$ to be*

$$\text{sdp-cost}(\varphi, E) \equiv \frac{1}{2} \sum_{(u,v)\in E} \|\varphi(u) - \varphi(v)\|^2,$$

and the cost of the SDP solution restricted to the set $\mathcal{O}$ to be

$$\text{sdp-cost}_{|\mathcal{O}}(\varphi, E) \equiv \frac{1}{2} \sum_{\substack{(u,v)\in E \\ u\in\mathcal{O} \text{ or } v\in\mathcal{O}}} \|\varphi(u) - \varphi(v)\|^2.$$

Φ–feasibility captures sets that require no further processing, to belong to a solution. For example, Φ-feasible sets correspond to small enough sets for the Balanced Cut or Small Set Expansion problems, and to sets which do not contain any terminal pairs for the Multicut problem.

Definition 4.3. *Let V be a set of vertices and $\Phi \subset \{\varphi : V \to \mathcal{H}\}$ be a set of SDP solutions. We say that a subset $S \subset V$ is Φ–feasible if there exists $\varphi^* \in \Phi$ such that*

for every $u, v \in S$,

$$\|\varphi^*(u) - \varphi^*(v)\|^2 \leq \frac{1}{4}.$$

An SDP solution φ classifies the vertices into two types (heavy or light) depending on the number of vertices in their δ–neighborhoods.

Definition 4.4. *Let* V *be a set of* n *vertices, and* $M \subseteq V$*. Consider an SDP solution* $\varphi : V \to \mathcal{H}$*. We say that a vertex* $u \in M$ *is* δ*–heavy in* M *if the* ℓ_2^2*-ball of radius* δ *around* $\varphi(u)$ *contains at least* $\delta^2 n$ *vectors from* $\varphi(M)$ *i.e.,* $|\{v \in M : \varphi(v) \in \text{Ball}(\varphi(u), \delta)\}| \geq \delta^2 n$*. We denote the set of all heavy vertices by* $H_{\delta,\varphi}(M)$*.*

The following property of semi-random instances is crucially used in our algorithms.

Definition 4.5. (Geometric Expansion) *A graph* $G = (V, E)$ *satisfies the geometric expansion property with cut value* X *at scale* δ *if for every SDP solution* $\varphi : V \to \mathcal{H}$ *and every subset of vertices* $M \subseteq V$ *satisfying* $H_{\delta,\varphi}(M) = \varnothing$*,*

$$|\{(u, v) \in E \cap (M \times M) : \|\varphi(u) - \varphi(v)\|^2 \leq \delta/2\}| \leq 2\delta^2 X.$$

A graph $G' = (V, E')$ *satisfies the geometric expansion property with cut value* X *up to scale* 2^{-T} *(*$T \in \mathbb{N}$*) if it satisfies the geometric expansion property for every* $\delta \in \{2^{-t} : 1 \leq t \leq T\}$*.*

We can slightly simplify the definition[1] above by requiring that φ satisfies the condition $H_{\delta,\varphi}(V) = \varnothing$ and $M = V$. See Section 4.3 for details.

In section 4.3, we will see that in semi-random instances $SR(\mathcal{P}, \varepsilon)$, the graph consisting of the random edges (V, E_R) is geometrically expanding w.h.p. for sufficiently large ε.

[1]We note that every Ramanujan expander is geometrically expanding with some parameters. However, we omit the details here.

4.2 Hidden Solution Sparsification

The main technical ingredient of the algorithm is the Hidden Solution Sparsification step, which guarantees the following

Theorem 4.6. (Hidden Solution Sparsification)
There exists a polynomial-time randomized algorithm that given a graph $G = (V, E)$, a SDP relaxation Φ of a partition $\mathcal{P}$ (note: the partition $\mathcal{P}$ are "hidden" and are not known to the algorithm), and a parameter $D = 2^T$ ($T \in \mathbb{N}$, $T > 1$), partitions the set of vertices V into a set M and a collection of disjoint sets $\mathcal{Z}$

$$V = M \cup \bigcup_{Z \in \mathcal{Z}} Z,$$

and also partitions the set of edges into two disjoint sets E^+ and E^-

$$E = E^+ \cup E^-$$

such that

- *all edges cut by the partition $V = M \cup \bigcup_{Z \in \mathcal{Z}} Z$ lie in E^-, i.e.,*

$$E^+ \subset (M \times M) \cup \bigcup_{Z \in \mathcal{Z}} (Z \times Z);$$

- *if the graph $(V, \mathrm{cut}(\mathcal{P}, E))$ satisfies the geometric expansion property with cut value X up to scale $1/\sqrt{D}$, then then cost of the optimal solution in the remaining instance*

$$\mathbb{E}[\mathrm{cost}_{|M}(\mathcal{P}, E^+)] \leq C\ X/D; \tag{4.1}$$

(the expectation is taken over random bits of the algorithm) and the number of cut edges

$$|E^-| \leq C\ X; \tag{4.2}$$

- *each $Z \in \mathcal{Z}$ is Φ–feasible.*

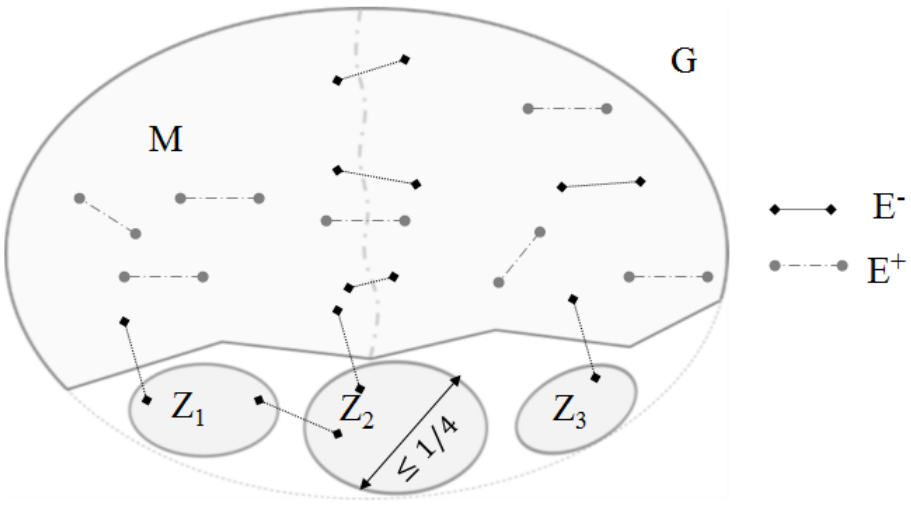

Figure 4.1: Output of the Hidden Solution Sparsification algorithms

Figure 4.1 on page 50 shows the output of the HSS algorithm in Fig. 4.2 on a graph G. The algorithm runs in $O(\log \log n)$ phases. In each round, we first solve the SDP on the current instance. The heavy vertices w.r.t. to this vector solution are first processed and removed using the algorithm from Section 4.2.1. In the remaining graph, we remove (cut) long edges to further "sparsify" the hidden solution (E_R) and produce the instance for the next phase.

Proof of Theorem 4.6. We analyze the algorithm given in Figure 4.2 on page 51. We note that the step A of finding φ_k can be performed in polynomial-time using semidefinite programming; the step B is performed using the algorithm described in the next subsection.

At every iteration, the algorithm removes all edges crossing the partition $\Delta\mathcal{Z}_t$ from E_t^+ and adds them to E_t^-, hence the first item of Theorem 4.6 holds. The third item holds, because every set $Z \in \mathcal{Z}$ belongs to some $\Delta\mathcal{Z}_t$ and, thus by Lemma 4.7 (see below), $\mathrm{diam}(\varphi_t(Z)) \leq 1/4$.

Hidden Solution Sparsification Algorithm

Input: a graph $G = (V, E)$ and a separation oracle for a set of SDP solutions $\Phi \subset \{V \to \mathcal{H}\}$.
Output: partitions $V = M \cup \bigcup_{Z \in \mathcal{Z}} Z$ and $E = E^+ \cup E^-$.

- Let $M_0 = V$, $\mathcal{Z}_0 = \varnothing$, $E_0^+ = E$, $E_0^- = \varnothing$, $T = \frac{1}{2}\log_2 D$, and $\delta_t = 2^{-t}$ for all $t = 1, \ldots, T$.
- for $t = 1, \ldots, T$ do

  A. Solve the SDP for the remaining graph: Find

  $$\varphi_t = \arg\min_{\varphi \in \Phi} \text{sdp-cost}(\varphi, E_{t-1}^+ \cap (M_{t-1} \times M_{t-1})).$$

  B. Remove δ_t–heavy vertices: run Heavy Vertices Removal Algorithm (described in Section 4.2.1) with parameters V, M_{t-1}, φ_t, and obtain a collection of Φ–feasible sets ΔZ_t. Add edges in E_{t-1}^+ cut by ΔZ_t to the set ΔE_t^-. Let

  $$\mathcal{Z}_t = \mathcal{Z}_{t-1} \cup \Delta\mathcal{Z}_t; \;\; M_t = M_{t-1} \setminus \bigcup_{Z \in \Delta\mathcal{Z}_t} Z.$$

  C. Remove δ_t–long edges from E^+: Find

  $$L_t = \{(u, v) \in E^+ : u, v \in M_t, \|\varphi_t(u) - \varphi_t(v)\|^2 \geq \delta_t\}.$$

  Let

  $$E_t^+ = E_{t-1}^+ \setminus (\Delta E_t^- \cup L_t); \;\; E_t^- = E_{t-1}^- \cup (\Delta E_t^- \cup L_t).$$

- **return** $M = M_T$, $\mathcal{Z} = \mathcal{Z}_T$, $E^+ = E_T^+$, $E^- = E_T^-$.

Figure 4.2: Algorithm for Hidden Solution Sparsification

We now show that the second item of Theorem 4.6 holds. We first prove that

$$\mathrm{cost}_{|M_t}(\mathcal{P}, E_t^+) \leq 2\ X \cdot \delta_t^2$$

for every $t \in \{0, \ldots, T\}$. The Heavy Vertices Removal Procedure returns set M_t that does not contain any δ_t–heavy vertices w.r.t. φ_t i.e., $H_{\delta_t,\varphi_t}(M_t) = \varnothing$ (see Lemma 4.7). Using the geometric expansion property of the graph $(V, \mathrm{cut}(E, \mathcal{P}))$, we get

$$\left|\{(u,v) \in \mathrm{cut}(\mathcal{P}, E) \cap (M_t \times M_t) : \|\varphi_t(u) - \varphi_t(v)\|^2 \leq \delta_t/2\}\right| \leq 2\delta_t^2 X.$$

The algorithm removes all $\delta_t/2$–long edges at step C, thus the set $E_t^+ \cap (M_t \times M_t)$ contains only edges (u, v) for which $\|\varphi_t(u) - \varphi_t(v)\|^2 \leq \delta_t/2$. Combining this observation with the previous inequality, and using that edges in E_t^+ do not cross the boundary of M_t, we get

$$\mathrm{cost}_{|M_t}(\mathcal{P}, E_t^+) = \left|\mathrm{cut}(E, \mathcal{P}) \cap E_t^+ \cap (M_t \times M_t)\right| \leq 2\delta_t^2 X. \tag{4.3}$$

For $t = T$, we get $\mathrm{cost}_{|M}(\mathcal{P}, E^+) \leq 2\ X/D$, as required.

Finally, to estimate the size of the set $|E^-|$, we use the fact that Φ is a relaxation of the partition $\mathcal{P}$. For graph $G = (V, E_{t-1}^+ \cap (M_{t-1} \times M_{t-1}))$, we obtain inequality

$$\begin{aligned}\text{sdp-cost}(\varphi_t, E_{t-1}^+ \cap (M_{t-1} \times M_{t-1})) &\leq C_1\, \mathrm{cost}_{|M_{t-1}}(\mathcal{P}, E_{t-1}^+) \leq 2CX \cdot \delta_{t-1}^2 \\ &= 8C_1\delta_t^2 X.\end{aligned}$$

The third line of the inequality follows from (5.3).

Now, we bound the number of edges removed from E_{t-1}^+ and added to E_t^- in terms of "sdp-cost". At step t, we add two sets of edges to E^-: the edges coming out of the cut-out pieces ΔE_t^- and the long edges L_t. Since all edges (u, v) in L_t are $\delta_t/2$–long

(i.e., $\|\varphi(u) - \varphi(v)\|^2 \geq \delta_t/2$),

$$\begin{aligned}\text{sdp-cost}(\varphi_t, E^+_{t-1} \cap (M_{t-1} \times M_{t-1})) &\equiv \sum_{(u,v)\in E^+_{t-1}\cap(M_{t-1}\times M_{t-1})} \frac{\|\varphi(u)-\varphi(v)\|^2}{2} \\ &\geq \frac{|L_t|\cdot\delta_t/2}{2}.\end{aligned}$$

Hence, $|L_t| \leq 32C_1\delta_t X$. The probability that the Heavy Vertices Removal Procedure separates two vertices u and v connected with an edges in E^+_{t-1} is at most (from Lemma 4.7)

$$C_2\big(\delta_t^{-1} + \delta_t^{-2}\mathbb{E}|M_{t-1} \setminus M_t|/n\big) \cdot \|\varphi(u) - \varphi(v)\|^2$$

Thus, the expected total number of edges in the set ΔE^-_t is at most

$$\begin{aligned}\mathbb{E}\left[|\Delta E^-_t|\right] &\leq C_2\Big(\delta_t^{-1} + \delta_t^{-2}\frac{\mathbb{E}|M_{t-1}\setminus M_t|}{n}\Big)\cdot \text{sdp-cost}(\varphi_t, E^+_{t-1}\cap(M_{t-1}\times M_{t-1})) \\ &\leq 8C_1C_2\Big(\delta_t + \frac{\mathbb{E}|M_{t-1}\setminus M_t|}{n}\Big)X.\end{aligned}$$

The total number of edges in E^- (in expectation) is bounded by

$$\begin{aligned}\mathbb{E}\left[|E^-|\right] &\leq \sum_{t=1}^{T}(32C_1\delta_t + 8C_1C_2\delta_t + 8C_1C_2\cdot\frac{\mathbb{E}|M_{t-1}\setminus M_t|}{n})X \\ &\leq (32C_1 + 8C_1C_2 + 8C_1C_2)X.\end{aligned}$$

□

4.2.1 Heavy Vertices Removal Procedure

In this section, we describe the algorithm which deals with the heavy vertices in a vector solution. Note that in the intended vector solution for all the above problems, all vertices are heavy (the intended solution for Balanced Cut has $n/2$ vectors at a fixed unit vector $\bar{v}_0$ and the rest $n/2$ of them at $-\bar{v}_0$). This algorithm also shows how

we can take advantage of vector solutions which look like the intended solution (say, roughly low-dimensional solutions), with many heavy vertices.

Lemma 4.7. *There exists a polynomial-time algorithm that given a set of vertices V, an SDP solution $\varphi : V \to \mathcal{H}$, a subset $M \subseteq V$, finds a set of vertices $M' \subset M$ and a partition of $M \setminus M'$ into disjoint sets $Z \in \Delta\mathcal{Z}$ such that*

- *the set M' does not contain any δ–heavy vertices w.r.t. φ (that is, $(H_{\delta,\varphi}(M') = \varnothing)$).*
- $\operatorname{diam}(\varphi(Z)) \le 1/4$ *for every $Z \in \Delta\mathcal{Z}$;*
- *for every two vertices u^* and v^*, the probability that u^* and v^* are separated by the partition is bounded as follows:*

$$\begin{aligned}\Pr(&\exists Z \in \Delta\mathcal{Z} \text{ s.t. } I_Z(u^*) \neq I_Z(v^*)) \\ &\le C\Big(\delta^{-1} + \delta^{-2}\,\frac{\mathbb{E}[|M \setminus M'|]}{n}\Big)\|\varphi(u^*) - \varphi(v^*)\|^2.\end{aligned}$$

We remark that some of the heavy vertices $H_{\delta,\varphi}(M)$ may belong to M', but they are not heavy anymore (w.r.t M').

Proof. We use the following algorithm. If $\delta \ge 1/32$, we run the algorithm with $\delta' = 1/32$.

Analysis. It is clear that the algorithm in Figure 4.3 on page 55 always terminates in polynomial-time (since at every step at least one vertex is removed). When the algorithm terminates $H_{\delta,\varphi}(M) = \varnothing$ by the condition of the "while" loop. Every set $\varphi(B_u)$ removed from M and added to $\Delta\mathcal{Z}$ at one of the iterations is contained in a ball of radius at most 2δ; every set $\varphi(B_U)$ is contained in the 2δ–neighborhood of a set $\varphi(U)$ (for some $U \in \mathcal{U}$) whose diameter is at most $1/8$. Thus, the diameter of each $\varphi(B_u)$ and $\varphi(B_U)$ is at most $1/8 + 4\delta \le 1/4$.

Heavy Vertices Removal Procedure

Input: a set of vertices V, a subset $M \subseteq V$, an SDP solution $\varphi : V \to \mathcal{H}$, a parameter $\delta \in (0, 1/32]$;
Output: a set $M \subseteq V$, partition $V \setminus M = \bigcup_{Z \in \Delta_{\mathcal{Z}}} Z$;

- while $(H_{\delta,\varphi}(M) \neq \varnothing)$
  - Connect heavy vertices in M at ℓ_2^2 distance at most 4δ with an edge and denote the new set of edges by $A = \{(u, v) \in H_{\delta,\varphi}(M) \times H_{\delta,\varphi}(M) : \|\varphi(u) - \varphi(v)\|^2 \leq 4\delta\}$.
  - Break graph $(H_{\delta,\varphi}(M), A)$ into connected components.
  - Pick a random $r \in [\delta, 2\delta)$.
  - *Remove components of small diameter:* For each connected component U with $\operatorname{diam}(\varphi(U)) \leq 1/8$, let

    $$B_U = \{v \in M : \exists u \in U \text{ s.t. } \|\varphi(u) - \varphi(v)\|^2 \leq r\}.$$

    Denote the set of all connected components of diameter at most $1/8$ by $\mathcal{U}$.
  - *Remove a maximal independent set:* In the remaining set $H_{\delta,\varphi}(M) \setminus \bigcup_{U \in \mathcal{U}} U$ find a maximal independent set[2] S. For each $u \in S$, let $B_u = \{v : \varphi(v) \in \operatorname{Ball}(u, r)\}$.
  - Remove sets B_U and B_u from M:

    $$M = M \setminus \Big(\bigcup_{U \in \mathcal{U}} B_U \cup \bigcup_{u \in S} B_u \Big);$$

- **return** $M' = M$.

Figure 4.3: Algorithm for Removing Heavy Vertices

Let us now verify the third item of Lemma 4.7. Fix two vertices u^* and v^*; and consider one iteration of the algorithm. We may assume that the algorithm first picks the independent set S and a collection of connected components $\mathcal{U}$, and only then chooses random $r \in [\delta, 2\delta)$. Observe, that the distance between (images of) any two vertices in S is at least 4δ (because S is an independent set), the distance between every two sets in $\mathcal{U}$ is at least 4δ (because every $U \in \mathcal{U}$ is a connected component), and the distance between every $U \in \mathcal{U}$ and $u \in S$ is at least 4δ (again because U is a connected component, and $u \notin U$). Thus, $\varphi(u^*)$ may belong to at most one $\mathrm{Ball}(U, 2\delta)$ or $\mathrm{Ball}(u, 2\delta)$. If $\varphi(u^*) \in \mathrm{Ball}(u, 2\delta)$, then

$$\Pr\left[\varphi(u^*) \in \mathrm{Ball}(u, r),\ \varphi(v^*) \notin \mathrm{Ball}(u, r)\right] \leq \delta^{-1}\|\varphi(u) - \varphi(v)\|^2.$$

Of course, if $\varphi(u^*) \notin \mathrm{Ball}(u, 2\delta)$, then

$$\Pr\left[\varphi(u^*) \in \mathrm{Ball}(u, r), \varphi(v^*) \notin \mathrm{Ball}(u, r)\right] \leq \Pr\left[\varphi(u^*) \in \mathrm{Ball}(u, r)\right] = 0.$$

The same statements hold if we replace $u \in S$ with $U \in \mathcal{U}$. Thus, at one iteration, the probability that u^* belongs to a removed ball but v^* does not belong to the same ball is at most $\delta^{-1}\|\varphi(u) - \varphi(v)\|^2$. Denote by T the number of iterations of the algorithm. Then, the probability that u^* and v^* are separated at one of the iterations is at most $2\delta^{-1}\mathbb{E}[T]\|\varphi(u) - \varphi(v)\|^2$.

We now prove that at every iteration but possibly the last, the algorithm removes at least δn vertices from M. Thus, $\mathbb{E}[T] \leq 1 + \mathbb{E}|M' \setminus M|/(\delta n)$, and the third item of Lemma 4.7 follows. Observe, that if the independent set $S = \varnothing$, then the algorithm terminates. If $S \neq \varnothing$, there exists at least one connected component L with $\mathrm{diam}(\varphi(L)) \geq 1/8$. The maximal independent set in L must contain at least $\Omega(\delta^{-1})$ vertices, since for every edge $(u, v) \in A$, $\|\varphi(u) - \varphi(v)\|^2 \leq 4\delta$. Thus, $|S| \geq \Omega(\delta^{-1})$. Since each $u \in S$ is δ–heavy and $r \geq \delta$, $|B_u| \geq \delta^2 n$. Hence (using the fact that sets

B_u are disjoint),

$$\left|\bigcup_{u\in S} B_u\right| = \sum_{u\in S} |B_u| \geq \delta n.$$

□

4.3 Structural Theorem

We now prove that semi-random graphs are geometrically expanding, namely we prove that with high probability for every semi-random graph $G = (V, E) \in SR(\mathcal{P}, \varepsilon)$ the graph $(V, \mathrm{cut}(\mathcal{P}, E))$ is geometrically expanding. This establishes that the conditions under which the guarantees of Theorem 4.6 hold, to be true for semi-random instances.

Theorem 4.8. *I. For every set of vertices V of size n, every partition $\mathcal{P}$, and every $\varepsilon \in (0, 1)$, $D = 2^T$ ($T \in \mathbb{N}$, $T > 1$), with high probability, the random set $SR(\mathcal{P}, \varepsilon)$ satisfies the following property: for every graph $G = (V, E) \in SR(\mathcal{P}, \varepsilon)$, the graph $(V, \mathrm{cut}(\mathcal{P}, E))$ is geometrically expanding with cut cost*

$$X = C \max\{\text{sr-cost}(\mathcal{P}, \varepsilon),\ nD(\log^2 D)\}$$

up to scale $1/\sqrt{D}$.

II. Moreover, a slightly stronger statement holds. For every set of vertices V of size n, every partition $\mathcal{P}$, and every $\varepsilon \in (0, 1)$, $D = 2^T$ ($T \in \mathbb{N}$, $T > 1$) with high probability, the random set $SR(\mathcal{P}, \varepsilon)$ satisfies the following property: for every graph $G = (V, E) \in SR(\mathcal{P}, \varepsilon)$ and every $U \subset V$, the graph $(U, \mathrm{cut}(\mathcal{P}, E) \cap (U \times U))$ is geometrically expanding with cut cost

$$X = C \max\{\text{sr-cost}(\mathcal{P}_{|U}, \varepsilon),\ nD(\log^2 D)\}$$

up to scale $1/\sqrt{D}$. *Here* $\mathcal{P}_{|U} = \{P \cap U : P \in \mathcal{P}\}$ *denotes the restriction of the partition* $\mathcal{P}$ *to the subset* U.

We defined Geometric Expansion in Section 4.1. We now give a slightly different definition of Geometric Expansion which is equivalent to Definition 4.5, but is more convenient for proving Theorem 4.8.

Definition 4.9. (Geometric Expansion; see Definition 4.5) *A graph* $G = (V, E)$ *satisfies the geometric expansion property with cut value* X *at scale* δ *if for every SDP solution* $\varphi : V \to \mathcal{H}$ *satisfying* $H_{\delta,\varphi}(V) = \varnothing$,

$$|\{(u, v) \in E : \|\varphi(u) - \varphi(v)\|^2 \leq \delta/2\}| \leq 2\delta^2 X.$$

A graph $G = (V, E)$ *satisfies the geometric expansion property with cut value* X *up to scale* 2^{-T} *(*$T \in \mathbb{N}$*) if it satisfies the geometric expansion property for every* $\delta \in \{2^{-t} : 1 \leq t \leq T\}$.

Claim 4.10. *Definitions 4.5 and 4.9 are equivalent.*

Proof. It is easy to see that every graph satisfying Definition 4.5 satisfies Definition 4.9: we simply let $M = V$. We show that the converse follows by a simple modification of the SDP solution. Assume that $G = (V, E)$ satisfies Definition 4.9. Consider an SDP solution $\varphi : V \to \mathcal{H}$ and a set M such that $\varphi_{\delta,\varphi}(M) = \varnothing$. Replace φ with φ': $\varphi'(u) = \varphi(u)$ if $u \in M$, and $\varphi'(u) = e_u$ otherwise, where $\{e_u\}_u$ is a collection of orthogonal unit vectors, orthogonal to all vectors $\varphi(u)$. The ℓ_2^2–distance between every vector $\varphi'(u) = e_u$ ($u \in V \setminus M$) and any other vector $\varphi'(v)$ is at least 1. Thus, $H_{\delta,\varphi}(M) \subset H_{\delta,\varphi}(V) = \varnothing$. Hence,

$$\begin{aligned}\big|\{(u, v) \in E \cap (M \times M) : \|\varphi(u) - \varphi(v)\|^2 \leq \delta/2\}\big| = \\ = \big|\{(u, v) \in E : \|\varphi'(u) - \varphi'(v)\|^2 \leq \delta/2\}\big| \leq 2\delta^2 X.\end{aligned}$$

□

Proof of Theorem 4.8. We use Definition 4.9 in this proof. Let $E_K = \{(u,v) \in V \times V : \mathcal{P}(u) \neq \mathcal{P}(v)\}$ and $E_R \subset E_K$ be the set of random edges chosen for the set $SR(\mathcal{P}, \varepsilon)$ as in Definition 3.3. Since $\text{cut}(\mathcal{P}, E) \subset E_R$ it suffices to show that the graph (V, E_R) is geometrically expanding with high probability. We fix the parameter $\delta = 2^{-t}$ (where $1 \leq t \leq T$), and prove that the graph (V, E_R) is geometrically expanding with cut value X at scale δ. Then we apply the union bound for all $T = \log_2 D$ possible choices of δ.

We use the technique developed by Kolla, Makarychev and Makarychev [61]. Observe that the condition $H_{\delta,\varphi}(V) = \varnothing$ implies that

$$\forall u \in V, \qquad |\{v \in V : \|\varphi(u) - \varphi(v)\|^2 \leq \delta\}| \leq \delta^2 n,$$

and, consequently,

$$|\{(u,v) \in V \times V : \|\varphi(u) - \varphi(v)\|^2 \leq \delta\}| \leq \delta^2 n^2.$$

Thus we need to bound the probability of the bad event: *there exists an SDP solution* $\varphi : V \to \mathcal{H}$ *such that*

$$|\{(u,v) \in V \times V : \|\varphi(u) - \varphi(v)\|^2 \leq \delta\}| \leq \delta^2 n^2 \tag{4.4}$$

and

$$|\{(u,v) \in E_R : \|\varphi(u) - \varphi(v)\|^2 \leq \frac{\delta}{2}\} \geq 2\delta^2 X. \tag{4.5}$$

We now show that if such φ exists then there exists an embedding $\varphi' : V \to N_\delta$ to a relatively small set $N_\delta \subset \mathcal{H}$ such that similar conditions hold. Then, we show

that these bounds do not hold with high probability, after a union bound over the all feasible solutions from the smaller set.

Easier bound : Geometric Expansion of $\max\{\text{sr-cost}(\mathcal{P}, \varepsilon),\ nD \log n(\log^2 D)\}$**.**

We use the following simple lemma proved in [61].

Lemma 4.11. (Lemma 3.7 [61], arXiv version) *For every positive ζ, η and ν, there exists a set N_δ of unit vectors of size at most*

$$\exp\left(O(\zeta^{-2}\log(1/\eta)\log(1/\nu))\right)$$

such that for every set of unit vectors Z there exists a randomized mapping $\psi : Z \to N$ satisfying the following property: for every $u, v \in \mathcal{Z}$,

$$\Pr\Big((1+\zeta)^{-1}\|u-v\|^2 - \eta^2 \le \|\psi(u)-\psi(v)\|^2 \le (1+\zeta)\|u-v\|^2 + \eta^2\Big) \ge 1-\nu. \tag{4.6}$$

The proof of Lemma 4.11 is based on the Johnson–Lindenstrauss lemma: The set N is an "epsilon–net" in a low dimensional space. To construct ψ we first project Z in a low dimensional space using the Johnson–Lindenstrauss transform and then "round" each vector to the closest vector in N. See [61] for details.

We now show that if such φ exists, then there exists an embedding $\varphi'' : V \to N_\delta$ to a relatively small set $N_\delta \subset \mathcal{H}$ satisfying

$$|\{(u,v) \in V \times V : \|\varphi''(u) - \varphi''(v)\|^2 \le \frac{3}{4}\,\delta\}| \le \delta^2 n^2, \tag{4.7}$$

and,

$$|\{(u,v) \in E_R : \|\varphi''(u) - \varphi''(v)\|^2 \le \frac{3}{4}\,\delta\}| \ge 2\delta^2 X. \tag{4.8}$$

We set parameters $\zeta = 1/7$, $\eta^2 = \delta/8$ and $\nu = 1/n^4$ and pick N_δ as in Lemma 4.11. Hence $N_\delta \subset \mathcal{H}$ is a set of size $\exp(O(\log n \log^2(1/\delta))$.

Then we choose a deterministic $\psi(u) : \varphi(V) \to N$ such that the condition

$$\begin{aligned}\frac{7}{8}\|\varphi(u) - \varphi(v)\|^2 - \frac{\delta}{8} &\leq \|\psi(\varphi(u)) - \psi(\varphi(v))\|^2 \\ &\leq \frac{8}{7}\,\|\varphi(u) - \varphi(v)\|^2 + \frac{\delta}{8}\end{aligned}$$

holds for all pairs $u, v \in V$ and all edges $(u, v) \in E_R$ w.h.p. Define $\varphi''(u) = \psi(\varphi(u))$. We get, with probability $1 - o(1)$:

- for all pairs $u, v \in V$, if $\|\varphi(u) - \varphi(v)\|^2 > \delta$, then $\|\varphi''(u) - \varphi''(v)\|^2 \geq 7\delta/8 - \delta/8 = 3\delta/4$;
- for all edges $(u, v) \in E_R$ if $\|\varphi(u) - \varphi(v)\|^2 \leq \delta/2$, then $\|\varphi''(u) - \varphi''(v)\|^2 \leq 8/7 \cdot \delta/2 + \delta/8 < 3\delta/4$.

Therefore, inequalities (4.7) and (4.8) hold.

Observe, that $\mathbb{E}|E_R| = \text{sr-cost}(\mathcal{P}, \varepsilon) \leq X$. Hence, by the Chernoff bound (for some absolute constant C_1),

$$\Pr(|E_R| \geq 2X) \leq e^{-C_1 X}.$$

Similarly, by the Chernoff bound, inequalities (4.7) and (4.8) simultaneously hold with probability at most $e^{-C_2\delta^2 X}$. Thus, a fixed $\varphi'' : V \to N$ satisfies (4.7) and (4.8) with probability (over random choice of E_R) at most $e^{-C_3\delta^2 X}$. Since $\delta \geq 1/\sqrt{D}$, the total number of different embeddings $\varphi'' : V \to N$ equals $|N|^n \leq \exp(C_4 n \log n \log^2 D)$. By the union bound the probability that at least one such φ' exists is at most $e^{-C_3\delta^2 X + C_4 n \log n \log^2 D} \leq e^{-n}$. Hence, the first bound follows.

Improved Bound : Geometric Expansion of $\max\{\text{sr-cost}(\mathcal{P}, \varepsilon),\ nD(\log^2 D)\}$**.**

To get the improved bound, we observe that while performing dimension reduction, we do not need all the distances to be well-preserved. It suffices that at least $1-O(\delta^2)$ fraction of both the edge distances, and of all pairwise distances are preserved.

Hence, we show that if φ satisfying equations (4.5) and (4.4) exists then there exists an embedding $\varphi' : V \to N_\delta$ to a relatively small set $N_\delta \subset \mathcal{H}$ satisfying slightly relaxed conditions:

$$|\{(u,v) \in V \times V : \|\varphi'(u) - \varphi'(v)\|^2 \leq \frac{3}{4}\ \delta\}| \leq \frac{5}{4}\ \delta^2 n^2, \tag{4.9}$$

and,

$$|\{(u,v) \in E_R : \|\varphi'(u) - \varphi'(v)\|^2 \leq \frac{3}{4}\ \delta\}| \geq \frac{3}{2}\delta^2 X. \tag{4.10}$$

Here $N_\delta \subset \mathcal{H}$ is a set of size $\exp(O(\log^2(1/\delta))$ depending only on δ. Then, we argue that such φ' exists with very small probability.

Claim 4.12. *If $|E_R| \leq 2X$ and there exists $\varphi : V \to \mathcal{H}$ satisfying (4.4) and (4.5), then there exists $\varphi' : V \to N_\delta$ satisfying (4.9) and (4.10).*

Proof. We set parameters $\zeta = 1/7$, $\eta^2 = \delta/8$ and $\nu = \delta^2/8$ and pick N_δ as in Lemma 4.11. Then we choose a deterministic $\psi(u) : \varphi(V) \to N$ such that the condition

$$\begin{aligned}\frac{7}{8}\|\varphi(u) - \varphi(v)\|^2 - \frac{\delta}{8} &\leq \|\psi(\varphi(u)) - \psi(\varphi(v))\|^2 \\ &\leq \frac{8}{7}\ \|\varphi(u) - \varphi(v)\|^2 + \frac{\delta}{8}\end{aligned}$$

holds for at least a $(1-\delta^2/4)$ fraction of all pairs $u, v \in V$ and at least a $(1-\delta^2/4)$ fraction of all edges $(u,v) \in E_R$ (the existence of such ψ follows from (4.6), by the probabilistic method). Define $\varphi'(u) = \psi(\varphi(u))$. We get:

- for all but at most $\delta^2 n^2/4$ pairs $u, v \in V$, if $\|\varphi(u) - \varphi(v)\|^2 > \delta$, then $\|\varphi'(u) - \varphi'(v)\|^2 \geq 7\delta/8 - \delta/8 = 3\delta/4$;
- for all but at most $\delta^2|E_R|/4 \leq \delta^2 X/4$ edges $(u, v) \in E_R$ if $\|\varphi(u) - \varphi(v)\|^2 \leq \delta/2$, then $\|\varphi'(u) - \varphi'(v)\|^2 \leq 8/7 \cdot \delta/2 + \delta/8 < 3\delta/4$.

Therefore, inequalities (4.9) and (4.10) hold. □

As before we get by a simple Chernoff bound, that(for some absolute constant C_1),

$$\Pr(|E_R| \geq 2X) \leq e^{-C_1 X}.$$

Similarly, by the Chernoff bound, inequalities (4.9) and (4.10) simultaneously hold with probability at most $e^{-C_2\delta^2 X}$. Thus, a fixed $\varphi' : V \to N$ satisfies (4.9) and (4.10) with probability (over random choice of E_R) at most $e^{-C_3\delta^2 X}$. Since $\delta \geq 1/D$, the total number of different embeddings $\varphi' : V \to N$ equals $|N|^n \leq \exp(C_4 n \log^2 D)$. By the union bound over the different embeddings and the $O(\log D)$ scales, the probability that at least one such φ' exists is at most $e^{-C_3\delta^2 X + C_4 n \log^2 D} \leq e^{-n}$ here we use that $\delta^2 X \geq Cn(\log^2 D)$ for sufficiently large C.

Part II follows from Part I by taking the union bound over all 2^n possible choices of the set U. □

4.4 Balanced Cut

We show that there exists a constant factor bi-criteria approximation algorithm for the Balanced Cut problem in the semi-random model with $\varepsilon \geq \Omega(\sqrt{\log n}(\log\log n)^2/n)$.

Theorem 4.13. *There exists a randomized polynomial-time algorithm and absolute constants C, C_{BC}, such that for every set of vertices V of size n (for simplicity assume n is even), every partition $\mathcal{P} = \{L, R\}$, $|L| = |R| = n/2$, and every $\varepsilon \in (0, 1)$ with high probability over the random choices of $SR(\mathcal{P}, \varepsilon)$ the following statement holds:*

for every $G = (V, E) \in SR(\mathcal{P}, \varepsilon)$ the algorithm returns a balanced partition of V into sets L' and R' with $|L'|, |R'| \geq n/C$ and expected cost of the cut at most:

$$\mathbb{E}\big[\text{cost}(\{L', R'\}, E) \mid SR(\mathcal{P}, \varepsilon)\big] \leq C_{BC} \max\{\text{sr-cost}(\mathcal{P}, \varepsilon), n\sqrt{\log n}(\log\log n)^2\}.$$

Particularly, if $\varepsilon \geq \sqrt{\log n}(\log\log n)^2/n$, then

$$\mathbb{E}\big[\text{cost}(\{L', R'\}, E) \mid SR(\mathcal{P}, \varepsilon)\big] \leq 4C_{BC}\,\text{sr-cost}(\mathcal{P}, \varepsilon) = C_{BC}\varepsilon n^2.$$

We use the standard SDP relaxation for the Balanced Cut problem from Arora et al. [13] (given in Figure 4.4).

$$\begin{aligned}
\min \quad & \frac{1}{4}\sum_{(u,v)\in E(G)} \|\bar{u} - \bar{v}\|^2 \\
\text{subject to} \quad & \\
& \frac{1}{4}\sum_{u,v\in V} \|\bar{u} - \bar{v}\|^2 \geq \frac{n^2}{2} && \text{(Spreading constraint)} \\
\text{for all } u, v, w \in V, \quad & \|\bar{u} - \bar{v}\|^2 + \|\bar{v} - \bar{w}\|^2 \geq \|\bar{u} - \bar{w}\|^2 && (\ell_2^2\text{–triangle inequalities}) \\
\text{for all } u \in V, \quad & \|\bar{u}\|^2 = 1
\end{aligned}$$

Figure 4.4: SDP for minimum Balanced Cut

The SDP relaxation has a unit vector $\bar{u}$ for every vertex $u \in V$ and all vectors satisfy ℓ_2^2 triangle inequalities. The spreading constraint (we count every pair as (u, v) and (v, u)) tries to ensure that the partition is balanced. The SDP relaxation defines a set of feasible solutions Φ.

In the algorithm below, we use the algorithm of Arora, Rao, and Vazirani [13] for finding a balanced cut (in the worst-case). We denote the approximation factor of the algorithm by $D_{ARV} = O(\sqrt{\log n})$. For simplicity of exposition we assume that D_{ARV} is a power of 4.

Balanced Cut Algorithm in Semi-random Model

Input: a graph $G = (V, E) \in SR(\mathcal{P}, \varepsilon)$

Output: a cut (L', R'), with $|L'|, |R'| \geq n/C$

- Run the Hidden Solution Sparsification Algorithm with a separation oracle for Φ and obtain a set $M \subset V$, a partition $\mathcal{Z}$ of $V \setminus M$ in disjoint Φ–feasible sets and two disjoint sets of edges E^+ and E^- (with parameter $D = D_{ARV}$).
- Run the ARV algorithm on the graph $G = (V, E^+)$, and obtain a balanced partition (L', R');
- **return** (L', R').

Analysis. We show that every set Z in $\mathcal{Z}$ is balanced. Every set $Z \in \mathcal{Z}$ is Φ–feasible, that is, for some $\varphi \in \Phi$, $\varphi(Z)$ has ℓ_2^2 diameter at most $1/4$. Thus,

$$\begin{aligned}
&\frac{1}{2}\sum_{u,v\in V} \|\varphi(u)-\varphi(v)\|^2 \\
&\quad\leq \frac{1}{2}\sum_{u,v\in V} \max_{u,v\in V}(\|\varphi(u)-\varphi(v)\|^2) - \\
&\quad\quad - \frac{1}{2}\sum_{u,v\in Z} \left(\max_{u,v\in V}(\|\varphi(u)-\varphi(v)\|^2) - \max_{u,v\in Z}(\|\varphi(u)-\varphi(v)\|^2)\right) \\
&\quad\leq n^2 - \frac{7}{8}\,|Z|^2.
\end{aligned}$$

By the SDP spreading constraint, the left hand side is greater than or equal to $n^2/2$, thus $|Z| \leq \sqrt{4/7}\; n \leq 4/5\; n$.

By the Structural Theorem 4.8, with high probability for every graph $G = (V, E) \in SR(\mathcal{P}, \varepsilon)$, the graph $(V, \mathrm{cut}(\mathcal{P}, E))$ is geometrically expanding with cut cost

$$X = C \max\{\text{sr-cost}(\mathcal{P}, \varepsilon), n\sqrt{\log n}(\log\log n)^2\}$$

up to scale $1/\sqrt{D_{ARV}}$. Thus, by Theorem 4.6,

$$\text{cost}_{|M}(\{L, R\}, E^+) \leq C\ X/D_{ARV}.$$

Hence, there are at most $C\ X/D_{ARV}$ edges in E^+ going from $L \cap M$ to $R \cap M$. Observe, that $|L \cap M| \leq |L| = n/2$ and $|R \cap M| \leq |R| = n/2$. Therefore, there are at most $C\ X/D_{ARV}$ edges in E^+ cut by the the partition

$$V = (M \cap L) \cup (M \cap R) \cup \bigcup_{Z \in \mathcal{Z}} Z$$

(the only edges cut are the edges between $M \cap L$ and $M \cap R$) and each of the sets in the partition has size at most $4/5\ n$. These sets can be grouped into two balanced sets L^* and R^* with $|L^*|, |R^*| \geq 1/5\ n$. The ARV algorithm finds a possibly different balanced cut (L', R') (with slightly weaker bounds on $|L'|$, $|R'|$). The number of edges cut in E^+ is bounded (in expectation) by $D_{ARV} \times C\ X/D_{ARV} = C\ X$. The number of edges cut in E^- is bounded by $|E^-| \leq C\ X$.

4.5 Min Multicut

The algorithm for the Multicut problem is similar to the algorithm for Balanced Cut. We use the standard SDP relaxation for Multicut: The SDP has a unit vector $\bar{u}$ for every vertex u; vectors $\bar{s}_i$, $\bar{t}_i$ corresponding to source–sink pairs s_i, t_i are orthogonal ($\langle \bar{s}_i, \bar{v}_i \rangle = 0$); all vectors satisfy the ℓ_2^2 triangle inequality constraints.

The key observation is that every Φ–feasible set $Z \in \mathcal{Z}$ does not contain a source–sink pair s_i, t_i.

Lemma 4.14 (Multicut: Φ-feasibility)**.** *Given a multicut instance $G(V, E)$ with source-sink pairs $\{(s_i, t_i)\}_{1 \leq i \leq r}$, any Φ-feasible Z does not contain any of the source-sink pairs $\forall i, (s_i, t_i)$.*

$$\min \quad \frac{1}{2} \sum_{(u,v)\in E(G)} \|\bar{u} - \bar{v}\|^2$$

subject to

$$\begin{array}{rll} \text{for all } 1 \le i \le k, & \langle \bar{s}_i, \bar{t}_i \rangle = 0 & \\ \text{for all } u, v, w \in V, & \|\bar{u} - \bar{v}\|^2 + \|\bar{v} - \bar{w}\|^2 \ge \|\bar{u} - \bar{w}\|^2 & (\ell_2^2\text{–triangle inequalities}) \\ \text{for all } u \in V, & \|\bar{u}\|^2 = 1 & \end{array}$$

Figure 4.5: SDP for minimum Multicut

Proof. For any feasible SDP solution φ, we know that for all source-sink pairs $i \in [r]$,

$$\langle \varphi(s_i), \varphi(t_i) \rangle = 0$$
$$\text{Hence,} \frac{1}{2}\|\varphi(s_i) - \varphi(t_i)\|^2 = \|\varphi(s_i)\|^2 + \|\varphi(t_i)\|^2 = 1$$

Since Z is Φ-feasible, there exists a feasible sdp solution φ for G such that

$$\forall u, v \in Z, \|\varphi(u) - \varphi(v)\|^2 \le 1/4$$

Thus, Z can not both s_i and t_i for any of the $i \in [r]$ source-sink pairs. □

Theorem 4.15. *There exists a randomized polynomial-time algorithm and an absolute constant C, such that for every set of vertices V of size n, every partition $\mathcal{P}$, and every $\varepsilon \in (0,1)$ with high probability over random choice of $SR(\mathcal{P}, \varepsilon)$ the following statement holds: for every $G = (V, E) \in SR(\mathcal{P}, \varepsilon)$ and every set of demands (s_i, t_i) (satisfying $\mathcal{P}(s_i) \ne \mathcal{P}(t_i)$), the algorithm returns a partition $\mathcal{P}'$ of V separating the demands ($\mathcal{P}'(s_i) \ne \mathcal{P}'(t_i)$) with expected cost of the cut at most:*

$$\mathbb{E}\big[\text{cost}(\mathcal{P}', E) \mid SR(\mathcal{P}, \varepsilon)\big] \le C \max\{\text{sr-cost}(\mathcal{P},\ \varepsilon), n \log n (\log\log n)^2\}.$$

Proof. The proof follows exactly as in the proof of Theorem 4.13. We use the $O(\log n)$ worst-case approximation algorithm of Garg, Vazirani, and Yannakakis [44], on the remaining instance. Hence we set $D = \log n$.

By the Structural Theorem 4.8, with high probability for every graph $G = (V, E) \in SR(\mathcal{P}, \varepsilon)$, the graph $(V, \text{cut}(\mathcal{P}, E))$ is geometrically expanding with cut cost

$$X = C \max\{\text{sr-cost}(\mathcal{P}, \varepsilon), n \log n (\log \log n)^2\}$$

up to scale $1/\sqrt{\log n}$. Thus, by Theorem 4.6,

$$\text{cost}_{|M}(\{L, R\}, E^+) \leq C\ X/\log n.$$

Hence, there is a multicut solution in (V, E^+) of cost at most $C\ X/\log n$. Hence, the algorithm of [44] on $G(V, E^+)$ results in a cut of cost $O(\log n \cdot X/\log n) = O(X)$.

We need to ensure that after cutting the edges in the solution, all the source-sink pairs are disconnected. By Lemma 4.14, we know that each of the Φ-feasible pieces $Z \in \mathcal{Z}$ has no source-sink pair. The number of edges cut in E^- is at most $C\ X$. Hence, we get a constant factor approximation algorithm for the Multicut problem in the semi-random model with $\varepsilon \geq \log n (\log \log n)^2/n$. □

4.6 Further work

A similar statement holds for the Min Uncut problem if $\varepsilon \geq \sqrt{\log n}(\log \log n)^2/n$. A semi-random instance of Min Uncut is generated as follows: the adversary first chooses an arbitrary subset S of vertices, then the nature connects each pair of vertices $(u, v) \in S \times S \cup (V \setminus S) \times (V \setminus S)$ with an edge with probability ε, finally the adversary adds arbitrary edges between S and $V \setminus S$, and removes some random edges. The problem can be restated as a cut minimization problem that falls in

our framework (see e.g. [1]). Our algorithm for Min Uncut first runs the Hidden Solution Sparsification algorithm and then uses the algorithm of Agarwal, Charikar, Makarychev, and Makarychev [1] for Min Uncut. We defer the details to [69].

Chapter 5

Semi-Random Small Set Expansion and Recovering Partitions

In the previous chapter, we discussed how we can obtain $O(1)$ approximations using the Hidden Solution Sparsification (Theorem 4.6) for semi-random instances of Balanced Separator and Multicut. Unfortunately, the guarantees of Theorem 4.6 are not strong enough to yield similar approximations for Small Set Expansion (and Sparsest Cut). The difficulty for Small Set Expansion (SSE) is that we get a weaker guarantee on the size of the cut edges (the set E^- in Chapter 4), so the cost of the partition ($\{M_i, Z_j\}$ in the notation of Chapter 4) might be very high. In the notation of Chapter 4, we use an extra post-processing step to find a subset of edges in E^- that we really need to cut.

The main reason for the difficulty is that we cannot use the corresponding SDP relaxation of Bansal et al. [18] for SSE (given in figure 5.1).

Loosely speaking, it may "find" a good fractional cut that assigns zero vectors to the real solution, since the required set size $|S|$ could be much smaller than n. In that case, the SDP solution may provide little information about the hidden solution S.

$$\min \quad \frac{1}{2} \sum_{(u,v)\in E(G)} \|\bar{u} - \bar{v}\|^2$$

subject to

$$\begin{aligned}
\text{for all } u \in V, \quad & \sum_{v\in V} \max\{\langle \bar{u}, \bar{v}\rangle, \|\bar{u}\|^2\} \leq \rho n \|\bar{u}\|^2 && \text{(Spreading constraints)} \\
\text{for all } u, v, w \in V, \quad & \|\bar{u} - \bar{v}\|^2 + \|\bar{v} - \bar{w}\|^2 \geq \|\bar{u} - \bar{w}\|^2 && (\ell_2^2\text{–triangle inequalities}) \\
& \sum_{u\in V} \|\bar{u}\|^2 \leq \rho n && \text{(Size constraint)} \\
\text{for all } u, v \in V \quad & \langle \bar{u}, \bar{v}\rangle \geq 0 \\
\text{for all } u \in V, \quad & \|\bar{u}\|^2 \leq 1
\end{aligned}$$

Figure 5.1: SDP relaxation for Small Set Expansion due to [18]

This happens for instance, when the adversarial instance inside $V \setminus S$ is comprised of an integrality gap instance for the SDP relaxation given in figure 5.1.

To overcome this issue, we instead use a "Crude SDP" (C-SDP) for the problem (Figure 5.2 on page 80). C-SDPs were recently introduced in the work of Kolla, Makarychev and Makarychev [61]. The C-SDP for the Small Set Expansion is *not* a relaxation for the problem; its objective value may be much larger than the value of the optimal integral solution (in particular, the value of a C-SDP can be large even if the cost of the optimal solution is 0).

The algorithm for Small Set Expansion (SSE) requires several new ingredients. In fact, our algorithm for the Small Set Expansion (SSE) problem is the most involved, and uses the full power of the Hidden Solution Sparsification theorem. To solve a semi-random instance of the Small Set Expansion problem, we first apply the Hidden Solution Sparsification step. However, the number of edges in E^- is bounded in expectation by the cost of the C-SDP solution and may be much larger than the cost of the optimal solution. So our algorithm cannot afford to cut all these edges. Nevertheless, we prove that the number of edges in E^- incident to the set S (S is the

optimal solution, which is not known to the algorithm) is bounded by $O(OPT)$ (the total number of edges in E^- can be much larger than OPT). This gives our more powerful guarantees given in Theorem 5.3. Then we show how to find a good solution by combining the SDP based SSE algorithm [18] with a new linear programming relaxation based algorithm.

We extend this to an algorithm for semi-random Sparsest Cut as well, by first guessing the number of vertices in the Sparsest Cut and then using our algorithm for semi-random Small Set Expansion. While our algorithms for Small Set Expansion and Sparsest Cut are of interest in their own right, they are also useful primitives to recover the partitions, when we are given extra conditions about the expansion inside the partitions.

Recovering Partitions through Solution Purification. We show how we use our algorithm for semi-random Sparsest Cut to iteratively get closer and closer to the hidden partitions, under some additional assumptions about the stability of the optimal solution. More precisely, if we additionally assume that graphs $G[S]$ and $G[V \setminus S]$ are combinatorial expanders in the Balanced Cut or Small Set Expansion problem, then we can almost recover sets S and $G \setminus S$ (see Theorem 5.9) upto any accuracy. We do that by first finding a good approximate solution using our algorithm for Balanced Cut (or Small Set Expansion) and then improving the solution by repeatedly solving (semi-random) instances of the Sparsest Cut problem to obtain successively finer approximations to the planted partition.

Before we describe the more general Hidden Solution Sparsification theorem, we first introduce the concept of $\mathcal{O}$-local SDPs which will be important for the rest of the chapter.

5.1 Local SDP Relaxations

Let us first recollect how we measure the cost of the SDP solution φ.

Definition 5.1. *Let $G = (V, E)$ be a graph, $\mathcal{P}$ be a partition of V, and $\mathcal{O}$ be a subset of V. Define the cost of an SDP solution $\varphi : V \to \mathcal{H}$ to be*

$$\text{sdp-cost}(\varphi, E) \equiv \frac{1}{2} \sum_{(u,v)\in E} \|\varphi(u) - \varphi(v)\|^2,$$

and the cost of the SDP solution restricted to the set $\mathcal{O}$ to be

$$\text{sdp-cost}_{|\mathcal{O}}(\varphi, E) \equiv \frac{1}{2} \sum_{\substack{(u,v)\in E \\ u\in\mathcal{O} \text{ or } v\in\mathcal{O}}} \|\varphi(u) - \varphi(v)\|^2.$$

For any SDP relaxation, the cost of the optimum (minimum) SDP solution lower bounds the value of the best integral solution. However, this lower bound may not hold when restricted to a subset $\mathcal{O} \subseteq V$. This motivates the following definition:

Definition 5.2. *Let V be a set of vertices, $\mathcal{P}$ be a partition of V and $\mathcal{O} \subseteq V$. We say that a non-empty set of SDP solutions Φ is a $\mathcal{O}$–local relaxation of $\mathcal{P}$ if there exists a constant $C \geq 1$ such that for every graph $G = (V, E)$ on V and for*

$$\varphi = \arg\min_{\varphi\in\Phi} \text{sdp-cost}(\varphi, E) \equiv \arg\min_{\varphi\in\Phi} \frac{1}{2} \sum_{(u,v)\in E} \|\varphi(u) - \varphi(v)\|^2,$$

the following inequality holds

$$\text{sdp-cost}_{|\mathcal{O}}(\varphi, E) \leq C \, \text{cost}_{|\mathcal{O}}(\mathcal{P}, E).$$

Note that an SDP relaxation of a problem is always a V-local SDP relaxation of the optimal integral solution.

5.2 Hidden Solution Sparsification and Applications

We now present the more general Hidden Solution Sparsification Algorithm and prove the following.

Theorem 5.3. (Hidden Solution Sparsification)
There exists a polynomial-time randomized algorithm that given a graph $G = (V, E)$, a separation oracle for an $\mathcal{O}$–local SDP relaxation Φ of a partition $\mathcal{P}$ (note: the set $\mathcal{O} \subset V$ and partition $\mathcal{P}$ are "hidden" and are not known to the algorithm), and a parameter $D = 2^T$ ($T \in \mathbb{N}$, $T > 1$), partitions the set of vertices V into a set M and a collection of disjoint sets $\mathcal{Z}$

$$V = M \cup \bigcup_{Z \in \mathcal{Z}} Z,$$

and also partitions the set of edges into two disjoint sets E^+ and E^-

$$E = E^+ \cup E^-$$

such that

- *all edges cut by the partition $V = M \cup \bigcup_{Z \in \mathcal{Z}} Z$ lie in E^- (i.e., $\mathrm{cut}(\{M\} \cup \mathcal{Z}, E) \subset E^-$), or in other words,*

$$E^+ \subset M \times M \cup \bigcup_{Z \in \mathcal{Z}} Z \times Z;$$

- *if the graph $(V, \mathrm{cut}(\mathcal{P}, E))$ satisfies the geometric expansion property with cut value X up to scale $1/\sqrt{D}$, then (the expectation is taken over random bits of the algorithm)*

$$\mathbb{E}[\mathrm{cost}_{|\mathcal{O} \cap M}(\mathcal{P}, E^+)] \leq C\ X/D; \tag{5.1}$$

and

$$|\{(u,v) \in E^- : u \in \mathcal{O} \text{ or } v \in \mathcal{O}\}| \leq C\, X; \tag{5.2}$$

- *each $Z \in \mathcal{Z}$ is Φ–feasible.*

The algorithm for Hidden Solution Sparsification is exactly the same as in Section 4.2. However, we present it here again for completeness. As before, the algorithm runs in $O(\log \log n)$ phases, and in each phase, we first solve the SDP on the current instance and then cut out the heavy vertices w.r.t. to this vector solution before sparsifying the hidden solution in the remaining graph by cutting the long edges.

Hidden Solution Sparsification Algorithm

Input: a graph $G = (V, E)$ and a separation oracle for a set of SDP solutions $\Phi \subset \{V \to \mathcal{H}\}$.

Output: partitions $V = M \cup \bigcup_{Z \in \mathcal{Z}} Z$ and $E = E^+ \cup E^-$.

- Let $M_0 = V$, $\mathcal{Z}_0 = \varnothing$, $E_0^+ = E$, $E_0^- = \varnothing$, $T = \frac{1}{2}\log_2 D$, and $\delta_t = 2^{-t}$ for all $t = 1, \ldots, T$.

- for $t = 1, \ldots, T$ do

  A. *Solve the SDP for the remaining graph:* Find

  $$\varphi_t = \arg\min_{\varphi \in \Phi} \text{sdp-cost}(\varphi, E_{t-1}^+ \cap (M_{t-1} \times M_{t-1})).$$

  B. *Remove δ_t–heavy vertices:* run Heavy Vertices Removal Algorithm (described in Section 4.2.1) with parameters V, M_{t-1}, φ_t, and obtain a collection of Φ–feasible sets ΔZ_t. Add edges in E_{t-1}^+ cut by ΔZ_t to the set ΔE_t^-. Let

  $$\mathcal{Z}_t = \mathcal{Z}_{t-1} \cup \Delta\mathcal{Z}_t; \;\; M_t = M_{t-1} \setminus \bigcup_{Z \in \Delta\mathcal{Z}_t} Z.$$

C. Remove δ_t–long edges from E^+: Find

$$L_t = \{(u,v) \in E^+ : u, v \in M_t, \|\varphi_t(u) - \varphi_t(v)\|^2 \geq \delta_t\}.$$

Let

$$E_t^+ = E_{t-1}^+ \setminus (\Delta E_t^- \cup L_t); \;\; E_t^- = E_{t-1}^- \cup (\Delta E_t^- \cup L_t).$$

- **return** $M = M_T$, $\mathcal{Z} = \mathcal{Z}_T$, $E^+ = E_T^+$, $E^- = E_T^-$.

Proof of Theorem 5.3. The proof follows along the same lines as in Theorem 4.6 — however, the cost of solution is analysed in terms of an $\mathcal{O}$-local relaxation, to only consider the edges incident on $\mathcal{O}$.

As in Theorem 4.6, we first note the algorithm runs in polynomial time. At every iteration, the algorithm removes all edges crossing the partition $\Delta\mathcal{Z}_t$ from E_t^+ and adds them to E_t^-, hence the first item of Theorem 4.6 holds. The third item holds, because every set $Z \in \mathcal{Z}$ belongs to some $\Delta\mathcal{Z}_t$ and, thus by Lemma 4.7, $\operatorname{diam}(\varphi_t(Z)) \leq 1/4$.

We now show that the second item of Theorem 5.3 holds. We first prove that

$$\operatorname{cost}_{|M_t}(\mathcal{P}, E_t^+) \leq 2\, X \cdot \delta_t^2$$

for every $t \in \{0, \ldots, T\}$. The Heavy Vertices Removal Procedure returns set M_t that does not contain any δ_t–heavy vertices w.r.t. φ_t i.e., $H_{\delta_t,\varphi_t}(M_t) = \varnothing$ (see Lemma 4.7). Using the geometric expansion property of the graph $(V, \operatorname{cut}(E, \mathcal{P}))$, we get $\left|\{(u,v) \in \operatorname{cut}(\mathcal{P}, E) \cap (M_t \times M_t) : \|\varphi_t(u) - \varphi_t(v)\|^2 \leq \delta_t/2\}\right| \leq 2\delta_t^2 X$. The algorithm removes all $\delta_t/2$–long edges at step C, thus the set $E_t^+ \cap (M_t \times M_t)$ contains only edges (u,v)

for which $\|\varphi_t(u) - \varphi_t(v)\|^2 \leq \delta_t/2$. Combining this observation with the previous inequality, and using that edges in E_t^+ do not cross the boundary of M_t, we get

$$\mathrm{cost}_{|M_t}(\mathcal{P}, E_t^+) = \big|\mathrm{cut}(E, \mathcal{P}) \cap E_t^+ \cap (M_t \times M_t)\big| \leq 2\delta_t^2 X. \tag{5.3}$$

For $t = T$, we get $\mathrm{cost}_{|M}(\mathcal{P}, E^+) \leq 2\, X/D$.

Finally, we estimate the size of the set $\{(u, v) \in E^- : u \in \mathcal{O} \text{ or } v \in \mathcal{O}\}$. To do so, we use that Φ is a $\mathcal{O}$–local relaxation of the partition $\mathcal{P}$. For graph $G = (V, E_{t-1}^+ \cap (M_{t-1} \times M_{t-1}))$, we obtain inequality

$$\begin{aligned}
\text{sdp-cost}_{|\mathcal{O}}(\varphi_t, E_{t-1}^+ \cap (M_{t-1} \times M_{t-1})) &\leq C_1\, \mathrm{cost}_{|\mathcal{O}}(\mathcal{P}, E_{t-1}^+ \cap (M_{t-1} \times M_{t-1})) \\
&= C_1\, \mathrm{cost}_{|\mathcal{O} \cap M_{t-1}}(\mathcal{P}, E_{t-1}^+) \leq 2CX \cdot \delta_{t-1}^2 \\
&= 8C_1\delta_t^2 X.
\end{aligned}$$

The third line of the inequality follows from (5.3).

Now, we bound the number of edges removed from $E_{t-1}^+ \cap \mathcal{O}$ and added to $E_t^- \cap \mathcal{O}$ in terms of "sdp-cost". At step t, we add two sets of edges to E^-: ΔE_t^- and L_t. Since all edges (u, v) in L_t are $\delta_t/2$–long (i.e., $\|\varphi(u) - \varphi(v)\|^2 \geq \delta_t/2$),

$$\begin{aligned}
\text{sdp-cost}_{|\mathcal{O}}(\varphi_t, E_{t-1}^+ \cap (M_{t-1} \times M_{t-1})) &\equiv \sum_{\substack{(u,v) \in E_{t-1}^+ \cap (M_{t-1} \times M_{t-1}) \\ (u,v) \in \mathcal{O} \times V}} \frac{\|\varphi(u) - \varphi(v)\|^2}{2} \\
&\geq \frac{|L_t \cap (\mathcal{O} \times V)| \cdot \delta_t/2}{2}.
\end{aligned}$$

Hence, $|L_t \cap (\mathcal{O} \times V)| \leq 32C_1\delta_t X$. The probability that the Heavy Vertices Removal Procedure separates two vertices u and v connected with an edges in E_{t-1}^+ is at most $C_2\big(\delta_t^{-1} + \delta_t^{-2}\mathbb{E}|M_{t-1} \setminus M_t|/n\big) \cdot \|\varphi(u) - \varphi(v)\|^2$ (see Lemma 4.7). Thus, the expected

total number of edges cut in the set $\mathcal{O} \times V$ is

$$\begin{aligned}\mathbb{E}\left[|\Delta E_t^- \cap (\mathcal{O} \times V)|\right] &\leq C_2\Big(\delta_t^{-1} + \delta_t^{-2}\frac{\mathbb{E}|M_{t-1} \setminus M_t|}{n}\Big) \cdot \text{sdp-cost}_{|\mathcal{O}}(\varphi_t, E_{t-1}^+ \cap (M_{t-1} \times M_{t-1})) \\ &\leq 8C_1C_2\Big(\delta_t + \frac{\mathbb{E}|M_{t-1} \setminus M_t|}{n}\Big)X.\end{aligned}$$

The total number of edges in $E^- \cap (\mathcal{O} \times V)$ is bounded by

$$\begin{aligned}&\sum_{t=1}^{T}(32C_1\delta_t + 8C_1C_2\delta_t + 8C_1C_2 \cdot \frac{\mathbb{E}|M_{t-1} \setminus M_t|}{n})X \\ &\leq (32C_1 + 8C_1C_2 + 8C_1C_2)X.\end{aligned}$$

□

We now show how to construct constant factor approximation algorithms for Small Set Expansion and Sparsest Cut in semi-random instances.

5.3 Small Set Expansion

We now show how the more general Hidden Sparsification procedure gives our $O(1)$ for Small Set Expansion.

Theorem 5.4. *There exists a randomized polynomial-time algorithm and an absolute constant C, such that for every set of vertices V of size n, every partition $\mathcal{P} = \{S, V \setminus S\}$, $|S| = \rho n$ (for $\rho \in (0, 1/2)$) and every $\varepsilon \in (0, 1)$ with high probability over the random choices of $SR(\mathcal{P}, \varepsilon)$ the following statement holds: for every $G = (V, E) \in SR(\mathcal{P}, \varepsilon)$, the algorithm given G and ρ, returns a partition $\mathcal{P}' = (S', V \setminus S')$ of V such that $|S'| = \Theta(\rho n)$, $|S'| \leq |V|/2$ with expected cost of the cut at most:*

$$\begin{aligned}&\mathbb{E}\big[\text{cost}(\mathcal{P}', E) \mid SR(\mathcal{P}, \varepsilon)\big] \\ &\quad\leq C \max\{\text{sr-cost}(\mathcal{P},\ \varepsilon), n\sqrt{\log n \log(1/\rho)}(\log\log n)^2\}.\end{aligned}$$

Particularly, if $\varepsilon\rho \geq \sqrt{\log n \log(1/\rho)}(\log\log n)^2/n$, *then*

$$\begin{aligned}\mathbb{E}\big[\text{cost}(\{L,R\},E) \mid SR(\mathcal{P},\varepsilon)\big] &\leq C\,\text{sr-cost}(\mathcal{P},\varepsilon)\\ &= C\varepsilon\rho(1-\rho)n^2.\end{aligned}$$

Moreover, instead of requiring that $G=(V,E)\in SR(\mathcal{P},\varepsilon)$, *it suffices that the graph* $(V,\text{cut}(\mathcal{P},E))$ *is geometrically expanding with cut cost*

$$X = C'\max\{\text{sr-cost}(\mathcal{P},\ \varepsilon), n\sqrt{\log n\log(1/\rho)}(\log\log n)^2\}$$

(for some absolute constant C'*) up to scale*

$$s(n,\rho) = \Omega(\sqrt{\log n \log\frac{1}{\rho}}).$$

By Theorem 4.8, for every graph $G\in SR(\mathcal{P},\varepsilon)$, the graph $(V,\text{cut}(\mathcal{P},E))$ is geometrically expanding with cut cost

$$X = C'\max\{\text{sr-cost}(\mathcal{P},\ \varepsilon), n\sqrt{\log n\log(1/\rho)}(\log\log n)^2\}$$

up to scale $s(n,\rho)=\Omega(\sqrt{\log n\log\frac{1}{\rho}})$ with probability $1-o(1)$ over random choice of $SR(\mathcal{P},\varepsilon)$. We assume that the graph $(V,\text{cut}(\mathcal{P},E))$ is geometrically expanding. Otherwise, the algorithm fails (this happens with probability $o(1)$).

We use an analog of the Crude SDP (C–SDP) introduced in the paper of Kolla, Makarychev and Makarychev [61]. For each vertex $u\in V$ the C–SDP has a unit vector $\bar{u}\in\mathcal{H}$. All vectors satisfy triangle inequality constraints and spreading constraints (similar to constraints introduced in Bansal et al. [18]) : for every $u\in V$,

$$\sum_{v\in V}\langle u,v\rangle \leq \rho n.$$

We give the C-SDP in its entirety in figure 5.2.

$$\min \quad \frac{1}{2} \sum_{(u,v)\in E(G)} \|\bar{u} - \bar{v}\|^2$$

subject to

$$\begin{aligned}
&\text{for all } u \in V, && \sum_{v\in V} \langle \bar{u}, \bar{v} \rangle \leq \rho n && \text{(Spreading constraints)} \\
&\text{for all } u, v, w \in V, && \|\bar{u} - \bar{v}\|^2 + \|\bar{v} - \bar{w}\|^2 \geq \|\bar{u} - \bar{w}\|^2 && (\ell_2^2\text{–triangle inequalities}) \\
&\text{for all } u, v \in V && \langle \bar{u}, \bar{v} \rangle \geq 0 \\
&\text{for all } u \in V, && \|\bar{u}\|^2 = 1
\end{aligned}$$

Figure 5.2: Crude SDP for Small Set Expansion (SSE)

Note that this SDP is not a relaxation for SSE. However, it turns out that this is a S–local SDP relaxation of partition $(S, V \setminus S)$.

Lemma 5.5. *The set Φ of feasible solutions of the Crude SDP (C-SDP) given in Figure 5.2 on page 80 for the Small Set Expansion problem is a S-local relaxation of every partition $\mathcal{P} = \{S, V \setminus S\}$ (where $|S| = \rho n$).*

Proof. Let $\varphi = \arg\min_{\varphi\in\Phi} \text{sdp-cost}(\varphi, E)$. Denote $\bar{u} = \varphi(u)$. Define a new SDP solution

$$\bar{u}' = \begin{cases} \bar{e}_\perp & \text{if } u \in S \\ \bar{u} & \text{otherwise.} \end{cases}$$

where $\bar{e}_\perp$ is a unit vector orthogonal to all the vectors $\{\bar{v}\}_{v\in V(G)}$. This solution also satisfies the ℓ_2^2–triangle inequalities, the spreading constraints (because $|S| \leq \rho n$ and for all $u \in S$, $v \in V \setminus S$, $\langle \bar{u}', \bar{v}' \rangle = 0 \leq \langle \bar{u}, \bar{v} \rangle$), and for all $u, v \in V$, $\langle \bar{u}', \bar{v}' \rangle \geq 0$. Thus, it lies in Φ.

Compute the cost of the new solution and compare it with the cost of the optimal solution:

$$\begin{aligned}\text{sdp-cost}(u \to \bar{u}', E) &= \frac{1}{2} \sum_{(u,v)\in E(G)} \|\bar{u}' - \bar{v}'\|^2 \\ &= \frac{1}{2} \sum_{\substack{(u,v)\in E(G) \\ u\in S \text{ or } v\in S}} \|\bar{u}' - \bar{v}'\|^2 + \frac{1}{2} \sum_{\substack{(u,v)\in E(G) \\ u,v\in V(G)\setminus S}} \|\bar{u}' - \bar{v}'\|^2 \\ &= \text{cost}_{|S}(\mathcal{P}, E) + \frac{1}{2} \sum_{\substack{(u,v)\in E(G) \\ u,v\in V(G)\setminus S}} \|\bar{u} - \bar{v}\|^2.\end{aligned}$$

The cost sdp-cost$(u \to \bar{u}, E)$ of the optimal solution $\bar{u}$ equals

$$\text{sdp-cost}_{|S}(u \to \bar{u}, E) + \frac{1}{2} \sum_{\substack{(u,v)\in E(G) \\ u,v\in V(G)\setminus S}} \|\bar{u} - \bar{v}\|^2.$$

Thus, sdp-cost$_{|S}(u \to \bar{u}, E) \le \text{cost}_{|S}(\mathcal{P}, E)$. □

We now use the Hidden Solution Sparsification algorithm to find the set M and a partition of $V \setminus M$ into Φ–feasible sets $Z \in \mathcal{Z}$. Here Φ is the set of feasible C–SDP solutions. We set the weight of every vertex $u \in M$ to be the number of edges in E^- incident on u: $w_u = |\{v : (u, v) \in E^-\}|$. w_u corresponds to the cost we would pay for cutting the w_u edges incident on u from E^-, if u were included in the solution (small set). Observe, that the weight of the "hidden" set S is at most $C_1 X$ (for some absolute constant C_1, see (5.2)). Then, we consider two cases: $|M \cap S| \ge |S|/2$ and $|(V \setminus M) \cap S| \ge |S|/2$, depending on whether most of the hidden set S vertices belong to M or the pieces $Z \in \mathcal{Z}$ of the partition (the algorithm does not know which of the inequalities holds and tries both options). In the first case, since there is a good fraction of S is inside M, we just apply a worst-case algorithm for Small Set Expansion on M, as in Section 4.4. In the second case, the hidden solution S is

spread across the many *small* pieces $\{Z_i\}$. Here, we use a linear program to pick out the vertices in the solution from the many pieces.

Case I: This case is handled similar to the proof in Section 4.4. Since most of S (the hidden solution) belongs to M, we know that there is a good solution $M \cap S$ in $G(V, E^+)$ i.e. $S \cap M$ has size $\in [\rho n/2, \rho n]$ with weight $w(M \cap S) \leq C_1 X$, and there are at most CX/D_{SSE} from E^+ going out of $S \cap M$. Now, we use the following theorem of Bansal et al. [18] which finds small non-expanding sets.

Theorem 5.6. (Special case of Theorem 2.1 [18], arXiv version) *There exists a polynomial-time algorithm ("SSE algorithm") that given as input a graph $G = (V, E)$, a set of positive weights w_u ($u \in V$), $\rho \in (0, 1/2]$ and $W \in \mathbb{R}^+$, finds a non-empty set $S \subset V$ satisfying $|S| \in [\Omega(\rho n), 3\rho n/2]$, and $w(S) \equiv \sum_{u \in S} w_u \leq CW$, such that*

$$E(S, V \setminus S) \leq D_{SSE} \cdot \min\big\{E(S, V \setminus S) : |S| = \rho n,\ w(S) \leq W\big\},$$

where $D_{SSE} = O(\sqrt{\log n \log(1/\rho)})$.

We use this SSE algorithm on $G(V, E^+)$ to find a set S' with $|S'| \in [\Omega(\rho n), 3\rho n/2]$, $w(S') \equiv \sum_{u \in S'} w_u \leq C \cdot C_1 X$, and

$$\text{cost}(\{S', V \setminus S'\}, E^+) \leq D_{SSE} \times CX/D_{SSE} \leq CX.$$

The total cost of the cut $E(S', V \setminus S')$ is bounded by the number of edges cut in E^+ and E^- which is at most CX and CC_1X respectively, which is $O(X)$ as needed.

If $\rho \in (1/3, 1/2)$, the set S' may contain more than $n/2$ vertices, but no more than $3\rho n/2 \leq \frac{3}{4} n$. Then the algorithm returns $S'' = V \setminus S'$ satisfying $|S''| \in [n/4, n/2]$.

Case II: In this case, the hidden solution S could mostly be spread arbitrarily among the pieces $Z \in \mathcal{Z}$ in $V \setminus M$. Here, our algorithm uses an LP to extract

the solution from the set $V \setminus M$. The key observation is that this set is already partitioned into pieces of small size. Indeed, every $Z \in \mathcal{Z}$ is Φ–feasible, and thus for some $\varphi \in \Phi$, $\mathrm{diam}(\varphi(Z)) \leq 1/4$ and, consequently, for every $u, v \in Z$, $\langle \varphi(u), \varphi(v) \rangle = (\|\varphi(u)\|^2 + \|\varphi(v)\|^2 - \|\varphi(u) - \varphi(v)\|^2)/2 \geq 7/8$. Using the C-SDP spreading constraint (for an arbitrary $u \in Z$),

$$\sum_v \langle \varphi(u), \varphi(v) \rangle \leq \rho n,$$

we get $|Z| \leq 8/7\ \rho n$.

The LP has a variable $x_v \in [0,1]$ for every vertex $v \in V \setminus M$; and the only constraint is that $\sum_{u \in V \setminus M} x_u \geq \rho n/2)$. The objective function is

$$\min \sum_{u \in V \setminus M} w_u x_u \ + \sum_{\substack{(u,v) \in E^+ \\ u,v \in V \setminus M}} |x_u - x_v|. \tag{5.4}$$

The canonical solution to this LP is as follows: $x_u = 1$, if $u \in S \cap (V \setminus M)$; $x_u = 0$, otherwise. The LP cost of this solution is at most CX, because the first term in the objective function is bounded by $C_1\ X$ (see (5.2)), the second term is bounded by the size of the cut$(\mathcal{P}, E)$, which is at most $C_2\ X$ (The expected size of the cut equals sr-cost$(\mathcal{P}, \varepsilon)$; by the Chernoff bound the size of the cut is less than $2\,$sr-cost$(\mathcal{P}, \varepsilon)$ with very high probability). Thus, the cost of the optimal solution $\{x^*_u\}$, which we denote by LP^*, is at most $C\ X = (C_1 + C_2)X$. For an integral solution $S' \subset V$, we define the cost

$$\begin{aligned} f(S') &= \sum_{u \in S'} w_u \ + \ |E^+(S', V \setminus S')\}| \\ &\equiv \sum_{u \in S'} w_u \ + \ |\{(u,v) \in E^+ : u \in S', v \notin S'\}|. \end{aligned} \tag{5.5}$$

For every $r \in [0,1]$ define $S_r = \{u : x^*_u \geq r\}$. The algorithm finds r^* that minimizes the ratio $f(S_r)/|S_r|$ subject to $|S_r| \geq \rho n/4$ (note: $|S_1| = |V \setminus M| \geq \rho n/2$).

Then it sorts all sets $Z \in \mathcal{Z}$ in order of increasing ratio $f(S_{r^*} \cap Z)/|S_{r^*} \cap Z|$ (ignoring empty sets) and gets a list $Z_1, \dots Z_K$. It picks the first k pieces such that

$$|Z_1 \cap S_{r^*}| + |Z_2 \cap S_{r^*}| + \cdots + |Z_k \cap S_{r^*}| \in [\rho n/4, 2\rho n],$$

and returns

$$S' = \bigcup_{i=1}^{k} Z_i \cap S_{r^*}.$$

Note, that such k exists because each piece $Z_i \cap S_{r^*}$ has size at most $8/7\ \rho n$ (as $|Z_i| \leq 8/7\ \rho n$) and $\sum_{i=1}^{n} |Z_i \cap S_{r^*}| \equiv |S_{r^*}| \geq \rho n/4$.

Analysis of Case II. We first prove that

$$f(S_{r^*}) \leq 4LP^*/(\rho n) \cdot |S_{r^*}|.$$

Observe that

$$\int_0^1 f(S_r)dr = LP^* \qquad \int_0^1 |S_r|dr \geq \frac{\rho n}{2}.$$

The first equality easily follows from (5.4) and (5.5), the second equality follows from the LP constraint. Let $R = \{r : |S_r| \geq \rho n/4\}$. Then

$$\int_R |S_r|dr \geq \frac{\rho n}{2} - \int_{[0,1]\setminus R} |S_r|dr \geq \frac{\rho n}{4},$$

and, since $r^* = \min\{f(S_r)/|S_r| : r \in R\}$,

$$LP^* = \int_R f(S_r)dr \geq \int_R \frac{f(S_{r^*})}{|S_{r^*}|}\ |S_r|dr \geq \frac{f(S_{r^*})}{|S_{r^*}|} \cdot \frac{\rho n}{4}.$$

Thus, $f(S_{r^*}) \le 4LP^*/(\rho n) \cdot |S_{r^*}|$. Using that edges in E^+ do not cross the boundaries of sets Z_i, we get

$$f(S_{r^*}) = \sum_{i=1}^{K} f(Z_i \cap S_{r^*}) \le \frac{4LP^*}{\rho n} \sum_{i=1}^{K} |S_{r^*} \cap Z_i|.$$

Recall, that $\{f(Z_i \cap S_{r^*})/|S_{r^*} \cap Z_i|\}_i$ is an increasing sequence, thus

$$\begin{aligned} f(S') \equiv f\Big(\bigcup_{i=1}^{k} Z_i \cap S_{r^*}\Big) &= \sum_{i=1}^{k} f(Z_i \cap S_{r^*}) \\ &\le \frac{4LP^*}{\rho n} \sum_{i=1}^{k} |S_{r^*} \cap Z_i| = \frac{4LP^*}{\rho n} \cdot |S'| \le 16LP^*. \end{aligned}$$

5.4 Sparsest Cut

We now show how to find an approximate sparsest cut in a semi-random graph G using the algorithm for Small Set Expansion. Specifically, we give an algorithm that for every subset $U \subseteq V$ intersecting each of the pieces of the planted partition (S, T) (see below for details), returns a cut $(A, U \setminus A)$ of sparsity

$$\frac{E(A, U \setminus A)}{|A|} \le O(\varepsilon n).$$

In Section 5.5, we show that the Sparsest Cut algorithm can be used to recover pieces S and T assuming that the graphs $G[S]$ and $G[T]$ have large expansion. We remark that while we are usually concerned with the case when $U = V$ for the sparsest cut problem, the following stronger statement is also useful for Section 5.5.

Theorem 5.7. *There exists a randomized polynomial-time algorithm and an absolute constant C, such that for every set of vertices V of size n, every partition $\mathcal{P} = \{S, V \setminus S\}$ and every $\varepsilon, \eta \in (0, 1)$ satisfying $\varepsilon\eta \ge \sqrt{\log n}(\log\log n)^2/n$ with high probability over the random choices of $SR(\mathcal{P}, \varepsilon)$ the following statement holds: for*

every $G = (V, E) \in SR(\mathcal{P}, \varepsilon)$, every $U \subseteq V$ such that $|U \cap S| \geq \eta n$ and $|U \cap T| \geq \eta n$, the algorithm given G, returns a partition $(A, U \setminus A)$ of $G[U]$ with $|A| \leq |U|/2$ such that with probability exponentially close to 1,

$$\frac{|E(A, U \setminus A)|}{|A|} < C_{SC}\varepsilon n. \tag{5.6}$$

Sketch. We first give a proof assuming $\varepsilon\eta \geq \sqrt{\log n \log(1/\eta)}(\log\log n)^2/n$.

Our algorithm guesses the size of $|S \cap U|$, computes the size of $|T \cap U| = |U| - |S \cap U|$. Then, it runs the Small Set Expansion algorithm on $G[U]$ with $\rho = \min(|S \cap U|, |T \cap U|)/|U|$, obtains a set A ($|A| \leq |U \setminus A|$) of size $\Theta(\rho|U|)$ and returns the cut $(A, U \setminus A)$. We need to show that the size of the cut $(A, U \setminus A)$ is at most $O(\varepsilon\rho|U|n)$, so that the sparsity of the cut is then $O(\varepsilon n)$.

Let us explain why we can use the Small Set Expansion algorithm for the graph $G[U]$ and why the algorithm finds a cut of cost at most $O(\varepsilon\rho|U|n)$. By the structural theorem (Theorem 4.8 part II), with probability $1 - o(1)$, for every $U \subset V$, the graph $(U, E \cap (U \times U) \cap (S \times T))$ (i.e., the bipartite graph between pieces $U \cap S$ and $U \cap T$) is geometrically expanding up to scale $\sqrt{\log n \log(1/\eta)}$ with cut value

$$\begin{aligned} X &= C \max\{\text{sr-cost}(\mathcal{P}_{|U}, \varepsilon),\ n\sqrt{\log n \log(1/\eta)}(\log\log n)^2\} \\ &= C \max\{\varepsilon|S \cap U| \cdot |T \cap U|,\ n\sqrt{\log n \log(1/\eta)}(\log\log n)^2\}. \end{aligned}$$

Below, we assume that the graph $(U, E \cap (U \times U) \cap (S \times T))$ is geometrically expanding; otherwise our algorithm fails (which happens with probability $o(1)$ over the choice of $SR(\mathcal{P}, \varepsilon)$). Write the lower bound on $\varepsilon\eta$ and a trivial inequality on $\varepsilon|S \cap U| \cdot |T \cap U|$:

$$\begin{gathered} \sqrt{\log n \log(1/\eta)}(\log\log n)^2 \leq \varepsilon\eta n \leq \varepsilon \min(|S \cap U|, |T \cap U|)n; \\ \varepsilon|S \cap U| \cdot |T \cap U| \leq \varepsilon \min(|S \cap U|, |T \cap U|)n. \end{gathered}$$

Together these inequalities give us an upper bound on X:

$$X \leq C\varepsilon \min(|S \cap U|, |T \cap U|)n \leq C\varepsilon\rho|U|n.$$

By Theorem 5.3, the Small Set Expansion algorithm returns a cut of size $O(X)$ (Here we use that the graph $(U, E \cap (U \times U) \cap (S \times T))$ is geometrically expanding up to scale $\sqrt{\log n \log(1/\eta)} \geq \sqrt{\log n \log(1/\rho)}$).

We showed that the algorithm finds a cut of sparsity $\alpha = O(\varepsilon n)$ in expectation. By Markov's inequality, it finds a cut of sparsity at most 2α with probability at least $1/2$. So by repeating the algorithm many times and then picking the best solution, we can get a solution of cost at most 2α with probability exponentially close to 1.

Finally, let us briefly explain how to get rid of the $\sqrt{\log(1/\eta)}$ factor in the lower bound on $\varepsilon\eta$. Observe that it suffices for our algorithm to find a set A of size $|A| \in [\Theta(\rho|U|), |U|/2]$ i.e., we do not need a bound $|A| \leq O(\rho|U|)$. So we slightly modify the Small Set Expansion algorithm so that it works for smaller $\varepsilon\eta$, but possibly returns $|A| \gg \rho|U|$. In Case I of the algorithm (see Theorem 5.4), we use Theorem 2.1 (part I) instead of Theorem 2.1 (part II) of Bansal et al. [18] with $\rho = 1/2$. This algorithm returns a sparse cut of size at most $\rho|U| = |U|/2$ of sparsity $O(\sqrt{\log n})OPT$ (where OPT is the optimal sparsity of the cut). We repeatedly apply this algorithm and obtain disjoint sets $A_1, \ldots, A_T$. After we get a set A_i, we remove it from U. We stop when $|\cup A_t| \geq \rho|U|/4$. We let $A = \cup A_t$. It is not hard to show that the sparsity of A is at most $O(\sqrt{\log n})OPT$ (where OPT is the value of the sparsest cut in (U, E^+)) and $|A| \in [\Theta(\rho|U|), 1/2|U|]$. The proof is similar to the proof of Theorem 2.1 (part II) in [18]. □

5.5 Recovering the Partitions in the Planted Model

In the case of the Balanced Cut and Small Set Expansion problems, we can obtain better guarantees when the sets of the partition $\mathcal{P} = \{S, T\}$ have enough expansion within them. Note that to recover the planted partition, we need some conditions on the graph expansion inside $G[S]$ and $G[T]$: otherwise, there may exist a sparse cut in G cutting both S and T (for example, if the graphs $G[S]$ and $G[T]$ are random $G(n/2, \varepsilon)$ graphs, then the graph G is a $G(n, \varepsilon)$, and thus the sets S and T are indistinguishable from other sets of size $n/2$). This assumption is in the flavor of planted instances of Balanced Cut (or Small Set Expansion problem), where the cut given by the partition (S, T) is much sparser (sparser by a constant factor) than any cut inside the (adversarial) graph restricted to S or T. (This assumption is also similar to the stability assumption of Balcan, Blum, and Gupta [17] for clustering problems, where a c-factor approximation to the partitioning problem is $\eta(c)$-close to the target partition.) In this case, we can find the partition (S, T) up to $(1 + \eta)$-accuracy for some sub-constant $\eta > 0$ i.e., a partition differing from (S, T) in at most ηn vertices. We obtain these guarantees by repeatedly defining instances of the Sparsest Cut problem, and using our algorithms for the semi-random model to obtain increasingly finer approximations to the planted partition.

Definition 5.8. *Denote by $h(G)$ the expansion of the graph $G = (V_G, E_G)$:*

$$h(G) \equiv \min_{\substack{S \subset V_G \\ 0 < |S| \leq \frac{1}{2}|V_G|}} \frac{E(S, V_G \setminus S)}{|S|}.$$

Theorem 5.9. *There exists a randomized polynomial-time algorithm and positive absolute constants C, C_{exp}, such that for every set of vertices V of size n, every partition*

$\mathcal{P} = \{S, T\}$, $|S| = \rho n$ *(for* $\rho \in (0, 1/2]$*) and every* $\varepsilon \in (0, 1)$, $\eta \in (0, 1)$, *satisfying*

$$\eta \geq \frac{C\sqrt{\log n}(\log\log n)^2}{\varepsilon n},$$

the following statement holds with high probability over the random choices of $SR(\mathcal{P}, \varepsilon)$*: For every* $G = (V, E) \in SR(\mathcal{P}, \varepsilon)$ *satisfying* $h(G[S]) \geq C_{exp}\varepsilon n$ *and* $h(G[T]) \geq C_{exp}\varepsilon n$*, the algorithm given* G *and* ε*, returns a partition* (X, Y) *of* V *such that*

$$|X \triangle S| = |Y \triangle T| \leq \eta n \quad \text{or} \quad |X \triangle T| = |Y \triangle S| \leq \eta n.$$

Remark 1: The conditions $h(G[S]) \geq C_{exp}\varepsilon n$ and $h(G[T]) \geq C_{exp}\varepsilon n$ can be slightly relaxed, by requiring that only sets of size at least ηn expand in $G[S]$ and $G[T]$.
Remark 2: We assume that $\eta \leq \rho/3$. Otherwise, if $\rho \leq \eta$, then the trivial solution $(\varnothing, V)$ satisfies the conditions of the theorem. If $\eta \in [\rho/3, \rho]$, we may replace η with $\eta' = \rho/3$ and slightly change the absolute constant C.

Our algorithm relies on the Sparsest Cut algorithm for the semi-random model presented in Section 5.4. We denote the approximation factor of the Sparsest Cut algorithm by C_{SC} (see 5.6). We let $C_{exp} = 4C_{SC}$. We will use this algorithm for finding approximate sparsest cuts in $G[X]$ for various $X \subset V$ satisfying $|X \cap S|, |X \cap T| \geq \eta n/2$ (sometimes these conditions on X may be violated, then we assume that the algorithm returns a solution A, but the cut $(A, X \setminus A)$ may be arbitrarily bad). By Theorem 5.7, the Balanced Cut algorithm finds a cut of sparsity at most $C_{SC}\varepsilon n$ with probability exponentially close to 1 unless the graph G does not satisfy the "strong geometric expansion" property described in Theorem 4.8, part II. This happens with probability $o(1)$; and in this case, the partition recovering algorithm described below fails as well.

We introduce a potential function f that measures the quality of a partition (X, Y):

$$f(X, Y) = C_{SC}\varepsilon n \min(|X|, |Y|) - |E(X, Y)|. \tag{5.7}$$

The algorithm presented below tries to maximize f by finding non-expanding subsets A in X and moving them to Y and finding non-expanding subsets B in Y and moving them to X.

Algorithm. The algorithm first finds an approximate sparsest cut (X_0, Y_0) in G using the Sparsest Cut algorithm for semi-random graphs. Then, it repeats the following refinement procedure: find approximate sparsest cuts $(A, X_t \setminus A)$ in the graph $G[X_t]$ and $(B, Y_t \setminus B)$ in the graph $G[Y_t]$ using the Sparsest Cut algorithm for semi-random graphs and

- if $f(X_t \setminus A, Y_t \cup A) \geq f(X_t, Y_t) + \frac{1}{4}$, move A from X_t to Y_t i.e., set $X_{t+1} = X_t \setminus A$ and $Y_{t+1} = Y_t \cup A$; otherwise,
- if $f(X_t \cup B, Y_t \setminus B) \geq f(X_t, Y_t) + \frac{1}{4}$, move B from Y_t to X i.e., set $X_{t+1} = X_t \cup B$ and $Y_{t+1} = Y_t \setminus B$.

The order in which the algorithm considers the cases above does not matter. After each iteration the algorithm increases the counter t. The algorithm stops and outputs the cut (X_t, Y_t), when neither moving A from X to Y, nor moving B from Y to X increases $f(X, Y)$ by at least $\frac{1}{4}$.

Analysis. Notice that the number of iterations of the algorithm is polynomial, since $f(X, Y)$ is upper bounded by $C_{SC}\varepsilon n^2$, lower bounded by $-|E|$, and at every iteration (but last) f is increased by at least $\frac{1}{4}$. Thus, the algorithm runs in polynomial time. To prove that the algorithm works correctly, we need to show that the algorithm does not stop till (X_t, Y_t) is η-close to the planted solution (S, T) i.e., till $|X_t \triangle S| \leq \eta n$ or $|Y_t \triangle S| \leq \eta n$.

We first prove that $f(X_t, Y_t)$ is positive for every t. The Sparsest Cut algorithm finds a cut (X_0, Y_0) of sparsity at most $C_{SC}\varepsilon n$, hence $f(X_0, Y_0) \equiv C_{SC}\varepsilon n \min(|X_0|, |Y_0|) - |E(X_0, Y_0)| > 0$. Since the sequence $f(X_t, Y_t)$ is increasing, $f(X_t, Y_t)$ is positive for every t. Consequently, the sparsity of every cut (X_t, Y_t) is at most $C_{SC}\varepsilon n$.

We show that every relatively small set in G expands.

Claim 5.10. *For every set $U \subset V$ of size at most $2\rho n/3$, $E(U, V \setminus U) \geq C_{exp}\varepsilon n|U|/2$.*

Proof. Since $h(G[S]) \geq C_{exp}\varepsilon n$, we have

$$\begin{aligned} E(U \cap S, S \setminus (U \cap S)) &\geq C_{exp}\varepsilon n \cdot \min(|U \cap S|, |S \setminus (U \cap S)|) \\ &\geq C_{exp}\varepsilon n|U \cap S|/2, \end{aligned}$$

where the second inequality follows from $|U \cap S| \leq 2\rho n/3 \leq 2(|S| - |U \cap S|) = 2|S \setminus (U \cap S)|$. Similarly,

$$E(U \cap T, T \setminus (U \cap T)) \geq C_{exp}\varepsilon n|U \cap T|/2.$$

Thus, $E(U, V \setminus U) \geq C_{exp}\varepsilon n|U|/2$. □

As a corollary, we get that $|X_t| \geq 2\rho n/3$ and $|Y_t| \geq 2\rho n/3$ for every t (otherwise, the sparsity of the cut (X_t, Y_t) would be large). To argue that the Sparsest Cut algorithm finds a $C_{SC}\varepsilon n$ sparse cut in $G[X_t]$ or $G[Y_t]$, we need to prove the following claim.

Claim 5.11. *Suppose that the partition (X_t, Y_t) is not ηn close to the planted partition (S, T) i.e., $|X_t \triangle S| = |Y_t \triangle T| > \eta n$ and $|X_t \triangle T| = |Y_t \triangle S| > \eta n$, then one of the following two statements holds:*

- *$|X_t \cap S| \geq \eta n/2$ and $|X_t \cap T| \geq \eta n/2$; or*

- $|Y_t \cap S| \geq \eta n/2$ *and* $|Y_t \cap T| \geq \eta n/2$.

Proof. The set X_t is covered by S and T, and thus $|X_t \cap S| \geq |X_t|/2$ or $|X_t \cap T| \geq |X_t|/2$. Assume that $|X_t \cap S| \geq |X_t|/2$. Then $|X_t \cap S| \geq |X_t|/2 \geq \rho n/3 \geq \eta n$. If also $|X_t \cap T| \geq \eta n/2$, we are done. Otherwise, $|X_t \cap T| \leq \eta n/2$, and $|X_t \setminus S| = |X_t \cap T| \leq \eta n/2$. Consequently, $|Y_t \cap S| = |X_t \triangle S| - |X_t \setminus S| \geq \eta n - \eta n/2 = \eta n/2$. Also, $|Y_t \cap T| = |T| - |X_t \cap T| \geq \rho n - \eta n/2 \geq \eta n$.

The case $|T \cap X_t| \geq |X_t|/2$ is handled similarly. (Note that we have not used in the proof that $|S| \leq |T|$; we only used that $|T| \geq \rho n$.) □

Apply Claim 5.11 and suppose without loss of generality that $|X_t \cap S| \geq \eta n/2$ and $|X_t \cap T| \geq \eta n/2$. Then, the set X_t is partitioned in two pieces $X_t \cap S$ and $X_t \cap T$ each of size at least $\eta n/2$. Thus (as discussed in the beginning of the proof), the Balanced Cut algorithm finds a cut $(A, X_t \setminus A)$ (where $|A| \leq |X_t|/2$) of sparsity at most $C_{SC}\varepsilon n$. We now show that the cut $(A, V \setminus A)$ is large.

Claim 5.12. *Suppose that the graph G is partitioned into three non-empty sets U_1, U_2, U_3, then one of the sets U_i has large expansion: for some i,*

$$E(U_i, V \setminus U_i) \geq C_{exp}\varepsilon n|U_i|.$$

Proof. Observe that for one of the sets U_i, $|U_i \cap S| \leq |S|/2$ and $|U_i \cap T| \leq |T|/2$. For this set, $E(U_i \cap S, S \setminus U_i) \geq C_{exp}\varepsilon n|U_i \cap S|$ and $E(U_i \cap T, T \setminus U_i) \geq C_{exp}\varepsilon n|U_i \cap T|$. Hence, $E(U_i, V \setminus U_i) \geq C_{exp}\varepsilon n|U_i|$. □

Consider the partition of G into three sets $X_t \cap A$, $X_t \setminus A$ and Y_t. One of them has expansion $C_{exp}\varepsilon n$. It cannot be the set Y_t, since the expansion of Y_t is at most

$C_{SC}\varepsilon n$. Then,

$$\begin{aligned} E(X_t \setminus A, V \setminus (X_t \setminus A)) &\le E(X_t, Y_t) + E(X_t \setminus A, A) \\ &\le C_{SC}\varepsilon n|X_t| + C_{SC}\varepsilon n|A| \\ &\le 3C_{SC}\varepsilon n|X_t \setminus A| < C_{exp}\varepsilon n|X_t \setminus A|. \end{aligned}$$

Thus the set with expansion at least $C_{exp}\varepsilon n$ is A, that is, $E(A, V \setminus A) \ge C_{exp}\varepsilon|A|\, n$.

Estimate the change in the potential function f after moving A from X_t to Y_t:

$$\begin{aligned} f(X_t \setminus A, Y_t \cup A) - f(X_t, Y_t) &\ge -C_{SC}|A|\varepsilon n - \big(E(A, X_t \setminus A) \\ &- E(A, Y_t)\big) = -C_{SC}|A|\varepsilon n - E(A, X_t \setminus A) \\ &+ (E(A, V \setminus A) - E(A, X_t \setminus A)) = -C_{SC}|A|\varepsilon n \\ &- 2E(A, X_t \setminus A) + E(A, V \setminus A) \ge -C_{SC}|A|\varepsilon n \\ &- 2C_{SC}\varepsilon|A|\, n + \frac{3}{4}\, C_{exp}\varepsilon|A|\, n + \frac{1}{4}\, E(A, V \setminus A) \\ &= \frac{1}{4}\, E(A, V \setminus A) \ge \frac{1}{4}. \end{aligned}$$

Chapter 6

Graph Partitioning under Stability: Spectral expanders

In the previous chapters, we have seen how we can get much better approximation algorithms for basic graph partitioning problems in the average-case: the semi-random instances we considered in Chapter 4 and Chapter 5 have the property that the edges E_K crossing the boundaries of a *hidden partition* $\mathcal{P}$ satisfy a structural property (geometric expansion), which we can exploit.

In their recent exciting work, Balcan, Blum and Gupta [17] ask if we can design better algorithms, if we assume that even approximately optimal solutions are close to the target. To reiterate Question 3,

Question 3. *Can we design better approximation algorithms, if we assume that the instance has a unique optimal solution with stability?*

Balcan et al. [17] applied the stability assumption to metric clustering problems like k-means or k-median, and designed PTAS for these problems, when only $O(1)$ factor approximations were known in the worst-case [1]. One of the main questions they left open was

[1] In fact, the k-median problem is known to be APX-hard to approximate [84] in the worst-case

Question 4. *[17, 16] Can we design $O(1)$ approximations for graph partitioning problems like Sparsest Cut under the stability assumption?*

In Section 5.5, we saw how such a stability assumption in semi-random instances actually allows us to even *recover* the optimal solution (target) upto any accuracy. In this chapter, we will see how we can design $O(1)$ approximations for graph partitioning problems under some (strong) stability assumptions, even when the instance is not semi-random.

We obtain good approximation algorithms for Balanced Cut and Small Set Expansion, when the edges $\widetilde{E_K}$ not crossing the partition boundaries satisfy an algebraic expansion condition. In particular, if the expansion given by the planted partition $\mathcal{P}$ (whose edges are $\widetilde{E_K}$) is ε, then we need that the normalized algebraic expansion (see 2.9 for details) of the subgraphs inside the partitions (say, $G_{|P_1}$) is larger than ε. The famous Cheeger's inequality Lemma 2.10 relates the algebraic expansion to combinatorial expansion: when the combinatorial expansion of the subgraph $G_{|P_1}$ is large enough (stability), then they have sufficient algebraic expansion to fit into our setting.

This is a much weaker condition than edges of $\widetilde{E_K}$ being chosen independently at random. More crucially, in this case, the edges E_K can be arbitrary. Our algorithms are inspired by the results of [11, 68], where they infer global correlations between the vectors from local correlations and algebraic expansion. We first define our model more formally.

Definition ((same as 3.5) Planted Spectral Expander)**.** *We are given a graph $G = (V, E)$ on n vertices with a "planted" bisection $\mathcal{P} = \{P_1, P_2\}$ having $|P_1| \leq |P_2|$ (not known to the algorithm) of cut value at most εm and a parameter $C' > 1$. G is a Planted Spectral Expander if it satisfies:*

- *A subset of its edges $E_1 \subseteq E(G_{|P_1})$ has size m.*

- *the graph $G' = (P_1, E_1)$ is a regular expander with a (normalized) algebraic expansion*[2] $\lambda(G') > C'\varepsilon$.

Note that in the definition of a Planted Spectral Expander, for Balanced Cut we only need one of the subgraphs $G_{|P_1}$ or $G_{|P_2}$ to satisfy the required properties. We now present the $O(1)$-approximations in the Planted Spectral Expander model, when the parameter C' is a large enough constant. We will also see how this implies $O(1)$ factor approximations under strong stability assumptions.

6.1 Balanced Cut

We will in fact show the following stronger theorem:

Theorem 6.1 (Balanced Cut)**.** *There is a polynomial-time algorithm, that given a graph $G = (V, E)$ on n vertices with a "planted" bisection $\mathcal{P} = \{P_1, P_2\}$ (not known to the algorithm) of cut value εm, such that for some subset of edges $E_1 \subset E$ of size $|E_1| = m$, the graph $G_1 = (P_1, E_1)$ is a regular expander with a (normalized) algebraic expansion $\lambda(G_1) > 64\varepsilon$, finds a balanced cut of sparsity $O(\varepsilon)$.*

Proof. Consider the Balanced Cut SDP used in Section 4.4. Since this is a relaxation, the SDP value $SDP \leq \varepsilon m$. In particular,

$$\frac{1}{2}\,\text{sdp-cost}(u \to \bar{u}, E_1) \equiv \frac{1}{4}\sum_{(u,v)\in E_1} \|\bar{u} - \bar{v}\|^2 \leq \varepsilon m,$$

and, since $|E_1| = m$,

$$\frac{1}{4}\mathbb{E}_{(u,v)\in E_1}\left[\|\bar{u} - \bar{v}\|^2\right] = \frac{1}{4|E_1|}\sum_{(u,v)\in E_1} \|\bar{u} - \bar{v}\|^2 \leq \varepsilon.$$

[2] eigenvalue gap of the normalized Laplacian (see 2.9)

For the regular graph G_1, we have

$$\lambda(G_1) \equiv \min_{\{\bar{u}\}_{u\in V}} \frac{\mathbb{E}_{(u,v)\in E_1}\left[\|\bar{u}-\bar{v}\|^2\right]}{\mathbb{E}_{u,v\in P_1}\left[\|\bar{u}-\bar{v}\|^2\right]},$$

thus

$$\frac{1}{4}\mathbb{E}_{u,v\in P_1}\left[\|\bar{u}-\bar{v}\|^2\right] \leq \frac{\varepsilon}{\lambda(G_1)} < \frac{1}{64}.$$

Hence, there exists $u^* \in P_1$ such that $\mathbb{E}_{v\in P_1}\left[\|\bar{u}^*-\bar{v}\|^2\right] \leq 1/16$. Denote $d(u,v) = \|\bar{u}-\bar{v}\|^2$. By Markov's inequality,

$$|\operatorname{Ball}_d(u^*, \frac{1}{8}) \cap P_1| \geq \frac{|P_1|}{2} = \frac{n}{4}.$$

On the other hand, $|\operatorname{Ball}_d(u^*, 1/4)| < 4/5\ n$ (as shown in Section 4.4).

We are ready to describe the algorithm: The algorithm guesses the vertex u^* and picks a ball $S = \operatorname{Ball}_d(u^*, r)$ of radius $r \in [1/16, 1/4]$ around u^* with the smallest edge boundary. The cost of the cut $(S, V \setminus S)$ is at most $32 \cdot SDP$, and (since $\operatorname{Ball}(u^*, 1/16) \subset S \subset \operatorname{Ball}(u^*, 1/4)$),

$$\frac{n}{8} \leq |S| \leq \frac{4n}{5}.$$

□

This immediately leads to the following corollary:

Corollary 6.2 (Balanced Cut)**.** *There is a polynomial-time algorithm, that given a graph $G = (V, E)$ which is a planted spectral expander with parameter $C' > 64$, finds a balanced cut of sparsity $O(\varepsilon)$.*

The following corollary to Theorem 6.1 shows how we can obtain $O(1)$ approximations under strong stability assumptions.

Corollary 6.3 (Balanced Cut under Stability)**.** *There is a polynomial-time algorithm, that given a graph $G = (V, E)$ on n vertices with a "planted" bisection $\mathcal{P} = \{P_1, P_2\}$ (not known to the algorithm) of cut value εm, such that if for some subset of edges $E_1 \subset E$ of size $|E_1| = m$, the graph $G_1 = (P_1, E_1)$ is a regular expander with conductance $\Phi(G_1) > C\sqrt{128\varepsilon}$, finds a balanced cut of sparsity $O(\varepsilon)$.*

Proof. G_1 is a regular expander with $\Phi(G_1) > C\sqrt{128\varepsilon}$. By Cheeger's inequality Lemma 2.10, we have

$$\lambda(G_1) \geq \Phi(G_1)^2/2 > 64\varepsilon$$

as required for Theorem 6.1. □

6.2 Small Set Expansion

The algorithm and guarantees for Small Set Expansion follow a similar approach. Let P_1 be the side of the partition with the required properties. The main difficulty for Small Set Expansion is that the set S may contain vertices from P_2 and, moreover, it may cut many edges in $E(P_2 \cap S, V \setminus S)$. However, we overcome this issue using an additional Linear programming relaxation as in Section 5.3.

Theorem 6.4 (Small Set Expansion)**.** *There is a randomized polynomial-time algorithm, that given a graph $G = (V, E)$ with a "planted" partition $\mathcal{P} = \{P_1, P_2\}$ ($|P_1| = \rho n$) (not known to the algorithm) with $E(P_1, P_2) \leq \varepsilon m$ such that for some subset of edges $E_1 \subset V$ of size $|E_1| = m$, the graph $G_1 = (P_1, E_1)$ is a regular expander with a (normalized) algebraic expansion $\lambda(G_1) > 16\varepsilon$, finds a set S of size $\rho n/4 \leq |S| \leq 2\rho n$ with expected cost of the cut $O(\varepsilon m)$.*

Proof. Let $\{\bar{u}\}_{u\in V(G)}$ be the solution of the C-SDP for the Small Set Expansion considered in Section 5.3. Denote

$$SDP_{|P_1} = \text{sdp-cost}(u \to \bar{u}, E_1) \equiv \frac{1}{2}\sum_{(u,v)\in E_1} \|\bar{u}-\bar{v}\|^2.$$

Since SDP is P_1-local relaxation of $\mathcal{P}$, $SDP_{|P_1} \leq OPT \equiv \varepsilon m$ (see Lemma 5.5). We first proceed similarly to the proof of Theorem 6.1. Write,

$$\lambda \equiv \min_{\{\bar{u}\}_{u\in V}} \frac{\mathbb{E}_{(u,v)\in E_1}\left[\|\bar{u}-\bar{v}\|^2\right]}{\mathbb{E}_{u,v\in P_1}\left[\|\bar{u}-\bar{v}\|^2\right]},$$

then

$$\mathbb{E}_{u,v\in P_1}\left[\|\bar{u}-\bar{v}\|^2\right] \leq \frac{SDP_{|P_1}}{\lambda(G_1)} \leq \frac{OPT}{\lambda(G_1)} \leq \frac{1}{16}.$$

Hence, there is a vertex $u^* \in P_1$, such that

$$\mathbb{E}_{v\in P_1}\left[\|\bar{u}-\bar{v}\|^2\right] \leq 1/16.$$

Let $d(u,v) = \|\bar{u}-\bar{v}\|^2$. By the SDP spreading constraint (as shown in Section 5.3), $\text{Ball}_d(u^*, \frac{1}{4}) \leq 8/7\ \rho n$.

Thus, for some radius $r \in [1/16, 1/4]$ (the algorithm can guess u^* and r by considering all possibilities), the set $S = \text{Ball}_d(u^*, r)$ contains at least $|P_1|/2$ vertices from P_1, but at most $8/7\ \rho n$ vertices in total. Furthermore, the cost of the cut $E(P_1 \cap S, V \setminus S)$ is at most $32OPT$.

The main difference in this proof (as opposed to the proof of Theorem 6.1 is that the set S may contain vertices from P_2 and, moreover, it may cut many edges in $E(P_2 \cap S, V \setminus S)$. However, we have already dealt with a similar problem in Section 5.3. We use the LP (from the proof of Theorem 5.4, Case 2) to extract solution of cost at most $O(OPT)$ from S. The LP is feasible with LP value at most $O(OPT)$,

because one integral "canonical" solution exists: it is the set $S' = P_1 \cap S$. Indeed, $E(P_1 \cap S, P_2 \cap S) \leq E(P_1, P_2) \equiv OPT$, and thus $E(P_1 \cap S, V \setminus (P_1 \cap S)) \leq 33OPT$. $\square$

This immediately leads to the following corollary

Corollary 6.5 (Small Set Expansion)**.** *There is a polynomial-time algorithm, that given a graph $G = (V, E)$ which is a planted spectral expander that has a "planted" partition $\mathcal{P} = \{P_1, P_2\}$ ($|P_1| = \rho n$) and the parameter $C' > 64$, finds a set S of size $\rho n/4 \leq |S| \leq 2\rho n$ with expected cost of the cut $O(\varepsilon m)$.*

The following corollary of Theorem 6.4 shows how we can obtain $O(1)$ approximations under strong stability conditions.

Corollary 6.6 (Small Set Expansion under Stability)**.** *There is a randomized polynomial-time algorithm, that given a graph $G = (V, E)$ with a "planted" partition $\mathcal{P} = \{P_1, P_2\}$ ($|P_1| = \rho n$) (not known to the algorithm) with $E(P_1, P_2) \leq \varepsilon m$ such that for some subset of edges $E_1 \subset V$ of size $|E_1| = m$, the graph $G_1 = (P_1, E_1)$ is a regular expander with conductance $\Phi(G_1) > C\sqrt{128\varepsilon}$, finds a set S of size $\rho n/4 \leq |S| \leq 2\rho n$ with expected cost of the cut $O(\varepsilon m)$.*

Proof. G_1 is a regular expander with $\Phi(G_1) > C\sqrt{128\varepsilon}$. By Cheeger's inequality Lemma 2.10, we have

$$\lambda(G_1) \geq \Phi(G_1)^2/2 > 64\varepsilon$$

as required for Theorem 6.4. $\square$

Chapter 7

Densest k-subgraph: an Average-Case Study

In the last few chapters, we have seen how we can obtain better approximation guarantees by considering realistic average-case models : such algorithms show that these problems can be much easier in practice, than in the worst case. In the rest of thesis, we will focus on the DENSEST k-SUBGRAPH problem[1], a fundamental combinatorial problem which is notorious for its poor understanding from both algorithmic and inapproximability perspectives (please see Section 1.1.2 for details about its definition and history). Here, we will see that the guarantees that we can obtain in the worst-case will be identical to the best guarantees in the average-case. In fact, we will see how algorithms that we develop for a natural average case model can be translated systematically to algorithms for the worst-case which achieve the state-of-the-art algorithms for DENSEST k-SUBGRAPH: an approximation ratio of (roughly) $n^{1/4}$. This average-case study of DENSEST k-SUBGRAPH will also help capture the extent of current algorithmic approaches for the problem. This suggests that the approximability of the problem is similar from both worst-case and average-case perspectives.

[1] given a graph G and a parameter k, find the subgraph on at most k vertices with highest density (or the most number of edges)

Densest k-subgraph is closely related to two problems of very different complexity. If we remove the size restriction, we have the problem of finding the subgraph in G of maximum density — one that has a polynomial time algorithm [43, 27]. On the other hand, the k-clique problem [2] is a famous NP-hard problem (in fact, Max Clique can not be approximated within $n^{1-o(1)}$ factor unless $P = NP$). The main challenge for Densest k-subgraph seems to arise from the size constraint : the difficulty involved in identifying a *small* dense subgraph, in a graph which otherwise does not contain dense subgraphs. This inspires the average-case models we will study in this chapter.

The average-case models for DkS are reminiscent of the well-studied average-case variant of k-Clique : the Planted Clique problem [3, 41]. Here, we need to find a clique of size k that is planted inside a random graph drawn from $\mathcal{G}(n, 1/2)$ (the largest clique in $\mathcal{G}(n, 1/2)$ is of size $2 \log n$ with high probability). The best known algorithms use spectral techniques to identify planted cliques of size $\Omega(\sqrt{n})$.

In this chapter, we will first study average-case models for Densest k-subgraph, with increasing levels of generality, leading up to the Planted DkS model that forms the heart of our approach. We will then develop counting-based algorithms for the Planted DkS model, which will directly inspire our worst-case algorithms in Chapter 8.

7.1 Average Case Models for Densest k-subgraph

Random graphs have no dense subgraphs. In a $\mathcal{G}(n, p)$ random graph, every k-subgraph has density (average degree) $\tilde{O}(kp)$ with high probability. Any potential algorithm for DkS should be able to certify that random graphs have no dense k-subgraph, and better still identify a dense k-subgraph planted inside such a random

[2] The corresponding optimization problem Max Clique involves finding the largest clique in G

graph. This defines a natural average-case problem: the task is to *distinguish* between a random graph and a random graph with a dense k-subgraph planted inside it.

We will explore two different models, of increasing generality, depending whether the dense k-subgraph is random or arbitrarily chosen. The *Random model* is the simplest model, and it will motivate the notion of *log-density*, which will play a crucial role in understanding algorithms for DkS. This is analogous to the random planted models for graph partitioning [70]. Then, we will study the *Planted* DkS *model* and design the counting-based algorithms which will also form the basis for the worst-case algorithms. The Planted DkS model will allow adversarial choices in the choice of the dense k-subgraph, and hence, this can be seen as a semi-random model for DkS.

7.1.1 The Random Planted Model

The simplest setting is one where both the planted graph and the ambient graph are random:

Definition 7.1 (Random Planted Model)**.** *Distinguish between:*

- $\mathcal{D}_1$*: Graph* G *is picked from* $G(n,p)$*.*
- $\mathcal{D}_2$*:* G *is picked from* $G(n,p)$ *as before. A set* S *of* k *vertices is chosen arbitrarily, and the subgraph on* S *is replaced with a random graph* H *from* $\mathcal{G}(k,q)$ *on* S*.*

To distinguish these two distributions, one approach is look for constant size subgraphs L which act as 'witnesses'. If $G \sim \mathcal{D}_1$, we want that w.h.p. G will not have a subgraph isomorphic to L, while if $G \sim \mathcal{D}_2$, w.h.p. G should have such a subgraph.

Question 5. *For what parameters* p, q *does this approach work?*

Here, we exploit the well-known threshold phenomenon for presence of subgraphs in random graphs. If a graph $G \sim \mathcal{G}(n, p = n^{-\theta})$, standard probabilistic analysis (cf. [4]) shows that there exist constant size graphs H on ℓ_1 vertices and with ℓ_2 edges[3] (for example, K_5 is a witness for $\theta = -1/2$) such that if $\alpha < -\ell_1/\ell_2$, then G will not contain any copy of L (no subgraph isomorphic to L) w.h.p, whereas if $\alpha > \ell_1/\ell_2$, then G will have at least one copy of L. This motivates the following simple notion of *log-density*, which will play a crucial role in understanding the extent of algorithms for DkS over the next few chapters:

Definition 7.2 (Log-density)**.** *A graph on n vertices has log-density δ, if its average degree is n^{δ}.*

By using the above-outlined approach of counting constant-sized subgraphs, we can distinguish (with high probability) between the following two distributions whenever α, β are constants with $0 \leq \alpha < \beta \leq 1$:

1. $\mathcal{D}_1$: Random graph on n vertices of log-density α i.e. $\mathcal{G}(n, p = n^{\alpha}/n)$.

2. $\mathcal{D}_2$: Random graph on n vertices with some subgraph on k-vertices replaced by a random graph H of log-density β i.e. $\mathcal{G}(k, q = k^{\beta}/k)$ (this we refer to as the planted k-subgraph).

The algorithm will pick constants r, s with $\alpha < r/s \leq \beta$, use a witness on $(s - r)$ vertices and with s edges.

Observe that for $G \sim \mathcal{D}_1$, a k-subgraph would have expected average degree $kp = kn^{\alpha-1}$. Further, it can be shown that densest k-subgraph in G will have average degree $\max\{kn^{\alpha-1}, 1\}$, w.h.p. Thus if we can solve the above distinguishing problem Def. 7.1, its 'distinguishing ratio' would be $\min_{\beta}(k^{\beta}/\max\{kn^{\alpha-1}, 1\})$, where β ranges over all values for which we can distinguish (for the corresponding values of k, α). If

[3]Most "symmetric" graphs on ℓ_1 vertices, with ℓ_2 edges will work.

this is the case for all $\beta > \alpha$, then (as follows from a straightforward calculation), the distinguishing ratio is never more than

$$\begin{aligned}
\frac{k^\alpha}{\max\{kn^{\alpha-1}, 1\}} &= \min\left\{\left(\frac{n}{k}\right)^{1-\alpha}, k^\alpha\right\} \\
&= n^{\alpha(1-\alpha)} \cdot \min\left\{\left(\frac{n^{1-\alpha}}{k}\right)^{1-\alpha}, \left(\frac{k}{n^{1-\alpha}}\right)^{\alpha}\right\} \\
&\leq n^{\alpha(1-\alpha)} \\
&\leq n^{1/4}.
\end{aligned}$$

The algorithms for random planted model above, though interesting, does not lead to algorithms which work for the general DkS problem. In particular, if the planted k-subgraph were an arbitrary graph of log-density β, simply looking for the occurrence of subgraphs need not work, as the planted graph could be very dense and yet not have the subgraph we are looking for (for example, a complete bipartite graph will not have any triangles or odd-cycles). This leads us to the more general average-case model where the planted dense k-subgraph is adversarially chosen.

7.1.2 Planted DkS: Dense vs Random model

In the random planted model, we saw that when can detect higher log-densities i.e. a random graph of log-density δ has a random k-subgraph of higher log-density $> \delta$ (average degree $k^{\delta+\epsilon}$), then we can detect it. Here we ask:

Question 6. *Can we detect arbitrary k-subgraphs of higher log-density inside a random graph?*

This leads to the following natural average-case model:

Definition 7.3 (Planted DkS model)**.** *We are given n vertices and parameters D and d.*

1. *Generate* $G \sim \mathcal{G}(n, p = D/n)$.

2. *Choose an arbitary set of* k *vertices* S*, and replace them by an arbitrary subgraph* H *on* k *vertices, with average degree* d.

The Planted DkS model can be viewed as a semi-random model because of the adversarial choices incorporated in step 2. As before, we will try to solve the corresponding problem of distinguishing between a random graph and a random graph with an arbitrary k-subgraph of larger density. Since our aim is a $n^{1/4}$ approximation for DkS (or distinguishing ratio in this case), we will design algorithms for the following problem:

Definition 7.4 (Planted DkS: Dense vs Random)**.** *Given* n *and a constant* $\delta \in (0, 1)$*, distinguish between:*

- $\mathcal{D}_1$*: Random graph* G *of log-density* δ *i.e.* $\mathcal{G}(n, p = n^{\delta-1})$.

- $\mathcal{D}_2$*: Random graph* G *of log-density* δ *with arbitrary planted* k*-subgraph of log-density* $\geq \delta + \epsilon$ *for some small constant* $\epsilon > 0$ *i.e., first pick* G *according to* $\mathcal{D}_1$*, and then replace the subgraph on* k *of its vertices by an arbitary graph on* k *vertices with density* $d = k^{\delta+\epsilon}$.

We will design an algorithm for this distinguishing problem that uses a more clever counting of certain kinds of "witness structures". We will ensure that a large number of these witness structures will exists when there is any k-subgraph of log-density $> \delta$, whereas very few (polylogarthmic number) of them will exist in random graphs of log-density δ with high probability.

7.2 Algorithms for Planted Densest k-subgraph

We now present the algorithm which identifies the presence of k-subgraphs of higher log-density in the model we defined in Section 7.1.2. In fact our algorithms will work for a stronger model, as stated in the following theorem.

Theorem 7.5. *Let k be a given parameter which is $(\log n)^{\omega(1)}$, and $0 < \delta < 1$ be a (given) constant. Then for any $\epsilon > 0$, there exists a polynomial time algorithm that can distinguish between*

YES: G has a k-subgraph of density $k^{\delta+\epsilon}$, and

NO: G is a random graph drawn from $G(n,p)$, with $p = n^{\delta}/n$.

That is, we need a procedure that always outputs YES if G has a subgraph of the specified density, and outputs NO on almost all graphs (of specified degree). Note that we consider graphs with degree n^{δ} for $0 < \delta < 1$, so the theorem does not say anything about very dense graphs (like $\mathcal{G}(n, 1/2)$). To simplify the presentation, we may assume (please see Lemma 8.8) that the *minimum* degree in H is k^{β} as opposed to average degree (this will greatly simplify the counting argument).

To overcome the problem with the algorithm in the previous section, we will use a different kind of "local" witness that will involve 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 below a threshold 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 (or subgraph) with log-density above the same threshold, there exists a set of vertices U which coincide with the leaves of at least n^{ϵ} copies of T (for some constant $\epsilon > 0$). Let us now formally define what we mean by local counting.

Definition 7.6 (Witness Template)**.** *1. $\mathcal{W} = (W, L)$ is said to be a witness template if (a) W is a tree, and (b) L is the set of leaves (vertices of degree 1) of W.*

2. An occurrence *of $\mathcal{W}$ in graph G is a subgraph W' of G that is isomorphic to W. W' is said to be* supported on *the vertices $\mathcal{L} \subseteq V(W')$ which are the images of vertices in G under the isomorphism.*

3. $u \in G$ is a candidate for a node $w \in W$, if there exists an occurence of $\mathcal{W}$ such that u is the image of w in the isomorphism.

The particular witness templates we will use are *caterpillars* (trees of depth 1). Before they define them in general, let us look at the example when the log-density is 2/3.

Example: log-density $\delta = 2/3$. In this case, the template W we consider is the tree $K_{1,3}$ (a claw with three leaves). For any triple of vertices U, we count the number of copies of W with U as the set of leaves – in this case this is precisely the number of common neighbors of the vertices in U.

If $G \sim \mathcal{D}_1$, with $\delta \leq 2/3$ (i.e. $p = n^{-1/3}$)

$$\mathbb{E}\left[\text{Number of common neighbors for triple } U\right] = n \cdot p^3 = O(1)$$

Hence, using standard concentration bounds followed by a union bound over all triples, we can see that *every* triple of vertices has at most $O(\log n)$ common neighbors.

For the planted k-subgraph H, however, the log-density is $2/3 + \epsilon$ (average degree $d = k^{2/3+\epsilon}$). There are k^3 triples U supported in just H. The total number of

occurences W' of the structure, with all vertices in H is

$$\begin{aligned}\text{Number of occurences of } W &= \Omega(\sum_{u \in H} d_H(u)^3) \\ &= \Omega(k \cdot d^3) \quad \text{by convexity} \\ &= \Omega(k^{3+3\epsilon})\end{aligned}$$

Hence, there exists some triple in H with at least k^ϵ common neighbors. Since for ranges of parameters of interest $k^\epsilon = \omega(\log n)$, we have a distinguishing algorithm.

Witness Templates for general δ. Let us now consider a log-density threshold of r/s (for some relatively prime integers $s > r > 0$). The tree W we will associate with the corresponding template witness will be a caterpillar – a single path called the *backbone* from which other paths, called *hairs* or leaves, emerge. More formally,

Definition 7.7 (Witness $W(r, s)$)**.** *An* (r, s)-caterpillar *is a tree constructed inductively as follows:*

1. *Begin with a single vertex as the leftmost node in the backbone.*
2. *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).*

Let $B = \{b_1, b_2, \dots\}$ denote the vertices on the backbone.

Figure 7.1 on page 110 shows some examples of caterpillars for various values of δ. We also call the end-points of the hair steps as leaves, and the backbone vertices also as internal vertices. We now make a few simple, yet important observations:

Observation 7.8. *In the notation of Def. 7.7, we have*

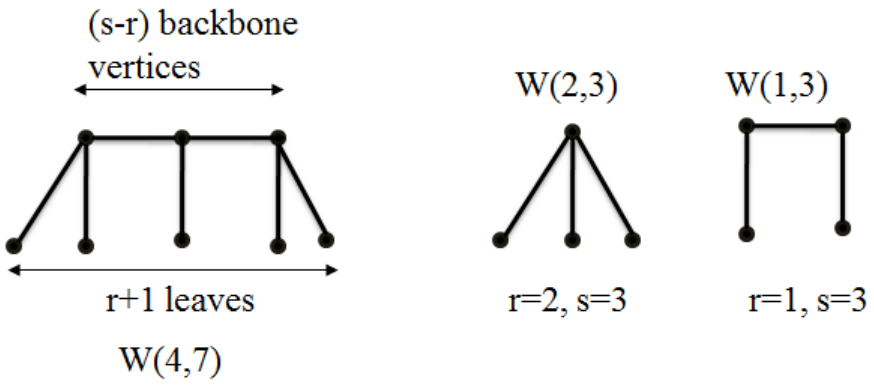

Figure 7.1: Caterpillar structures $W(r,s)$ for different values of r and s

- *The first and last steps are always hair steps.*
- *There are $r+1$ leaves (hair steps) and $s-r$ vertices in the backbone.*
- *The number of hair steps encountered in the first t steps is $\lfloor tr/s \rfloor + 1$.*

Lemma 7.9. *In the notation of Def. 7.7, the structure $W(r,s)$ is symmetric i.e. if t^{th} step is a hair step iff the $(s+1-t)^{th}$ step is a hair step.*

Proof. First, observe that this is true for $t = 1$, because the first and last steps are hair steps. Further, t is a hair step iff there is an integer a such that $(t-1)\cdot\frac{r}{s} < a < t\cdot\frac{r}{s}$. Then $(t-j)\cdot\frac{r}{s} < t+1-a < (t-j+1)\cdot\frac{r}{s}$. □

The distinguishing algorithm is in fact a set of algorithms which are parameterized by r, s. Given an input of log-density δ, we choose the algorithm corresponding to the witness template $W(r,s)$ satisfying $\delta \leq r/s < \delta + \epsilon$.

DistinguishRandom $DR_{r,s}$

- For all $\mathcal{L} \subseteq V(G)$ with $|\mathcal{L}| = r + 1$,
  - Count the number of occurrences of structures $W(r, s)$ in G which are supported on $\mathcal{L}$.
  - If this count is greater than $\log^{s-r} n$, declare YES.
- Else declare NO.

Note that the algorithm has a running time of n^{s-r}, which is polynomial for any fixed rational number δ.

These are not the only template witnesses that can be used for distinguishing for a particular δ, but they turn out to be useful when we consider the general case. Also, they have some properties that help in analyzing the distinguishing algorithm: for instance, they are 'symmetric' (see Lemma 7.9), which will help us bound the number of such structures in a random graph.

7.3 Analyzing the algorithm

To prove the correctness of the algorithm, we need to prove the following two theorems. The first one establishes that for the correct choice of r, s, for *every* choice of $(r + 1)$-sized tuple in the random graph will have few occurence of the template $W(r, s)$. This forms the bulk of the technical content to show the correctness of the algorithm. The second theorem shows that when there is a tuple in the dense k-subgraph which has many occurence of the witness $W(r, s)$.

Theorem 7.10. *Let G be a $\mathcal{G}(n, p)$ random graph with log-density δ ($p = n^{\delta-1}$) and let $\delta \leq r/s$. Then for* any *set $\mathcal{L} \subseteq V$ of size $r + 1$, the number of occurrences of $W(r, s)$ supported on $\mathcal{L}$ is at most $O(\log^{s-r} n)$ with high probability.*

Theorem 7.11. *Suppose H is a graph on k vertices with average degree $d \geq k^{\delta+\epsilon}$ for some constant $\epsilon > 0$. If $r/s < \delta + \epsilon$, then there exists a set $\mathcal{L}$ of size $r+1$ in H such that the number of occurrences of $W(r,s)$ supported on $\mathcal{L}$ is at least k^{ϵ}.*

Since we are interested in k super-logarithmic, when G has a dense enough k-subgraph, Theorem 7.11 ensures that the algorithm always outputs YES.

Let us start with some observations and definitions.

In what follows, $\text{fr}(x) = x - \lfloor x \rfloor$ is the fractional part of x.

We first show the proof of the simpler theorem, which gives the lower bound on the number of structures.

7.3.1 Proof of Theorem 7.11

The proof is by a simple counting argument. Note that we may assume that the *minimum* degree is $\Omega(d)$, because we can just remove the vertices of degree $< d/4$ and work with the remaining graph (see Lemma 8.8). For now suppose the minimum degree is d.

Let us count the total number of structures $W(r,s)$ in the graph. We do this as follows: view the tree as being rooted at b_1 – we have k choices for this, then we will move along the tree in a breadth first manner, filling the vertices. At an intermediate vertex v during this process, we have to choose $d(v) - 1$ vertices for the next level. Our structure needs distinct vertices of G, and since the degree of any vertex in G is at least d, there are at least $d - |r| > d/2$ neighbors that are not in any of the slots. So there are at least $(d/2)^{d(v)-1}$ ways of filling the slots neighboring v.

Proceeding this way, it follows that the number of tuples after fixing a root is at least $(d/2)^s$, where s is the number of edges in $W(r,s)$. Since s is a constant, it follows that the total number of structures is $\Omega(kd^s)$, and $kd^s = k \cdot k^{1+r+s\epsilon} > k^{r+1} \cdot k^{s\epsilon}$. Now $r+1$ is precisely the number of leaves in $W(r,s)$, and each tree is supported on one

of the k^{r+1} leaf tuples. Thus some tuple has $k^{s\epsilon}$ structures supported on it, finishing the proof. □

7.3.2 Proof of Theorem 7.10

Outline.

The structure $W(r,s)$ has a backbone of size $s-r$ and $r+1$ leaves (hairs) distributed among these $s-r$ backbone vertices (internal vertices). We bound the number of structures supported on any fixed set of leaves by showing that having fixed the set of leaves, the *number of candidates* for each internal vertex (backbone vertex) b_j is bounded by $O(\log n)$. Hence, for any fixing of the leaves (hairs), the number of copies of the template supported on it is $O(\log^{s-r} n)$.

We now concentrate on showing that for every fixing of all the leaves, the number of each internal vertex is bounded. We do this as follows: inductively, for any *prefix* Z_t (the first t steps) of the template $W(r,s)$ we will bound the "number of candidates" for the rightmost backbone vertex of Z_t after fixing the leaves of Z_t. We will show that this quantity is roughly $n^{\mathrm{fr}(tr/s)}$ with high probability. Since, the structure is symmetric (see Lemma 7.9), we can also achieve a similar bound from the right i.e. for every *suffix* of $W(r,s)$. The template witness structure is chosen such that for $\delta \leq r/s$, we have that these two upper bounds will combine to show that for every internal vertex, the number of candidates in $O(1)$ in expectation and $O(\log n)$ with high probability.

Upper bounding the number of candidates.

Let us fix the leaves to be $\mathcal{L} = \{v_0, v_1, \ldots, v_r\}$. From Observation 7.8, we know that among the first t steps, we have $\lfloor tr/s \rfloor$ hair steps (leaves).

Definition 7.12. *In the notation of Def. 7.7, given parameters r, s*

- *Prefix Z_t of the caterpillar template $W(r,s)$ is the subgraph of $W(r,s)$ defined by the first t steps of the inductive process given in Def. 7.7.*

- *For each $t = 1, \ldots, s$, we denote by $S(t)$ the set of candidates for the rightmost backbone vertex in prefix Z_t, after fixing the leaves of Z_t to be $v_0, v_1, \ldots, v_{\lfloor tr/s \rfloor}$.*

We begin by bounding the number of candidates for the rightmost backbone vertex of Z_t. However, counting naïvely would involve random variables which are not independent. We get around this by assuming that $V(G)$ is partitioned in to $V_1, \ldots, V_j, \ldots V_{(s-r)}$, and bound only the number of structures with the property that the backbone (internal) vertex b_j is from V_j. If there are many structures supported on $\mathcal{L}$, and the partitioning is done randomly, this number will also be large w.h.p. This is due to the "color coding" trick of Alon, Yuster and Zwick [5].[4]

In each *backbone step* the expected number of candidates increases by a factor of $np = D$, whereas for each *hair step*, the expected number of candidates goes down by a factor of $p = D/n$. The structure of the witness template was chosen in such a way that for $np = n^{r/s}$ (the correct log-density), the expected number of candidates $|S_{v_0,\ldots,v_{\lfloor tr/s \rfloor}}(t)| = n^{\mathrm{fr}(tr/s)}$. For $1 < t < s$, this number is always in $[n^{\epsilon'}, n^{1-\epsilon'}]$ for some small constant $\epsilon' > 0$ (since r and s are co-prime). The following lemma in fact shows that this number is concentrated around its expectation i.e. $n^{\mathrm{fr}(tr/s)}$:

Lemma 7.13 (Inductive upper bound on candidates)**.** *Suppose G is drawn from $G(n,p)$, with $p \leq n^{r/s-1}$. Then for every $1 \leq t \leq s$, every sequence of vertices $U_t = v_0, \ldots, v_{\lfloor tr/s \rfloor}$, we have with high probability (over the randomness in the input)*

$$|S_{v_0,\ldots,v_{\lfloor tr/s \rfloor}}(t)| \leq n^{\mathrm{fr}(tr/s)}(1 + o(1)). \tag{7.1}$$

[4]Roughly, the color coding theorem says the following: suppose we randomly color the vertices of a graph G with C colors. Suppose G has M copies of some structure $\mathcal{W}$ that has C vertices. Then w.h.p. there exist M/C^C 'colorful' copies of $\mathcal{L}$ (a copy is said to be colorful if each vertex in $\mathcal{W}$ has a different color).

Proof. We will show this inductively, like the construction of the template.

Base case $t = 1$. We know from Obs.7.8 that the first step is a hair (leaf) step. The prefix Z_1 consists of just one leaf and the internal vertex b_1. Now, after fixing the leaf to be v_0, the candidates b_1 are just neighbors of v_0.

Hence $|S_{v_0}(t = 1)| \leq (1 + o(1))np = (1 + o(1))n^{r/s}$, as required.

Inductive claim. Let us assume that the statement is true for all $t' \leq t - 1$.

We have two cases depending on whether step t is a hair step or a backbone step.

Case 1: Step t is a hair(leaf) step. Since step t is a hair step, the interval $[(t-1)r/s, tr/s]$ contains an integer: this has to be $\lceil (t-1)r/s \rceil$ (since $t \leq s$). The candidates $S(t)$ are precisely those candidates for $S(t-1)$ which are also neighbors of $v_{\lfloor tr/s \rfloor}$. Since the leaves are all disjoint, events involving the leaf $v_{\lfloor tr/s \rfloor}$ are independent of the all previous events (till step $t - 1$). Hence,

$$\Pr[u \in S(t)] \leq \Pr[u \in S(t-1)] \cdot \Pr\left[(u, v_{\lfloor tr/s \rfloor}) \in E(G)\right]$$

For $u \in S(t-1)$, let X_u denote the event that $(u, v_{\lfloor tr/s \rfloor}) \in E(G)$. Hence for $u \in S(t-1)$, $X_u = 1$ with probability p and 0 with probability $1 - p$. Further, $|S(t)| = \sum_u X_u$.

Each of the events X_u are independent, and the expectation of this quantity

$$\begin{aligned}
\mathbb{E}[|S(t)|] &= |S(t-1)| \cdot p \\
&\leq (1 + o(1))n^{\mathrm{fr}((t-1)r/s)} \cdot n^{r/s-1} \\
&= (1 + o(1))n^{\mathrm{fr}(tr/s)}
\end{aligned}$$

The last equality is true since the interval $[(t-1)r/s, tr/s]$ contains an integer.

As we have seen earlier, $n^{\mathrm{fr}(tr/s)} \in [n^{\Omega(1)}, n^{1-\Omega(1)}]$. Hence, by Chernoff bounds with high probability,

$$\sum_u X_u = |S(t)| \leq (1 + o(1))n^{\mathrm{fr}(tr/s)}$$

Case 2: Step t is a backbone step. Since step t is a backbone step, Z_t has the same set of leaves as Z_{t-1}. So, $\lfloor (t-1)r/s \rfloor = \lfloor tr/s \rfloor$ and $\mathrm{fr}(tr/s) = \mathrm{fr}((t-1)r/s) + r/s$. Let b_{j-1} and b_j be the righmost backbone vertices of Z_{t-1} and Z_t respectively.

Hence, the candidates $S(t)$ are precisely the neighbors of the candidates for the previous backbone vertex b_{j-1} i.e. $\Gamma\big(S(t-1)\big)$ which are in V_j.

For $u \in V_j$, let X_u denote the event that $\exists v \in S(t-1)$ such that $(u, v) \in E(G)$. Hence, if $X_u = 1$, u is in the candidate set $S(t)$.

$$\Pr[X_u = 1] = \Pr[\exists v \in S(t-1) : (u, v) \in E(G)]$$

By the color-coding trick, V_j is disjoint from the candidates for the previous backbone vertices, and in particular, $S(t-1)$.

$$\begin{aligned}
\Pr[X_u = 1] &\leq p \cdot |S(t-1)| \\
&\leq (1 + o(1))n^{r/s-1} \cdot n^{\mathrm{fr}((t-1)r/s)} \qquad \text{by inductive hypothesis} \\
&\leq (1 + o(1))n^{\mathrm{fr}(tr/s)-1} \quad \text{since } \mathrm{fr}(tr/s) = \mathrm{fr}((t-1)r/s) + r/s
\end{aligned}$$

Again, since the events $\{X_u\}_{u \in V_j}$ are independent events, we see that $\sum_u X_u$ concentrates around its mean, by Chernoff bounds. Hence with high probability,

$$|S(t)| \leq (1 + o(1))n^{\mathrm{fr}(tr/s)}$$

This completes the proof by induction. □

We know from Lemma 7.9 that the witness template is symmetric. If $S'(t')$ represents the set of candidate vertices for the position $s+1-t'$ after having fixed the leaves $v_{r-\lfloor(s+1-t')r/s\rfloor}, v_{r-1}, v_r$, then we have the following lemma by an identical proof:

Lemma 7.14. *Suppose G is drawn from $G(n,p)$, with $p \leq n^{r/s-1}$. Then for every $1 \leq t' \leq s$, every sequence of vertices $U_t = v_r, v_{r-1}, \ldots, v_{\lfloor(s+1-t')r/s\rfloor}$, we have with high probability (over the randomness in the input)*

$$|S'(t')| \leq n^{\mathrm{fr}(t'r/s)}(1+o(1)). \tag{7.2}$$

Completing the proof.

Let us denote by $C(j)$ the set of candidates for the backbone vertex b_j after having fixed *all* the leaves $v_0, v_1, \ldots v_r$. We will see that the expected number of candidates $C(j)$ is $O(1)$, and due to concentration, we can say that $C(j) = O(\log n)$ with high probability.

Let t be the largest integer such that b_{j-1} is the rightmost vertex in the prefix tree Z_t. Similarly, let $t' \in [1,s]$ be the largest integer such that b_{j+1} is leftmost vertex in the suffix $Z'_{s-t'}$, and let L_j be the fixing of the leaves attached to backbone vertex b_j. Clearly $s = t + |L_j| + t' + 2$ (s is the total number of steps).

For $u \in V_j$, let X_u be the indicator of the event u has an edge to each leaf in L_j and that u has a neighbor in both $S(t)$ and $S'(t')$. Since V_j is disjoint from the set of candidates for other backbone vertices, we have that

$$\begin{aligned}\Pr[X_u = 1] &= \Pr[\exists v \in S(t) : (u,v) \in E(G))] \cdot \Pr[\exists v \in S'(t') : (u,v) \in E(G)] \times \\ &\quad \times \Pi_{\ell \in L_j} \Pr[(u, \ell_j) \in E(G)] \\ &\leq p|S(t)| \cdot p|S'(t')| \cdot p^{|L_j|} \\ &\leq p^{|L_j|+2} \cdot |S(t)| \cdot |S'(t')|\end{aligned}$$

From Lemma 7.13 and Lemma 7.14, we have

$$\begin{aligned}|S(t)| &\leq n^{\mathrm{fr}(tr/s)}(1 + o(1)) \\ |S'(t')|] &\leq n^{\mathrm{fr}((t'r/s)}(1 + o(1)).\end{aligned}$$

Simplifying, we see that

$$\Pr[X_u = 1] \leq O(1)$$

Hence, $\mathbb{E}[|C(j)|] = O(1)$. Since the X_u are independent variables, by Chernoff bounds, taking a union bound over all fixing of the r leaves, we see that $C(j) = O(r \log n)$ w.h.p. Thus the number of structures is at most $(r \log n)^{s-r}$ w.h.p., thus finishing the proof of the theorem. □

This shows that if G is a random graph with density at most n^δ, then with high probability, every subset S of V of size $|G|$ has $O(\log^l n)$ structures supported on it.

Distinguishing factor.

Our algorithms distinguish random graphs of degree n^δ from graphs with k-subgraphs of density $k^{\delta+\epsilon}$. For fixed δ, we compute the distinguishing factor of our algorithms, i.e. the ratio between the densities of the optimal k-dense subgraph solutions in the YES and NO instances for which our algorithms succeed. For convenience, we ignore

log factors and ϵ here. For $k \leq n^{1-\delta}$, the optimal density in the random case is constant and the distinguishing ratio is $k^\delta \leq n^{\delta(1-\delta)}$. For $k = n^{1-\gamma}$ where $\gamma < \delta$, the optimal density in the random case is $n^{\delta-\gamma}$. Hence the distinguishing ratio is $k^\delta/n^{\delta-\gamma} = n^{\delta(1-\gamma)-\delta+\gamma} = n^{\gamma(1-\delta)} < n^{\delta(1-\delta)}$, which is at worst $n^{1/4}$.

7.3.3 SDP based approach

We have so far seen one way to distinguish random graphs of density n^δ from graphs with subsets of size k and density $k^{\delta+\epsilon}$. The natural question is if this is some kind of a lower bound. We now show that when k is sufficiently large ($k > \sqrt{n}$), a natural SDP can can detect k subgraphs with density just above $n^{\delta/2}$, which turns out to be smaller than k^δ.

We now consider a natural SDP relaxation for the densest k subgraph problem considered by Feige and Seltser [42], and show that this does better than the algorithm presented earlier when k is bigger than $\sqrt{n}$. The SDP is as follows

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{(i,j)\in E(G)} X_{ij} \quad \text{subject to} \\ & \textstyle\sum_i X_{ii} = k \\ & \textstyle\sum_j X_{ij} = kX_{ii} \qquad \text{for all } i \\ & X_{ij} \leq X_{ii} \qquad \text{for all } i,j \\ & X \succeq 0 \end{aligned}$$

This is a relaxation for the problem since it is easy to see that there exists an SDP solution of value $|E(H)|$, where H is a k-subgraph. We now show that the SDP value for a random graph is upper bounded by exhibiting a suitable dual solution.

Theorem 7.15. *For a random graph $G(n,p)$, the value of the SDP is at most $k(n^{\delta/2} + kn^{1-\delta})$ w.h.p.*

Proof. (of theorem 7.15) Let us consider the dual. We have the variables t, y_i, z_{ij} corresponding to the first, second and third constraints resp. The dual is

$$\text{minimize} \qquad kt \qquad \text{subject to}$$

$$U - A \succeq 0$$

$$U_{ii} = t - ky_i - \textstyle\sum_j z_{ij} \qquad \text{for all } i$$

$$U_{ij} = y_i + z_{ij} \qquad \text{for all } i, j$$

For a random graph, note that

$$\frac{d}{n}J - A + \lambda_2 I \succeq 0$$

where J is the all-ones matrix (It is easy to see this considering two cases: the all ones vector, and vectors orthogonal to it). Hence if we set $y_i = \frac{d}{n}$, $t = \lambda_2 + \frac{kd}{n}$ and the rest to 0, we get a feasible dual solution of value $\frac{k^2 d}{n} + k\lambda_2 \leq k^2 \frac{n^\delta}{n} + kn^{\delta/2}$. In the last step, we used the fact that the second eigenvalue is at most $\sqrt{np}$ w.h.p. This completes the proof. □

As an immediate corollary, we have

Corollary 7.16. *The SDP described above can be used to distinguish between a random graph $G(n,p)$, with $p = n^\delta/n$ and a graph with a k-subgraph of density $n^{\delta/2} + kn^\delta/n$.*

Proof. Observe that the second term is always smaller than k^δ, while the first is better for $k > \sqrt{n}$. Hence the SDP is better in this range. □

Chapter 8

Worst-Case algorithms for Densest k-subgraph

We now give a algorithm for worst-case DENSEST k-SUBGRAPH, which achieve the same guarantees as in the average-case setting (Section 7.1.2). In fact, this algorithm is directly inspired by the counting based algorithms in Section 7.2. We will actually use a family of algorithms, each parameterized by integers r, s (as in Section 7.2) and output the best k-subgraph found. The best algorithm will depend on the degree (roughly log-density) — for a graph G with maximum degree $D = n^{r/s}$ (log-density $\leq r/s$), we will use an algorithm corresponding to the (r, s)-caterpillar template as in Section 7.2. The worst-case algorithm (for $W(r, s)$) will also depend roughly on the number of occurences of the template witnesses $W(r, s)$. However, our objective is to find a dense k-subgraph and not just output YES/NO as in the distinguishing problem. To this end, we will instead use the candidate sets of vertices $S(t)$ (that was used in the analysis of our average-case algorithm in Section 7.3) in our algorithm to find the dense k-subgraph in the general case. To carry forth our intuition from the algorithms for average-case to the worst-case, we will use a linear programming

relaxation (see Section 8.2.1). Our algorithm will have worst-case approximation ratio of roughly $n^{1/4}$:

Theorem 8.1. *For any constant $\epsilon > 0$, there is an algorithm which gives a $O(n^{1/4+\epsilon})$ approximation for* Densest k-subgraph *and runs in time $O(n^{O(1/\epsilon)})$.*

In fact, these worst-case guarantees will match the distinguishing ratio of our average-case algorithm in Section 7.2. For the Planted DkS problem (average-case), we give a family of algorithms parameterized by r, s whose guarantees can be stated very succintly in terms of log-densities: in a graph G with log-density r/s (degree $n^{r/s}$), we can identify a k-subgraph H of log-density $r/s + \epsilon$ (degree $k^{r/s+\epsilon}$) in time $n^{O(r)}$. Our main technical contribution is that a result of this nature can be proven for the worst-case as well. We give a family of algorithms, parameterized by r, s whose *worst-case guarantees* can be informally stated as follows (see Theorem 8.16 for a more precise statement):

Theorem 8.2 (Informal)**.** *For relatively prime integers $0 < r < s$, given a graph G with maximum degree $D = n^{r/s}$, which contains a k-subgraph H with average degree d, our algorithm finds a k-subgraph of average degree $\Omega(d/D^{(s-r)/s})$ in time $n^{O(r)}$.*

While the above statement does not particularly resemble our average-case guarantees in terms of log-densities, we can perform some simple pre-processing of the graph to ensure that we restrict ourselves to the case when $kD \leq n$ (see Lemma 8.5), in which case we get the following statement:

Corollary 8.3 (Detecting Log-densities (informal))**.** *For relatively prime integers $0 < r < s$, given a graph G having degree $D = n^{r/s}$ such that $Dk \leq n$, which contains a k-subgraph H with average degree $d = k^{r/s+\epsilon}$, our algorithm runs in time $n^{O(r)}$ and finds a k-subgraph of average degree $\Omega(k^{\epsilon})$.*

Hence, even in worst-case, we can recover a k-subgraph of log-density ϵ, if a graph G of log-density δ has a k-subgraph H of log-density $\delta + \epsilon$[1] Before we proceed to our algorithm, we present a few simplifications (with proof), which will be useful for ease of exposition.

8.1 Simplifications

We present here a few simplifying assumptions which can be made with at most a constant factor loss in the approximation guarantee.

8.1.1 The Greedy Algorithm

It would be convenient for us to view G as a graph with maximum degree D rather than average degree D. For random graphs, maximum and average degrees are roughly the same. But for general graphs, they are not. In our context, it will be possible to think of D as the maximum degree, without losing any benefits that one may obtain by having D also be the average degree. The only use that we have for D being an average degree rather than maximum degree is that we can claim that G necessarily has a k-subgraph with average degree Dk/n (a random subgraph will do in expectation). The following argument, which is taken from [40] (Lemmas 3.2 and 3.3) allows us to make a related claim based on maximum degree rather than average degree.

In summary, we may assume that G has maximum degree D, and that nevertheless, one can extract a k-subgraph of average degree at least $\max\{Dk/n, 1\}$.

Lemma 8.4 (Greedy algorithm)**.** *There exists a polynomial time algorithm which, given a graph G containing a k-subgraph with average degree d, outputs a k-subgraph H' and an $(n - k/2)$-subgraph G' s.t.*

[1]While the theorem deals with the *maximum* degree in G instead of average degree (which defines the log-density), it turns out that this upper-bound on the log-density will suffice

1. *For some $D > 0$, G' has maximum degree $\leq D$ and H' has average degree* $\max\{\Omega(Dk/n), 1\}$.

2. *Either G' contains a k-subgraph with average degree $\Omega(d)$, or H' has average degree $\Omega(d)$.*

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 $U \cup U'$. Observe that H' has $\Omega(Dk^2/n)$ edges and average degree $\Omega(Dk/n)$ (it also has average degree at least 1, assuming G has at least $k/2$ edges).

Now let G' be the subgraph induced on $V \setminus U$ (note that by definition, G' has maximum degree $\leq D$). Let H be the densest k-subgraph in G, and let α be the fraction of edges of H that are incident with vertices of U. If $\alpha \leq 1/2$, then the $dk/2$ of the edges in H remain in G' (and thus G' still has a k-subgraph with average degree $\Omega(d)$. On the other hand, if $\alpha \geq 1/2$ then it is not hard to see that the H' must have average degree $\Omega(d)$. □

8.1.2 Bounding the product kD

It may simplify some of our manipulations if we can assume that $kD \leq n$. While we do not formally make this assumption anywhere, it is often instructive to consider this case for illustrative purposes. In fact, we can make this assumption, however at the cost of introducing randomness to the algorithm, and, more significantly, losing an $O(\sqrt{\log n})$ factor in the approximation guarantee.

Lemma 8.5. *Approximation algorithms for* DkS *lose at most a $O(\sqrt{\log n})$ factor by assuming the maximum degree of the input graph G has maximum degree D satisfying $kD \leq n$.*

Proof. Assume that contrary to our assumption, $D > n/k$. In this case, take a random subgraph G' of G by keeping every edge with probability $n/(kD)$. The new maximum degree D' now satisfies roughly $D' = Dn/(kD) = n/k$ as desired. A standard probabilistic argument (involving the combination of a Chernoff bound and a union bound) shows that for every vertex induced subgraph, if d denotes its average degree in G, then its average degree in G' is at most $O((1 + dD'/D)(1 + \sqrt{\log n/(1 + dD'/D)}))$.

Let ρ be the approximation ratio that we are aiming at. For H, the densest k-subgraph, we may assume that $d \geq \rho\sqrt{\log n}kD/n$, as otherwise the greedy algorithm in Lemma 8.4 provides a $\rho\sqrt{\log n}$ factor approximation. (This is where our argument loses more than a constant factor in the approximation ratio). Hence the average degree of H in G' is at least $\rho\sqrt{\log n}$. An approximation algorithm with ratio ρ will then find in G' a subgraph of average degree at least $\sqrt{\log n}$. Such a subgraph must have average degree a factor of D/D' higher in G, giving an approximation ratio of ρ. □

8.1.3 Other simplifications

We now justify the weakened requirement, that the algorithm should return a subgraph of size *at most* k (rather than exactly k).

Lemma 8.6. *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). □

Also, sometimes we return a subgraph of size $2k$ instead of k, but this only leads to a loss of a constant factor by sampling k vertices from it.

We now treat the assumption that G (and hence H) is bipartite:

Lemma 8.7. *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. □

Finally, we note that it suffices to consider minimum degree in H, rather than average degree.

Lemma 8.8. *Approximation algorithms for* DkS *lose at most a constant factor by assuming the densest k-subgraph has minimum degree d as opposed to average degree.*

Proof. Let H be the densest k-subgraph in G. Iteratively remove from H every vertex whose remaining degree into H is less than $d/2$. Since less than $kd/2$ edges

are removed by this procedure, a subgraph on $k' \leq k$ vertices and minimum degree at least $d/2$ must remain. As noted in Lemma 8.6, the fact that now we will be searching for subgraphs with $k' \neq k$ vertices can affect the approximation guarantee by at most a constant factor. □

8.2 The algorithm

Our worst-case algorithm for DkS mimicks our inductive proof in Lemma 7.13. We can imagine our algorithm as trying to show that the input graph G has no dense k-subgraph (no k-subgraph of higher log-density than G), by simulating the proof of Lemma 7.13. The steps of our algorithm correspond to the s steps in the inductive proof for the template $W(r, s)$.

We are given a graph G which contains a k-subgraph H with average degree d. Our goal is to find a k-subgraph of average degree ρ (which is $\Omega(d/D^{(s-r)/s})$). Let us first recall the definition of $S(t)$ from Section 7.3:

Definition 8.9 (Same as Def. 7.12)**.** *In the notation of Def. 7.12, given a graph G for each $t = 1, \ldots, s$, we denote by $S(t)$ the set of candidates in G for the rightmost backbone vertex in prefix of $W(r, s)$ defined by the first t steps, after fixing the leaves (hair steps) encountered so far to be $v_0, v_1, \ldots, v_{\lfloor tr/s \rfloor}$.*

We will use S to represent $S(t)$ when it is clear that we are at step t of the algorithm.

Our algorithm is roughly based on the following idea. In step t of $Alg_{r,s}$, we try to find a dense k-subgraph of sufficient density (by rounding a linear program) : if we succeed, we just output this k-subgraph, otherwise we show that G also satisfies the same upper bounds for candidate sets $S(t)$ as in the average-case (Lemma 7.13). However, these bounds can not hold for all of the s steps, since Theorem 7.11 tells us that if G contains any subgraph H of higher log-density, the number of candidates

$S(s)$ (for the rightmost backbone vertex) from just H after the correct fixing of all the leaves has to be large. Hence we should have found a dense k-subgraph in one of s steps from the rounding algorithm.

To carry forth this idea, we would like to inductively keep track of the size of the candidate set $S(t)$, and on the candidates (lower-bound) from the dense subgraph H (which is unknown to us, of course). The right quantity to look at is the ratio $|S(t) \cap H|/|S(t)|$. At the start of each step $t+1$ of the algorithm, we will have a lower bound on $|S(t-1) \cap H|/|S(t-1)|$. Depending on whether step t of $W(r,s)$ is a *backbone step* or a *hair step*, we will use a slightly different procedure. In case we do not find a k-subgraph of sufficient density, we will inductively obtain a lower bound on $|S(t+1) \cap H|/|S(t+1)|$ if it is a backbone step. If it is a hair step, we will find a good choice for the leaf (hair vertex) so that we can obtain the correct lower bound for $|S(t+1) \cap H|/|S(t+1)|$ (as dictated by our average-case intuition).

To this end, we will need a linear program which allows us to keep track of counts (like $|S(t) \cap H|$) even after fixing (conditioning on) some of the leaves. These are perfectly captured by linear programs that are obtained by systematic lift-and-project relaxations.

8.2.1 The Linear Program relaxation

Let us start by describing the LP relaxation.

Taking into account the simplification from Lemma 8.8, we can assume that there is a subgraph H of size at most k with minimum degree d. Hence, we get the following simple LP relaxation for DkS. While the program as stated is not linear, we guess the degree d and consider the feasibility linear program that is obtained.

We can define a hierarchy of LPs which is satisfied by a graph which contains a subgraph of size at most k with minimum degree at least d. This hierarchy is obtained

$$
\begin{aligned}
&\max \quad d \\
&\text{s.t.} \sum_{i \in V} x_i \leq k, \qquad \text{and} \\
\exists \{x_{ij} \mid i,j \in V\} &\text{ s.t.} \\
\forall i \in V \quad &\sum_{j \in \Gamma(i)} x_{ij} \geq d x_i \\
\forall i,j \in V \quad &x_{ij} = x_{ji} \\
\forall i,j \in V \quad &0 \leq x_{ij} \leq x_i \leq 1
\end{aligned}
$$

Figure 8.1: Min degree LP for DkS

from a subset of constraints of the Sherali-Adams relaxation for DkS in Figure 9.2 on page 148.

$$
\begin{aligned}
&\max \quad d, \text{ s.t.} \\
&\exists \{x_T \mid T \subseteq V, |T| \leq r\} \text{ s.t. } x_\emptyset = 1 \text{ and} \\
\forall T \subseteq V \text{ s.t } |T| \leq r-2 &: \\
&\sum_{i \in V} x_{T \cup \{i\}} \leq k x_T \qquad (8.1) \\
\forall i \in V \quad &\sum_{j \in \Gamma(i)} x_{T \cup \{i,j\}} \geq d x_{T \cup \{i\}} \qquad (8.2) \\
&0 \leq x_{T \cup \{i\}} \leq x_T \leq 1 \qquad (8.3)
\end{aligned}
$$

In the intended solution, x_T is 1, if all the vertices in T belong to the dense subgraph H. This LP allows us to condition on upto $r-2$ variables : when $x_T \neq 0$, $\{\frac{x_{T \cup \{i,j\}}}{x_T}\}_{i,j \in V(G)}$ form a valid solution to the basic LP in Figure 8.1 on page 129, which represents the LP solution after conditioning on the event that the vertices of T were chosen to be in our dense k-subgraph.

The exact linear programming relaxation that we need is the one presented below in Figure 8.2 on page 130. It uses a recursive definition which which will be easier for exposition. This hierarchy is at most as strong as the Lovász-Schrijver LP hierarchy based on the usual LP relaxation and is possibly weaker (see [31] for details). Specifically, for all integers $t \geq 1$, we define $\mathbf{DkS\text{-}LP}_t(G, k, d)$ to be the set of n-dimensional vectors $(y_1, \ldots y_n)$ satisfying: Given an LP solution $\{y_i\}$, we write

$$\sum_{i \in V} y_i \leq k \quad \text{and} \tag{8.4}$$

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

$$\forall i \in V \quad \sum_{j \in \Gamma(i)} y_{ij} \geq d y_i \tag{8.5}$$

$$\forall i, j \in V \quad y_{ij} = y_{ji} \tag{8.6}$$

$$\forall i, j \in V \quad 0 \leq y_{ij} \leq y_i \leq 1 \tag{8.7}$$

$$\text{if } t > 1, \ \forall i \in V \text{ s.t. } y_i \neq 0, \text{ we have}$$

$$\{y_{i1}/y_i, \ldots, y_{in}/y_i\} \in \mathbf{DkS\text{-}LP}_{t-1}(G, k, d) \tag{8.8}$$

Figure 8.2: Recursive definition of DkS-LP$_t(G, k, d)$

$\mathrm{LP}_{\{y_i\}}(S) = \sum_{i \in S} y_i$. When the solution is clear from context, we denote the same by $\mathrm{LP}(S)$. We call this the *LP-value* of S. When the level in the hierarchy will not be important, we will simply write DkS-LP instead of DkS-LP$_t$. A standard argument shows that a feasible solution to DkS-LP$_t(G, k, d)$ (along with all the recursively defined solutions implied by constraint (8.8)) can be found in time $n^{O(t)}$.

As before, we can think of the LP as giving a distribution over subsets of V, with y_i being the probability that i is in a subset. Similarly y_{ij} can be thought of as the probability that both i, j are 'picked'. We can now think of the solution $\{y_{ij}/y_i : 1 \leq j \leq n\}$ as a distribution over subsets, conditioned on the event that i is picked.

8.2.2 The Algorithm description

Let us now describe the algorithm in detail. The algorithm will take two kinds of steps, *backbone* and *hair*, corresponding to the two types of caterpillar edges. While these steps differ in the updates they make, both use the same procedure to search locally for a dense subgraph by looking at the current candidate set $S(t)$ and its set of neighbors $\Gamma(S(t))$. Let us now describe this procedure. From now onwards, we will use S to refer to $S(t)$ when we are at step t of the algorithm.

DkS-Local(S, k)

- Consider the bipartite subgraph induced on $(S, \Gamma(S))$.
- For all $k' = 1, \ldots, k$, do the following:
  - Let $T_{k'}$ be the set of k' vertices in $\Gamma(S)$ with the highest degree (into S).
  - Take the $\min\{k', |S|\}$ vertices in S with the most neighbors in $T_{k'}$, and let $H_{k'}(S)$ be the bipartite subgraph induced on this set and $T_{k'}$.
- Output the subgraph $H_{k'}(S)$ with the largest average degree.

We note that **DkS-Local**(S, k) can be just be executed for $k' = \lceil LP(\Gamma(S)) \rceil$ (the proof of Claim 8.10 is unchanged). While this would speed up the algorithm, we present it as above to highlight that **DkS-local** is just a simple greedy algorithm.

The overall algorithm takes as input a graph G, a parameter k, and $\{y_i\}$, a solution to DkS-LP$_{r+2}(G, k, d)$. We will in fact use a family of algorithms $Alg_{r,s}$ parameterized by positive integers r, s ($r < s$ and $gcd(r, s) = 1$), which will correspond to the caterpillar template $W(r, s)$. Throughout, a set $S \subseteq V$, and an LP solution $\{y_i\}$ are maintained. The description is not complete, since one of the steps depend on the description of Lemma 8.14.

DkS-Cat$_{r,s}(G, k, \{y_i\})$

- Let $S_0 = V$.
- For all $t = 1, \ldots, s$, do the following:
  - For $t > 1$, let H_t be the output of Procedure DkS-Local$(S(t-1), k)$.
  - If $[(t-1)r/s, tr/s]$ contains an integer, perform a **hair step**:
    Choose some vertex j_t given by Lemma 8.14 (or for $t = 1$, choose any j_1 such that $y_{j_1} > 0$), and
    - Let $S(t) = S(t-1) \cap \Gamma(j_t)$.
    - Replace the LP solution $\{y_i\}$ with $\{y_{ij_t}/y_{j_t} \mid i \in V\}$.
  - Otherwise, perform a **backbone step**:
    Let $S(t) = \Gamma(S(t-1))$.
- Output the subgraph H_t with the highest average degree.

Note that since the "conditioning" step (replacing y_i's by y_{ij}/y_j) in the hair steps is only performed $r+1$ times, then by constraint (8.8), at every step of the algorithm $\{y_i\}$ satisfies DkS-LP(G, k, d).

To summarize, the final algorithm first performs the pre-processing Section 8.1 (especially Section 8.1.1) and then runs **DkS-Cat**$_{r,s}$ for the correct value of r, s (such that maximum degree after preprocessing $D \leq n^{r/s}$)[2]. Then we return the denser of the subgraphs found by the Greedy algorithm of Section 8.1.1 and **DkS-Cat**$_{r,s}$. The running time is dominated by the time required to solve the LP, which as we saw earlier is $n^{O(r)}$.

[2]We can also run the algorithm for several values of co-prime r, s and output the densest subgraph found.

8.3 Analysis of the algorithm

8.3.1 The Backbone and Hair steps

We will analyze separately the performance of the procedure **DkS-Local** in the context of a hair-step (leaf) and that of a backbone-step.

It is useful to have in mind the case when $kD \leq n$. From Lemma 8.5, we know that this assumption is without loss of generality upto a $O(\log n)$ factor. However, this simplification gives more intuition to the bounds presented and draws a better parallel to the average-case. To that end, we also present the bounds for the case when $kD \leq n$ as corollaries. We begin by relating the performance of this procedure to an LP solution. In the analysis, we will ignore the roundoff error from $LP(\Gamma(S))$ since it affects the bounds neglibly in the context of the algorithm (we lose a small constant factor in total).

Claim 8.10. *Given a set of vertices $S \subseteq V$, and an LP solution $\{y_i\} \in$ DkS-LP(G, k, d). Then* DkS-Local(S, k) *outputs a subgraph with average degree at least*

$$\frac{1}{\max\{|S|, \text{LP}(\Gamma(S))\}} \cdot \sum_{j \in \Gamma(S)} y_j |\Gamma(j) \cap S|.$$

Proof. By constraint (8.4), $\lceil \text{LP}(\Gamma(S)) \rceil \leq k$. Consider the iteration where $k' = \lceil LP(\Gamma(S)) \rceil$. In Procedure DkS-Local, the vertices in $T_{k'}$ must have at least $\sum_{j \in \Gamma(S)} y_j |\Gamma(j) \cap S|$ edges to S (since $T_{k'}$ is the best k' in $\Gamma(S)$). After choosing the $\min\{k', |S|\}$ vertices in S with highest degree, the remaining subgraph $H_{k'}(S)$ has average degree at least

$$\frac{\min\{k', |S|\}}{|S|} \cdot \frac{1}{k'} \cdot \sum_{j \in \Gamma(S)} y_j |\Gamma(j) \cap S|.$$

This proves the claim. □

The backbone step in the algorithm first performs DkS-Local on the current S, and then sets S to be $\Gamma(S)$. The following lemma gives a way to inductively maintain a lower bound on $\mathrm{LP}(S(t))/|S(t)|$ assuming DkS-Local does not find a sufficiently dense subgraph.

Lemma 8.11 (Backbone step)**.** *Given $S \subseteq V$, and an LP solution $\{y_i\}$ for* DkS-LP(G, k, d)*: for any $\rho \geq 1$ such that* $\mathrm{LP}(S)/|S| \geq \rho/d$*, either* DkS-Local$(S, k)$ *outputs a subgraph with average degree at least ρ or we have*

$$\mathrm{LP}(\Gamma(S)) \geq \frac{d\mathrm{LP}(S)}{\rho} \tag{8.9}$$

$$\mathrm{LP}(\Gamma(S))/|\Gamma(S)| \geq \frac{d}{\rho D}\mathrm{LP}(S)/|S| \tag{8.10}$$

We immediately have the following corollary when $kD \leq n$:

Corollary 8.12 (Backbone step when $kD \leq n$)**.** *We are given a graph G with maximum degree $D = n^{r/s} = n/k$, having a k-subgraph H of minimum degree $d = \rho k^{r/s}$.*

If we have $S \subseteq V$, and an LP solution DkS-LP(G, k, d) *s.t.* $\mathrm{LP}(S)/|S| \geq \rho/d$*, either* DkS-Local$(S, k)$ *outputs a k-subgraph with average degree at least ρ or*

$$\mathrm{LP}(\Gamma(S)) \geq k^{r/s}\mathrm{LP}(S) \tag{8.11}$$

$$\mathrm{LP}(\Gamma(S))/|\Gamma(S)| \geq \frac{k^{r/s}}{n^{r/s}}\mathrm{LP}(S)/|S| \tag{8.12}$$

Proof of Lemma 8.11. We note that the second part follows from the first part by combining it with the knowledge that the maximum degree is D: hence $|\Gamma(S)| \leq D|S|$.

By the LP constraints (8.7) and (8.5), we have

$$\begin{aligned}\sum_{j \in \Gamma(S)} y_j |\Gamma(j) \cap S| &\geq \sum_{j \in \Gamma(S)} \sum_{i \in \Gamma(j) \cap S} y_{ij} \\ &= \sum_{i \in S} \sum_{j \in \Gamma(i)} y_{ij} \geq d\mathrm{LP}(S).\end{aligned}$$

By Claim 8.10, Dks-Local(S,k) outputs a subgraph with average degree $\geq$ $d\text{LP}(S)/\max\{|S|,k'\}$, where $k' = \text{LP}(\Gamma(S))$.

This suffices if $k' \leq |S|$, since by our assumption $d\text{LP}(S)/|S| \geq \rho$.

Now suppose $k' \geq |S|$. The output graph has average degree at least $d\text{LP}(S)/k'$. If this is at least ρ, then we have output a ρ-dense subgraph, as required.

If not, $k' = LP(\Gamma(S)) \geq d\text{LP}(S)/\rho$, which gives our required bound. □

Let us now consider a hair step. In this case, the algorithm performs DkS-Local on the current set, and then picks a vertex $j \in V$ to act as a "leaf". The new S is then set to equal $S \cap \Gamma(j)$. The following lemmas prove that either DkS-Local finds a sufficiently dense subgraph, or we can pick j so as to inductively maintain certain bounds. Let us first prove a simple averaging lemma.

Lemma 8.13. *Let x_j, $(1 \leq j \leq n)$ be reals in $[0,1]$, with $\sum_j x_j \leq k$. Let P_j and Q_j be some non-negative real numbers such that for some $P, Q > 0$,*

$$\sum_j x_j P_j \geq P, \text{ and } \sum_j x_j Q_j \leq Q. \tag{8.13}$$

Then there exists a j such that $P_j \geq P/(2k)$ and $P_j/Q_j \geq P/(2Q)$.

Proof. By our assumption $\sum_j x_j\big(P_j - \frac{P}{2k}\big) \geq P - \frac{P}{2} = \frac{P}{2}$. Thus from (8.13), it follows that there exists a j such that $x_j > 0$ and

$$P_j - \frac{P}{2k} \geq \frac{P}{2Q} \cdot Q_j.$$

This choice of j clearly satisfies the required properties. □

Lemma 8.14 (Hair step lemma)**.** *Let $S \subseteq V$, and let $\{y_i\} \in$ DkS-LP(G,k,d) be an LP solution (for which there exist corresponding $\{y_{ij}\}$). Then for any $\rho \geq 1$, either* DkS-Local(S,k) *outputs a ρ-dense subgraph, or there exists some vertex $j \in G$, such*

that $y_j > 0$*, and*

$$\mathrm{LP}_{\{y_{ij}/y_j | i \in V\}}(S \cap \Gamma(j)) \geq \frac{d \cdot \mathrm{LP}_{\{y_i\}}(S)}{2k} \tag{8.14}$$

$$\frac{\mathrm{LP}_{\{y_{ij}/y_j | i \in V\}}(S \cap \Gamma(j))}{|S \cap \Gamma(j)|} \geq \frac{d}{2\rho} \cdot \frac{\mathrm{LP}_{\{y_i\}}(S)}{\max\{k, |S|\}} \tag{8.15}$$

We immediately have the following corollary when $kD \leq n$:

Corollary 8.15 (Hair step when $kD \leq n$)**.** *Suppose graph* G *has maximum degree* $D = n^{r/s} = n/k$*, and a* k*-subgraph* H *of minimum degree* $d = \rho k^{r/s}$*.*

If $S \subseteq V$*, and* $\{y_i\} \in \text{DkS-LP}(G, k, d)$ *is an LP solution (with corresponding* $\{y_{ij}\}$*), then either* $\text{DkS-Local}(S, k)$ *outputs a* ρ*-dense* k*-subgraph, or there exists some vertex* $j \in G$*, such that* $y_j > 0$*, and*

$$\mathrm{LP}_{\{y_{ij}/y_j | i \in V\}}(S \cap \Gamma(j)) \geq \frac{\rho k^{r/s}}{2k} \cdot \mathrm{LP}_{\{y_i\}}(S) \tag{8.16}$$

$$\frac{\mathrm{LP}_{\{y_{ij}/y_j | i \in V\}}(S \cap \Gamma(j))}{|S \cap \Gamma(j)|} \geq \frac{k^{r/s}}{2} \cdot \frac{\mathrm{LP}_{\{y_i\}}(S)}{\max\{k, |S|\}} \tag{8.17}$$

Proof of Lemma 8.14. By constraints (8.5) of the LP we have

$$\begin{aligned}
\sum_{j \in \Gamma(S)} y_j LP_{\{y_{ij}/y_j | i \in V\}}(S \cap \Gamma(j)) &= \sum_{j \in \Gamma(S)} \sum_{i \in \Gamma(j) \cap S} y_{ij} \\
&= \sum_{i \in S} \sum_{j \in \Gamma(i)} y_{ij} \\
&\geq d\mathrm{LP}_{\{y_i\}}(S)
\end{aligned} \tag{8.18}$$

From Claim 8.10, it follows that if the subgraph found by DkS-Local has average degree less than ρ, we must have $\rho > \sum_{j \in \Gamma(S)} y_j |\Gamma(j) \cap S| / \max\{|S|, k'\}$, or in other words

$$\sum_{j \in \Gamma(S)} y_j |\Gamma(j) \cap S| \leq \rho \max\{|S|, k'\} \leq \rho \max\{|S|, k\}. \tag{8.19}$$

By using Lemma 8.13 with $P_j = LP_{\{y_{ij}/y_j | i \in V\}}(S \cap \Gamma(j))$ and $Q_j = |S \cap \Gamma(j)|$, and equations (8.18) and (8.19), we get the required guarantee. □

Before going into the rest of the proof, we summarize the guarantees for the *Hair step* and *Backbone step* in Table 8.1 and give more intuition into these bounds by drawing out parallels to the bounds in the average case. When S is updated and we do not find a dense k-subgraph, the lower bound on the ratio of $LP(S)/|S|$ decreases by a multiplicative factor of $d/(\rho D)$ for every backbone step, and increases by a factor of d/ρ for a hair step (when the correct conditions are met). The structure $W(r, s)$ is so designed that after all s steps, if we never found a ρ-dense k-subgraph, we get that $LP(S)/|S|$ becomes > 1, which is a contradiction! This argument forms the crux of the proof of Lemma 8.17 and hence Theorem 8.16. This is reminiscent of the average-case where the design of the structure $W(r, s)$ ensured that the inductively updated upper bounds for $|S(t)|$ brought about a contradiction in the final step. To note the similarity, notice that the ratio of the lower bounds for $LP(S(t))$ and $LP(S(t))/|S(t)|$ in the worst-case matches the upper bound for $|S(t)|$ in the average-case (upto a factor ρ).

Step	New Candidates $S(t+1)$	$LP(S(t))$ $\geq$	$LP(S(t))/\|S(t)\|$ $\geq$	Average-case $\|S(t)\| \leq$
Backbone step (Lemma 8.11)	$\Gamma(S)$	d/ρ [a]	$d/(\rho D)$ [a]	D
Hair step (Lemma 8.14)	$S \cap \Gamma(j)$ for a chosen leaf j	$d/2k$	$d/2\rho$ [b]	$1/k$

[a] when $LP(S(t))/|S(t)| \leq \rho/d$.
[b] when $|S(t)| \geq k$.

Table 8.1: Guarantees for Backbone (Lemma 8.11) and Hair steps (Lemma 8.14).

Recall that in the random case, the sets $S(t)$ had cardinality tightly concentrated around $n^{\mathrm{fr}(tr/s)}$. Similarly here, if we assume that $k = n/D (= D^{(s-r)/r})$, and that d (the density of the subgraph implied by the LP) is at least $\rho k^{r/s}$ (for some $\rho \geq 1$), then we show (see Corollary 8.18) that if until step t the algorithm has not found an

$\Omega(\rho)$-dense subgraph, then the current candidate set satisfies

$$\frac{\mathrm{LP}(S(t))}{|S(t)|} > \rho^{-\mathrm{fr}(tr/s)\cdot s/r}\left(\frac{k}{n}\right)^{\mathrm{fr}(tr/s)} = \left(\frac{1}{D}\right)^{\mathrm{fr}(tr/s)},$$

which will yield a contradiction at the final step (for $t = s$).

One difficulty is that we avoid making the assumption that $kD = n$ (which is possible, but would incur a $O(\sqrt{\log n})$ loss in the approximation guarantee). Instead, we use the fact that the greedy algorithm finds a k-subgraph with average degree $\gamma \geq \max\{1, Dk/n\}$. Specifically, we show that at step t of the algorithm, either a subgraph with average degree $\Omega(\rho)$ has already been found, or the greedy algorithm gives the desired approximation (i.e. $\gamma \geq \rho$), or we have the desired lower bounds on $\mathrm{LP}(S(t))$ and $\mathrm{LP}(S(t))/|S(t)|$.

8.3.2 Performance guarantee of the algorithm

The analysis is quite straightforward. We follow the various steps, and each time apply Lemma 8.11 or Lemma 8.14, as appropriate. Our main result is the following:

Theorem 8.16. *Let $s > r > 0$ be relatively prime integers, let G be an undirected (bipartite) graph with maximum degree $\leq D = n^{r/s}$, let $\{y_i\} \in \mathrm{DkS\text{-}LP}_{r+1}(G, k, d)$, and define $\gamma = \max\{Dk/n, 1\}$. Then if d' is the average degree of the subgraph found by $\mathrm{DkS\text{-}Cat}_{r,s}(G, k, \{y_i\})$, we have*

$$\max\{d', \gamma\} = \Omega(d/D^{(s-r)/s}).$$

From Section 8.1.1, we know that the Greedy algorithm produces a k-subgraph of density at least γ. Note that when the log-density α of G is not rational, we can choose rational $\alpha \leq r/s \leq \alpha + \epsilon$ for any small $\epsilon > 0$. We then still appeal to Theorem 8.16 as before, though the greedy algorithm might only return a subgraph of

average degree $\gamma' > \gamma/n^\epsilon$. Thus, the loss in the approximation ratio is at most n^ϵ. A fairly straightforward calculation shows that this implies a $O(n^{1/4+\epsilon})$-approximation in $n^{O(1/\epsilon)}$ time for all $\epsilon > 0$ (including $\epsilon = 1/\log n$).

Notation. In what follows, we let $\rho = d/(2D^{(s-r)/s})$ denote the desired average degree of the output subgraph (up to a constant factor). We use $bb(t) = t - \lfloor tr/s \rfloor - 1$ to denote the number of backbone steps among the first t steps, and $\ell f(t) = \lfloor tr/s \rfloor + 1$ to denote the number of hair/leaf steps in the first t steps.

We now state the main technical lemma.

Lemma 8.17. *Suppose G be an undirected (bipartite) graph with maximum degree at most $D = n^{r/s}$, which contains a k-subgraph with minimum degree at least $d > 0$. Let $s > r > 0$ be relatively prime integers, let and let $\{y_i\}$ be a solution to* DkS-$\mathrm{LP}_{r+1}(G, k, d)$. *Let $\gamma = \max\{Dk/n, 1\}$. If $\rho > \gamma$ and none of the subgraphs found by* DkS-$\mathrm{Cat}_{r,s}(G, k\{y_i\})$ *upto step t has average degree $\Omega(\rho)$, then*

$$\mathrm{LP}(S(t)) \geq d \cdot \left(\frac{d}{\rho}\right)^{bb(t)} \cdot \left(\frac{d}{2k}\right)^{\ell f(t)-1} \tag{8.20}$$

$$\frac{\mathrm{LP}(S(t))}{|S(t)|} \geq \frac{d}{D\gamma} \cdot \left(\frac{d}{2\rho}\right)^{\ell f(t)-1} \cdot \left(\frac{d}{\rho D}\right)^{bb(t)} \tag{8.21}$$

The following simple corollary immediately implies Theorem 8.16 (by contradiction) when we take $t = s$.

Corollary 8.18. *In the notation of Lemma 8.17, either the density d'_t of the densest k-subgraph found by* DkS-$\mathrm{Cat}_{r,s}$ *satisfies* $\max\{d'_t, \gamma\} = \Omega(d/D^{(s-r)/s})$, *or we have*

$$\frac{\mathrm{LP}(S(t))}{|S(t)|} \geq \frac{2^{t-\lfloor tr/s \rfloor}}{D^{\mathrm{fr}(tr/s)}}.$$

Proof. By definition, $\rho = \frac{d}{2D^{(1-r/s)}} \geq 1$. Hence, $2\rho D = dD^{r/s}$. The corollary follows just from the guarantees of Lemma 8.17:

$$\begin{aligned}
\frac{LP(S(t))}{|S(t)|} &\geq \frac{d}{D} \cdot \left(\frac{d}{2\rho}\right)^{\ell f(t)-1} \cdot \left(\frac{d}{\rho D}\right)^{bb(t)} && \text{By Lemma 8.17} \\
&\geq \left(\frac{d}{2\rho}\right)^{\ell f(t)-1} \cdot \left(\frac{d}{\rho D}\right)^{bb(t)+1} && \rho \geq \gamma \\
&\geq \left(D^{1-r/s}\right)^{\ell f(t)-1} \cdot \left(2D^{-r/s}\right)^{bb(t)+1} \\
&\geq 2^{bb(t)+1} D^{-tr/s} \cdot D^{\ell f(t)-1} \\
&= \frac{2^{t-\lfloor tr/s \rfloor}}{D^{tr/s-\lfloor tr/s \rfloor}}
\end{aligned}$$

□

Let us now proceed to the proof of Lemma 8.17.

Proof of Lemma 8.17. We prove by induction that if the algorithm does not find any $\Omega(\rho)$ dense subgraph in steps 1 through t, the lower bounds (8.20) and (8.21) hold. Intuitively, the lemma follows if the bounds given by in the table 8.1 hold (if we plug in the correct bounds for the first step). Hence, we will just need to check that the conditions of Lemma 8.11 hold and handle the case when $|S(t)| < k$ in Lemma 8.14.

For $t = 1$, the bounds hold trivially: Fix a node j_1 s.t. $y_{j_1} > 0$. Then $\mathrm{LP}_{\{y_{ij_1}/y_{j_1} | i \in V\}}(S(1)) \geq d$ (by constraint (8.5)) and so $\mathrm{LP}(S(1))/|S(1)| \geq d/|\Gamma(j_1)| \geq d/D$ (in particular $\geq d/(\gamma D)$), which is exactly what we need.

Now assume the lemma holds for some $1 \leq t \leq s-1$. We will show it for $t+1$, considering separately backbone and hair steps.

First, suppose $t+1$ is a backbone step. Since $t+1$ is a backbone step $[tr/s, (t+1)r/s]$ does not contain an integer. Hence, $\mathrm{fr}(tr/s) < 1 - r/s$. By Lemma 8.11, if $LP(S(t))/|S(t)| \geq \rho/d$, since we did not find a $\Omega(\rho)$ dense subgraph, the claims (corresponding to (8.20) and (8.21)) follows immediately from the inductive hypothesis. Thus it suffices to show that indeed $LP(S(t))/|S(t)| \geq \rho/d$. This follows from

Corollary 8.18 upto step t (induction hypothesis), which gives

$$\frac{\mathrm{LP}(S(t))}{|S(t)|} \geq \frac{2^{bb(t)+1}}{D^{\mathrm{fr}(tr/s)}} > \frac{2^{bb(t)+1}}{D^{1-r/s}} = \frac{2^{bb(t)+2}\rho}{d}.$$

Now, on the other hand, suppose $t+1$ is a hair step. Then the interval $[tr/s, (t+1)r/s]$ does contain an integer. Assuming Procedure DkS-Local$(S(t), k)$ does not return a subgraph with average degree at least ρ, by Lemma 8.14, there is some choice of vertex j_{t+1} such that, denoting $y'_i = y_{ij_{t+1}}$ for all $i \in V$, the following inequalities hold simultaneously:

$$\begin{aligned} \mathrm{LP}_{\{y'_i\}}(S(t+1)) &= \mathrm{LP}_{\{y'_i\}}(S(t) \cap \Gamma(j_{t+1})) \\ &\geq \frac{d \cdot \mathrm{LP}_{\{y_i\}}(S(t))}{2k}, \qquad (8.22) \\ \mathrm{LP}_{\{y'_i\}}(S(t+1))/|S(t+1)| &\geq \frac{d \cdot \mathrm{LP}_{\{y_i\}}(S(t))}{2\rho \cdot \max\{k, |S(t)|\}}. \qquad (8.23) \end{aligned}$$

Note that lower bound we require for $\mathrm{LP}(S(t+1))$ follows easily from (8.22) and the inductive hypothesis. If $|S(t)| \geq k$, the bound on $\mathrm{LP}(S(t+1))/|S(t+1)|$ similarly follows from (8.23) and the inductive hypothesis. Thus, it remains to show the required bound when $|S(t)| < k$. Here is where we use the lower bounds for $\mathrm{LP}(S(t+1))$ that we have been maintaining inductively all along. In this case, we show that while $\mathrm{LP}(S(t+1))/|S(t+1)|$ may not grow multiplicatively over $\mathrm{LP}(S(t))/|S(t)|$,

the lower bound claim corresponding to (8.21) still holds. We have from (8.23) that

$$\begin{aligned}
\mathrm{LP}(S(t+1))/|S(t+1)| &\geq \frac{d\mathrm{LP}(S(t))}{2\rho k} \\
&\geq \frac{d}{2k\rho} \times d\left(\frac{d}{\rho}\right)^{bb(t)} \cdot \left(\frac{d}{2k}\right)^{\ell f(t)-1} && \text{inductive hypothesis} \\
&\geq \frac{d}{\rho} \cdot \left(\frac{d}{\rho}\right)^{bb(t+1)} \cdot \left(\frac{d}{2k}\right)^{\ell f(t+1)-1} && \text{since } (t+1) \text{ is a leaf step} \\
&\geq \frac{d}{D\gamma}\left(\frac{d}{2\rho}\right)^{\ell f(t+1)-1} \cdot \left(\frac{d}{\rho D}\right)^{bb(t+1)} \times \\
&\qquad \times \frac{\gamma D^{1+bb(t+1)}\rho^{\ell f(t+1)-2}}{k^{\ell f(t+1)-1}}
\end{aligned}$$

We just need to show that the last term in this expression is ≥ 1.

$$\begin{aligned}
\frac{\gamma D^{1+bb(t+1)}\rho^{\ell f(t+1)-2}}{k^{\ell f(t+1)-1}} &= \frac{\gamma D^{1+bb(t)}\rho^{\ell f(t)-1}}{k^{\ell f(t)}} && \text{since } t+1 \text{ is a leaf step} \\
&\geq \frac{D^{1+t}}{n^{\ell f(t)}} && \text{since } \rho > \gamma \geq Dk/n \\
&\geq n^{(1+t)r/s-(1+\lfloor tr/s \rfloor)} \\
&\geq 1 && \text{since } [tr/s, (t+1)r/s] \text{ contains an integer.}
\end{aligned}$$

This concludes the proof. □

A combinatorial algorithm. Note that the only time the algorithm uses the LP values is in choosing the leaves. Thus, even in the absence of an LP solution, the algorithm can be run by trying all possible sequences of leaves (the analysis will still work by replacing the LP solution with the optimum $0-1$ solution). While this would take time $O(n^{r+1})$ as opposed to linear time (for the LP-based rounding algorithm), this is comparable to the time needed to solve the LP. An interesting open question is if it is possible to avoid the dependence on r, the number of leaves.

8.4 Subsequent Work

We mention in brief, a couple of extensions towards improving upon the $O(n^{1/4})$ approximation algorithm presented above. In [20], we describe an algorithm which gives an approximation ratio strictly better than $n^{1/4}$ for arbitrary graphs in subexponential ($2^{n^{O(\epsilon)}}$) time.

The algorithm is an extension to the caterpillar based algorithm presented in Section 8.2 to the case when the log-density of the subgraph H is (slightly) less than the log-density of the host graph (a crucial case if one wants to go beyond $n^{1/4}$-approximations). This is done at the expense of running time – we obtain a modification of our caterpillar-based algorithm, which yields an approximation ratio of $O(n^{(1-\epsilon)/4})$ in time $2^{n^{O(\epsilon)}}$. The main modification is that for each leaf, rather than picking an individual vertex, we will pick a cluster of roughly $O(n^{\epsilon})$ 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. with the union of all neighborhoods of vertices in the cluster).

Chapter 9

Integrality Gaps: How Hard is Average-Case Densest k-subgraph?

The clinching evidence for the hardness of a problem is through reductions from other well-known problems, which are believed to be hard (say, 3-SAT). For instance, from seminal works starting from the PCP theorem [14, 12, 55], we know that it is impossible to approximate MAX CLIQUE within $n^{1-o(1)}$ factor unless $P = NP$. When such satisfying evidence is not forthcoming, the next best alternative are lower bounds in restricted models of computation.

While only constant factor approximations for DkS have been ruled out, it is commonly believed that DkS is much harder to approximate (even upto polynomial factors in n). This believed inapproximability of DkS seems to come up in various other problems that look for solutions supported on *small* sets. Problems in network design (minimum λ-connected subgraph [65]), network reliability (finding small k-route cuts [32]), clustering (capacitated metric labeling [6]) have very large inapproximability assuming $n^{\Omega(1)}$ factor inapproximability for DkS. In fact, this hardness is believed to be true even on average (for a natural distribution on hard instances). Recently, average-case hardness assumptions based on the hardness of "planted" ver-

sions of DkS were used for public key cryptography [7] and in showing that financial derivates can be fraudulently priced without detection [8]. Given the importance of DkS as an average-case hardness assumption, it is much more satisfactory if we had some evidence for the average-case hardness of DkS.

In the context of approximation algorithms, linear programming (LP) and semi-definite programming (SDP) techniques are one of the most powerful tools at our disposal, and capture most of the state-of-the-art approximation algorithms known today [84]. Integrality gaps are lower bounds against these mathematical programming techniques, showing that such powerful methods can not be used to obtain good approximations. Lift-and-project methods (hierarchies) are systematic iterative procedures to obtain sequences of increasingly stronger linear and semi-definite programming relaxations for an integer optimization problem (e.g. Lovász-Schrijver, Sherali-Adams and Lasserre hierarchies. Please see the recent survey by Chlamtac and Tulsiani for details [31])). Typically, the relaxation obtained after r levels of these strengthenings can be solved in $n^{O(r)}$ time. In most cases of approximation algorithms that use strengthened LP and SDP relaxations, such relaxations can be obtained from a few levels of such lift-and-project procedures. Starting from the work of [9], a number of recent papers have studied the strength and limitations of such relaxations as a basis for designing approximation algorithms for various problems [9, 28, 29, 30, 45, 58, 75, 80, 79, 83]. In fact, the algorithms from chapter 8 give a $O(n^{1/4+\epsilon})$ approximation for DkS uses a linear programming relaxation which is weaker than that obtained from $O(1/\epsilon)$ levels of the Sherali-Adams hierarchy. [20] also show that the integrality gap becomes $O(n^{1/4-\epsilon})$ after $n^{O(\epsilon)}$ levels of the Sherali Adams LP hierarchy.

In this chapter, we show that polynomial time algorithms based on Sherali-Adams hierarchy based SDP relaxations can not give good approximations even in

the average-case. In particular, we show that even $\Omega(\frac{\log n}{\log\log n})$ levels of this hierachy can not distinguish between the following two distributions (roughly):

$\mathcal{D}_1$: graph G picked from $G(n, n^{-1/2})$, and

$\mathcal{D}_2$: graph G picked from $G(n, n^{-1/2})$ with the induced subgraph on $\sqrt{n}$ vertices replaced with $G(\sqrt{n}, n^{-(1/4+\epsilon)})$.

In what follows L will denote the number of levels of the hierarchy SA_L given in Fig. 9.2.

Theorem 9.1. *Let $L \leq \frac{\log n}{10\log\log n}$. In the notation given above, the integrality gap of SA_L is at least $\Omega\big(\frac{n^{1/4}}{L\log^2 n}\big)$.*

To prove Theorem 9.1, we present instances G where the relaxation has a solution with value $d = \Omega(n^{1/4}/L)$, while the *integer optimum*, i.e., the largest density of a k-subgraph in G is only $O(\log^2 n)$. It will be notationally convenient to construct gaps for $L/2$ levels.

Before we proceed further, we will first remind ourselves about the Sherali-Adams relaxation for Densest k-subgraph and then prove some simple random graph properties which will be useful for the proofs that follow.

9.1 The Sherali-Adams Relaxation for DkS

Let us first consider the basic linear program for DkS. The natural LP relaxation for DkS (LP1 in Fig. 9.1) was used in [82, 42]. It has variables $\{x_i\}$ to denote if vertex i belongs to the solution, and edge variables $\{x_{ij}\}_{(i,j)\in E(G)}$ to denote if both i, j are in the subgraph. This LP has an integrality gap of $\Omega(\frac{n}{k})$ ([40, 42]). We can also write another linear program (LP2 in Fig. 9.1), which is slightly stronger. Intuitively, LP2 tries to find a k-subgraph H where the *minimum degree* d_H is maximized (see Lemma 8.8 for why this is feasible). While the program as stated is not linear, we guess the degree d and consider the feasibility linear program that is obtained.

Natural LP (LP1):

$$\begin{aligned} \max \quad & \sum_{(i,j)\in E(G)} x_{ij} \\ \text{s.t.} \quad & \sum_{i\in V} x_i \le k \\ \forall i,j \in V, \quad & 0 \le x_{ij} \le x_i \le 1 \end{aligned}$$

Min. degree LP (LP2):

$$\begin{aligned} \max \quad & d \qquad \text{(guess } d \text{)} \\ \text{s.t.} \quad & \sum_{i\in V} x_i \le k, \qquad \text{and} \\ \exists \{x_{ij} \mid i,j \in V\} \ & \text{s.t.} \\ \forall i \in V \quad & \sum_{j\in\Gamma(i)} x_{ij} \ge d x_i \\ \forall i,j \in V \quad & x_{ij} = x_{ji} \\ \forall i,j \in V \quad & 0 \le x_{ij} \le x_i \le 1 \end{aligned}$$

Figure 9.1: Two Linear Programming relaxations for DkS

The Sherali-Adams hierarchy starts with a simple LP relaxation of a $\{0,1\}$ integer program, and obtains a sequence of successively tighter relaxations with more levels. For our integrality gaps, we will in fact start with the stronger basic (first-level) linear program (LP2 in Fig.9.1) that was used in the previous chapter (Figure 8.1 on page 129). From Lemma 8.6, it can be easily seen that the two LPs are almost equivalent upto a factor of 2.

We consider strengthening this LP by considering r levels of the Sherali-Adams hierarchy (SA_r, shown in Fig 9.2). In the *lifted* LP, the variable x_S is supposed to capture whether every vertex in S belongs to the chosen k-subgraph (i.e., $x_S = \prod_{i\in S} x_i$). Further if we take two sets S, S' of $\le r$ vertices, the local distributions induced by a feasible solution (using the inclusion-exclusion constraints), agree on the variables in the intersection $S \cap S'$. We follow the notation established in [31] while defining the hierarchy. We present the entire relaxation here again for the benefit of exposition.

$$\max \quad d, \text{ s.t.}$$
$$\exists\{x_S \mid S \subseteq V, |S| \leq r\} \text{ s.t. } x_\emptyset = 1 \text{ and}$$
$$\forall S, T \subseteq V \text{ s.t } |S| + |T| \leq r :$$
$$\sum_{i \in V} \sum_{J \subseteq T} (-1)^{|J|} x_{S \cup J \cup \{i\}} \leq k \sum_{J \subseteq T} (-1)^{|J|} x_{S \cup J} \tag{9.1}$$
$$\forall i \in V \quad \sum_{j \in \Gamma(i)} \sum_{J \subseteq T} (-1)^{|J|} x_{S \cup J \cup \{i,j\}} \geq d \sum_{J \subseteq T} (-1)^{|J|} x_{S \cup J \cup \{i\}} \tag{9.2}$$
$$0 \leq \sum_{J \subseteq T} (-1)^{|J|} x_{S \cup J} \leq 1 \tag{9.3}$$

Figure 9.2: Sherali-Adams LP relaxation (r levels) for DkS: SA_r

9.1.1 Sherali-Adams SDP hierarchy

The Sherali-Adams SDP hierarchy (also refered to as the mixed hierarchy or $SA+$) imposes an additional SDP constraint on top of the Sherali-Adams LP relaxation. In particular, it asks for the values x_{ij} to come from vector inner products i.e. the matrix $X = (x_{ij})$ is p.s.d. Most known algorithms which proceed by rounding a relaxation obtained from an SDP hierarchy [29, 30, 19] work with this mixed hierarchy. [75, 64] and [46] studied this hierarchy for constraint satisfaction problems and obtained integrality gaps for Unique Games and approximation-resistant CSPs respectively.

One level of the mixed h ierarchy for DkS gives the SDP relaxation introduced in [42, 82]. In fact, we showed in Chapter 7 how the mixed hierarchy performs better than log-density based arguments (which are captured by just the LP hierarchy) in the parameter range $D < n^{1/2}$. It is interesting in this light to obtain integrality gaps for mixed hierarchy.

9.2 Random graph properties

We prove that the properties used in our gap construction hold for $\mathcal{G}(n,p)$, with $p = n^{-1/2}(\log n)^{1/2}$. These properties are listed in Section 9.3.1. In what follows fix

p to be the value above. As in previous chapters, the phrase "with high probability" (w.h.p.) refers to 'with probability at least $1 - \frac{1}{q(n)}$', where $q(n)$ is an arbitrary polynomial in n (sometimes there will be a constant depending on the polynomial).

Lemma 9.2. *Every vertex of G (as defined above) has degree between $(n^{1/2}\log n)/2$ and $2n^{1/2}\log n$ w.h.p.*

Proof. Let $u \in V$. The degree $d(u)$ (as a random variable) is the sum of n i.i.d. Bernoulli random variables each having parameter $p = n^{-1/2}\log n$. The expected value is thus $n^{1/2}\log n$. This is $\gg \log n$, and thus by Chernoff bounds, the probability that $\Pr[|d(u) - n^{1/2}\log n| > t] \leq e^{-t^2/4np(1-p)} < \frac{1}{nq(n)}$, for any polynomial $q(n)$. Taking union bound gives the claim. □

Lemma 9.3. *Every pair of vertices in G (as defined above) have at most $2\log^2 n$ common neighbours w.h.p.*

Proof. Let $u, v \in V$. Let X_i be a random variable which is an indicator for $i \in \Gamma(u) \cap \Gamma(v)$. In $\mathcal{G}(n,p)$, we have $\mathbb{E}[X_i] = p^2 = \log^2 n/n$. Thus $\mathbb{E}[|\Gamma(u) \cap \Gamma(v)|] = np^2 = \log^2 n$. Thus the probability that it is $> 2\log^2 n$ is at most $e^{-\log^2 n/4}$. Taking union bound over all u, v, we obtain that this is smaller than any polynomial. □

Lemma 9.4. *In the notation of this section, every pair of vertices have at least one common neighbour w.h.p.*

Proof. As above, consider some u, v; we have $\mathbb{E}[|\Gamma(u) \cap \Gamma(v)|] = np^2 = \log^2 n$. Thus $\Pr[|\Gamma_u \cap \Gamma(v)| < \log n] \leq e^{-\log^2 n/4}$ (since we can use Chernoff bounds as long as the expectation $\gg \log n$). Taking union bound again implies the result. □

Lemma 9.5. *In the notation of this section, no induced subgraph on $n^{1/2}$ vertices has density $> 5\log n$ w.h.p.*

Proof. Let $S \subseteq V$ of size $n^{1/2}$. Then $\mathbb{E}[E(S,S)] = \binom{n^{1/2}}{2} \cdot p = n^{1/2}\log n/2$. Further the variance of this quantity is $\binom{n^{1/2}}{2}p(1-p) < n^{1/2}\log n$. Thus by Chernoff bound,

$$\Pr[|E(S,S) - \mathbb{E}[E(S,S)]| > t] \leq e^{-t^2/4n^{1/2}\log n}.$$

Picking $t = 4n^{1/2}\log n$, the probability upper bound is $e^{-4n^{1/2}\log n}$. Thus we can take a union bound over all the $\binom{n}{n^{1/2}}$ subsets S. This proves the claim. □

9.3 Integrality Gaps for Sherali-Adams

9.3.1 The instance

The distribution over instances will be given by $\mathcal{D}_1$, and prove that the desired gap holds with high probability. The instances we consider are $\mathcal{G}(n, p = n^{-1/2}\log n)$ (thus the expected degree of each vertex is $D = n^{1/2}\log n$). The parameter k is chosen to be $n^{1/2}$. An easy calculation shows that in any k subgraph, the density (and hence the min-degree) is at most $O(\log^2 n)$ (see Section 9.2). The meat of the argument is thus to show that there exists an LP solution to $\mathrm{SA}_{L/2}$ (Equations (9.1)-(9.3)) of value $d = \Omega(n^{1/4}/L)$ even for L of the order $\log n/\log\log n$.

The following are the properties of the distribution $\mathcal{G}(n,p)$ (with above parameters) we will truly be using [see Section 9.2 for proofs]. Any graph with these properties admits the solution to $\mathrm{SA}_{L/2}$ which we describe.

1. Every vertex has degree between $(n^{1/2}\log n)/2$ and $2n^{1/2}\log n$.

2. Any two vertices i, j have at least one common neighbor and has at most $O(\log n)$ common neighbours.

9.3.2 Feasible solution

Before formally giving the x_S values, we give intuition as to what they *ought to* be. First, we start out setting $x_i = n^{-1/2}$ (equal for all vertices, since $\sum_i x_i \leq k = n^{1/2}$ and no vertex is special). Next, suppose $S \subset V$ with $i \in S$ and think of $d \approx n^{1/4}$. Now (9.2) implies that $\sum_{j \in \Gamma(i)} x_{S \cup j} \geq n^{1/4} x_S$. Further from (9.1), we obtain $\sum_{j \in V} x_{S \cup j} \leq n^{1/2} x_S$. Thus we conclude that $x_{S \cup j}$ must be roughly $n^{-1/4} x_S$ for $j \in \Gamma(S)$, while for $j \notin \Gamma(S)$, it should be only $n^{-1/2} x_S$. Now consider $T \subset V$ which span a tree: we could imagine starting with one vertex and adding vertices one by one (each added vertex is a neighbour of the previous ones), and thus conclude that x_T is roughly $n^{-(|T|+1)/4}$ (since $x_i = n^{-1/2}$ to begin with). Now let S be an arbitrary set of vertices and consider a tree $T \supseteq S$: by monotonicity (a corollary of (9.3)), $x_S \geq x_T$, and since this is true for *every* such T, we need to set x_S to be at least $n^{-(\mathsf{st}(S)+1)/4}$, where $\mathsf{st}(S)$ is the number of vertices (size) in the minimum Steiner tree of S.

These, with additional 'dampening' factors (L-terms), are precisely the values we will set. More precisely we consider the solution

$$x_S = n^{-\frac{1}{4} \cdot (\mathsf{st}(S)+1)} \cdot L^{-|S|}, \tag{9.4}$$

where $\mathsf{st}(S)$, as above, is the size of the minimum Steiner tree of S. Thus for instance $x_i = n^{-1/2}/L$, while $x_{\{i,j\}} = 1/(n^{3/4}L^2)$ when $(i,j) \in E$ and $1/(nL^2)$ otherwise (the latter is because there is a path of length-2 between any $i, j \in G$ with high probability).

Let us fix $L \leq \log n/(10 \log\log n)$. We now show that the LP solution presented above is feasible for $\mathrm{SA}_{L/2}$ with high probability. The following lemma is useful in simplifying the analysis: it implies that we need to only consider $T = \emptyset$ while showing that the LP solution satisfies constraints (9.1) and (9.2). This is where the 'dampening' factors come into play.

Lemma 9.6. *Let S, T be disjoint subsets of V of size at most t and x_S be the solution described above. Then*

$$x_S \geq \sum_{J \subseteq T} (-1)^{|J|} x_{S \cup J} \geq \frac{x_S}{2}$$

Proof. One property of the assignment (9.4) is that $x_{S \cup i} \leq x_S / L$ for $i \notin S$. Further all the x_S are ≥ 0, and thus in the sum above, the term corresponding to $J \subseteq T$ contibutes positively when $|J|$ is even and negatively otherwise. Hence,

$$\begin{aligned}
\sum_{J \subseteq T} (-1)^{|J|} x_{S \cup J} &\geq \sum_{\ell=0}^{\ell=\lfloor \frac{|T|}{2} \rfloor} \sum_{\substack{J \subseteq T \\ |J|=2\ell}} \left(x_{S \cup J} - \sum_{i \in T \setminus S} x_{S \cup J \cup \{i\}} \right) \\
&\geq \sum_{\ell=0}^{\lfloor \frac{|T|}{2} \rfloor} \sum_{\substack{J \subseteq T \\ |J|=2\ell}} x_{S \cup J} (1 - |T|/L) \\
&\geq \sum_{\ell=0}^{\lfloor \frac{|T|}{2} \rfloor} \frac{1}{2} \cdot \sum_{\substack{J \subseteq T \\ |J|=2\ell}} x_{S \cup J} \geq \frac{x_S}{2} \qquad \text{(since } |T| \leq L/2 \text{)}
\end{aligned}$$

A similar proof shows the upper bound, since the $x_{S \cup \{i\}}$ terms for $i \in T$ dominate the contributions of $x_{S \cup J}$ for $|J| > 1$. □

Corollary 9.7. *In checking feasibility, it suffices to check* (9.1) *and* (9.2) *with $T = \emptyset$.*

Proof. Lemma 9.6 allows us to 'remove' the $\sum_{J \subseteq T}$ on both sides of the equations (and set $T = \emptyset$) by losing a factor of 2. Since we allow constant slack, the claim follows. □

We refer to the constraints (9.1) and (9.2) as the *size* and the *density* constraints respectively, because the former says that we should pick only a k-subgraph, and the latter says the minimum degree (density) is at least d. The assignment we described allows us to prove the density constraint easily.

Lemma 9.8. *(Density Constraint) The x_S described above satisfy constraints* (9.2).

Proof. Let $S \subset V$ and $i \in S$. We need to check that $\sum_{j \in \Gamma(i)} x_{S \cup j} \geq \frac{n^{1/4}}{L} \cdot x_S$. It is easy to see that for every $j \in \Gamma(i)$, $\mathsf{st}(S \cup j) \leq \mathsf{st}(S) + 1$, and thus $x_{S \cup j} \geq \frac{n^{-1/4}}{L} \cdot x_S$ (the L term is due to the dependence on $|S|$ in (9.4)). Since there are at least $n^{1/2} \log n/2$ terms in the LHS, the inequality follows. □

9.3.3 The Size Constraint and Minimum Steiner trees in $\mathcal{G}(n,p)$

By the above corollary, it suffices to check (noting $k = n^{1/2}$) that

$$\sum_{i \in V} x_{S \cup i} \leq n^{1/2} x_S \qquad \text{for all } S \subset V, |S| < t. \tag{9.5}$$

We show this by proving that $\mathsf{st}(S \cup i) \geq \mathsf{st}(S) + 2$ for *most* $i \in V$, in particular we bound the number of exceptions (lemmas below state the precise bounds). This then implies that (9.5) holds.

We start with some basic facts (and notation) about Minimum Steiner trees (minST) of $S(\subset V)$ in $\mathcal{G}(n,p)$, with our parameters. We will refer to the vertices in S as the *terminals*, and the rest of the vertices in a minST as the *non-terminals*. First, the minST must have all its leaves to be terminals. Further, since every two vertices in G have a path of length two, we must have $\mathsf{st}(S) \leq 2|S| - 1$ for all S. This helps us bound the number of *tree structures* the minST of S can have. We define this formally.

Given $S \subset V$, a *tree structure* for S is a tree T along with a mapping $g : V(S) \to V(T)$ which is one-one (not necessarily onto). The vertices in T without an inverse image in S are called *internal vertices* and the rest are also called *fixed vertices*. A tree structure for S is *valid* if it is possible to 'fill in' the internal vertices with distinct vertices from V such that all the edges in the tree are also present in G. [The relation to Steiner trees is apparent – the internal vertices are the Steiner vertices]. Given an

internal vertex in T, the vertices of G which take that *position* in some valid 'filling in' are called the set of *candidates* for that position.

Before we get to the lemmas, we note that the number of tree structures for S of size $\leq 2|S|$ is at most $(2|S|)^{2|S|}$ (this is just by a naïve bound using the number of trees). Let us now bound the number of $i \in V$ for which $\mathsf{st}(S \cup i) \leq \mathsf{st}(S) + 1$.

Lemma 9.9. *Let $G \sim \mathcal{G}(n,p)$ (p fixed above), $S \subset V(G)$ and T be a tree structure for a min Steiner tree of S (so the leaves of T are elements of S). Then the number of candidates for each of the* positions *in T is at most* $(\log n)^{|S|}$.

Proof. The proof is by induction on the size of S. The base case $|S| = 1$ is trivial. Assume the result for all tree structures of sets of size $\leq |S| - 1$. Now consider S. We may assume that T has at least one non-terminal, as otherwise there is nothing to prove.

First, note that there exists a vertex $u \in T$ which is adjacent to at most one non-leaf vertex in T. This is because deleting all the leaves in T gives a tree (which is not empty as there is at least one non-terminal in T), and a leaf in this tree our required u. If u is a terminal, we could remove the leaves attached to u (thus obtaining a subset S' of the terminals), and the remaining tree structure would be a valid min Steiner tree for S'. Further, the set of non-terminals is precisely the same, and thus the inductive hypothesis implies the claim for S. Thus suppose u is a non-terminal.

If the degree of u (in T) is 2, then u has precisely one leaf attached to it (call it ℓ). Consider the tree T' obtained by removing u, ℓ, and let $S' = S \setminus \ell$. Now T' is a min Steiner tree for S' (if not, we could consider use this smaller tree for S' along with a path of length 2 to ℓ to obtain a smaller minimum steiner tree for S). If b is the vertex in T' attached to u, there are at most $(\log n)^{|S|-1}$ candidates for b, by Induction Hypothesis. For each candidate b, the number of candidate u is only $(\log n)$, and since ℓ is a terminal. Thus the number of candidates for u is at most

$(\log n)^{|S|}$. The rest of the non-terminals in T are also present in T', and this gives the result.

If degree$(u) > 2$, then there are at least two leaves attached to u, thus the number of candidates for u is only $\log n$. Consider one candidate x for u. Let T' be the tree obtained by removing all the leaves attached to u (thus u is now a leaf), and S' be $S \cup x$ minus the set of leaves attached to u. Now T' is a min Steiner tree structure for S' (otherwise we can obtain a smaller tree for S). Thus by the inductive hypothesis, the number of candidates for any internal vertex in T' is at most $(\log n)^{|S|-1}$. Since there are only $\log n$ of the x's, it follows that the total #(candidates) for an internal vertex is at most $(\log n)^{|S|}$.

This completes the proof, by induction. □

An easy corollary is the following.

Corollary 9.10. *Let $S \subset V$, as in Lemma 9.9. There are at most $(2|S| \log n)^{2|S|}$ vertices i such that $\mathsf{st}(S \cup i) = \mathsf{st}(S)$.*

Proof. Each such i must be the internal vertex of some min Steiner tree for S, and there are at most $(2|S|)^{2|S|}$ tree structures. Lemma 9.9 now implies the claim. □

Lemma 9.11. *Let $S \subset V(G)$ where $G \sim \mathcal{G}(n,p)$ (p fixed above). There are at most $(4|S| \log n)^{4|S|} \times n^{1/2}$ vertices i such that $\mathsf{st}(S \cup i) = \mathsf{st}(S) + 1$.*

Proof. Let i be such a vertex. First, note that if there exists a min Steiner tree for $S \cup i$ with i as a leaf, we are done. This is because removing i gives a min Steiner tree for S, and thus i is a neighbour of an internal vertex in a min Steiner tree for S. Thus by 9.10 there are only $(2|S| \log n)^{2|S|} \times (2n^{1/2} \log n)$ such i.

Thus suppose that the min Steiner tree for $S \cup i$ has i as an internal vertex. We will prove the bound as follows: we consider a tree structure T of size $\mathsf{st}(S) + 1$ with leaves being terminals from S; then we show that the number of candidates for any

fixed position in T is at most $(2|S|\log n)^{2|S|}n^{1/2}$. This suffices, because the number of choices of tree structures adds an additional factor of $(2|S|)^{2|S|}$.

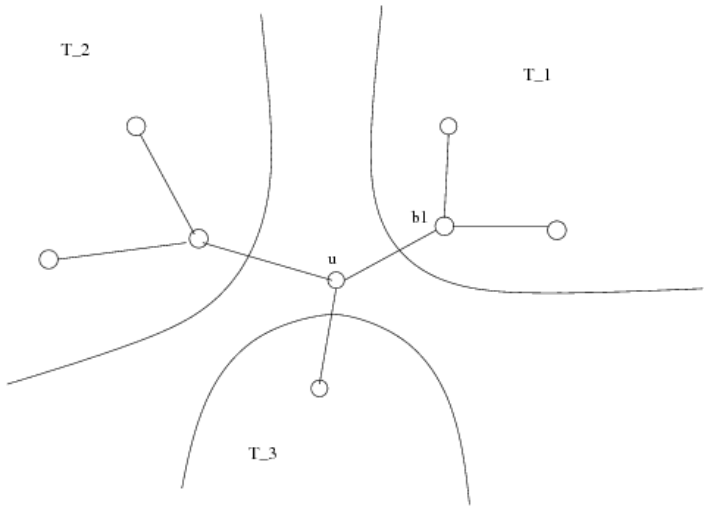

Figure 9.3: A min Steiner Tree for $S \cup u$ having u as internal vertex

Let us consider a structure T as above (in Figure 9.3), and a position u. Since u is not a leaf, it has degree at least 2. Let the degree be d, and let $T_1, \ldots, T_d$ be the subtrees of T formed by removing u (see figure ...). Now if for some i, T_i is the min Steiner tree for the terminals in T_i, we are done, because then, each candidate for u must be neighbour of an internal vertex in the tree, and by 9.10 there are only $\sqrt{n} \times (2|S|\log n)^{2|S|}$ candidates. Thus for *each* i, T_i must have a strictly smaller tree T_i'. Let the vertex in T_1 connected to u be called b_1. Now construct a new tree as follows: leave T_1 intact, and replace $T_2, \ldots, T_d$ by $T_2', \ldots, T_d'$; connect b_1 to $T_2', \ldots, T_d'$ using paths of length 2. The number of edges in the new tree is now at most $|T| - d - (d-1) + 2(d-1)$. The first term is the original cost, followed by removal of u, followed by the decrease by using T_i' as opposed to T_i, followed by the cost of adding length-2 paths.

Thus the new tree has cost at most $|T| - 1$, and thus it is optimal for S! Further, u is adjacent to b_1 which is an internal vertex, and thus the number of candidates is bounded by the desired quantity. □

Putting things together. Consider the sum $\sum_{i \in V} x_{S \cup i}$. 9.10 implies that there are at most $(L \log n)^L$ terms which contribute a value x_S/L. Lemma 9.11 implies that there are at most $n^{1/2} \cdot (2L \log n)^{2L}$ terms which contribute a value $x_S/(n^{1/4}L)$. Thus if we pick $(2L \log n)^{2L} < n^{1/4}$, we have the bound that the sum is at most $n^{1/2} x_S$, as desired.

Thus we have verified each of the constraints (9.1)-(9.3). This completes the proof of Theorem 9.1.

9.3.4 Gaps for the mixed hierarchy (SA+)

Consider the relaxation SA_t described in (9.1)-(9.2), along with the constraint: $Z = (x_{ij})_{1 \le i,j \le n} \succeq 0$. The solution considered earlier (Equation (9.4)) turns out to also satisfy this PSD condition with high probability. The entries of Z are

$$Z_{ij} = \begin{cases} n^{-1/2}/L \text{ if } i = j \\ n^{-\frac{3}{4}}/L^2 \text{ if } (i,j) \in E(G) \\ n^{-1}/L^2 \text{ otherwise} \end{cases}$$

Thus we have

$$Z = \frac{1}{L} \cdot \Big[\frac{1}{nL} J + \frac{1}{n^{1/2}} I + \frac{1}{n^{3/4}L} A \Big],$$

where A is the adjacency matrix of G. Now A is a $\mathcal{G}(n,p)$ matrix with $p = n^{-1/2} \log n$. Thus the least eigenvalue is at least $-2\sqrt{np(1-p)}$ with high probability (by the Semicircle law). This is at least $-4n^{1/4}(\log n)^{1/2}$.

Thus we have $A + 4n^{1/4}\sqrt{\log n} I \succeq 0$. Using the fact that $J \succeq 0$, we obtain that $Z \succeq 0$.

This shows that adding an SDP constraint at the *first level* does not give us any additional power – the relaxation obtained after $\Omega(\frac{\log n}{\log \log n})$ levels also has an integrality gap of $\tilde{O}(n^{1/4})$.

9.4 Conclusions and Subsequent work

These results are based on joint work with Bhaskara, Charikar, Guruswami and Zhou [21]. In the same work, we also consider the much stronger Lasserre hierarchy of relaxations for DkS. We show an integrality gap of polynomial ratio (n^ϵ, for small enough constant ϵ) for almost linear ($n^{1-O(\epsilon)}$) levels of the Lasserre relaxation. If we only aim at an integrality gap for polynomial (n^ϵ) levels of the Lasserre relaxation, the ratio of the gap can be as large as $n^{2/53-O(\epsilon)}$. However, these gap instances do not correspond to the natural average-case model for DkS. The distribution instead corresponds to the image of gadget reductions on random instances of a constraint satisfaction problem with large arity, over a large alphabet.

While prior results exhibiting gap instances for lift-and-project relaxations do so for problems that are *already known* to be hard to approximate under some suitable assumption; based on this hardness result, one would *expect* lift-and-project relaxations to have an integrality gap that matches the inapproximability factor. Our gap constructions for Densest k-Subgraph are a rare exception to this trend, as the integrality gaps we show are substantially stronger than the (very weak) hardness bounds known for the problem.

In the absence of inapproximability results for Densest k-Subgraph, our results show that beating a factor of $n^{\Omega(1)}$ is a barrier for even the most powerful SDPs, and in fact even beating the best known $n^{1/4}$ factor is a barrier for current techniques. These results are perhaps indicative of the hardness of approximating DkS within $n^{\Omega(1)}$ factors.

Our work leaves open two intriguing questions

Question 7. *1. Does Theorem 9.1 hold for $L \gg \log n$?*

2. Does the $n^{1/4}$ gap also hold for the Lasserre hierarchy?

We conjecture that even $L \approx n^{\epsilon}$ levels does not reduce the integrality gap substantially. We need a different approach (involving a better argument for bounding the number of trees) to extend the arguments above to this range of L.

Relation to the Small Set Expansion and Unique Games. A problem related to DkS is the Small Set Expansion (SSE) problem, which has received a lot of recent attention due to strong connections to the Unique Games conjecture [76]. One way to state the SSE conjecture [76] (which would imply the Unique Games conjecture) is as follows: for all $\epsilon > 0$, there exists δ, D (think of D as a constant), such that the following problem is hard:

Definition 9.12 (The Gap-SSE problem)**.** *Given a D-regular instance $G(V, E)$ with $k = \delta n$, the Gap-SSE problem is to distinguish between the following two cases.*

- ***Yes case.*** *There exists a subgraph of k vertices with average degree at least $(1 - \epsilon)D$.*
- ***No case.*** *All subgraphs of k vertices have average degree at most ϵD.*

Clearly, DkS is hard to approximate within any constant factor, assuming the Small Set Expansion conjecture. On the other hand, our results indicate that approximating DkS even within a polynomial factor may be a harder problem than Unique Games or Small Set Expansion, because these problems were recently shown to be solvable using $n^{\Omega(1)}$ rounds of the Lasserre hierarchy [19, 48].

Chapter 10

Open Problems

In this thesis we tried to better understand the approximability of two fundamental graph optimization problems : Graph Partitioning and Densest k-subgraph, by studying different average-case models and structured assumptions about the instances. We showed that we can design much better approximation algorithms for problems like Balanced Cut for realistic average-case models. The algorithmic framework that we introduced was fairly general and applies to several graph partitioning problems. Our average-case study of Densest k-subgraph revealed crucial insights into the approximability of the problem: the algorithms we designed for the average-case directly inspired worst-case approximation algorithms with the same guarantees. Besides, the natural average-case model exactly captured the extent of current techniques for Densest k-subgraph.

This average-case study of the two problems undertaken in this dissertation motivates such a study for other important classes of graph optimization problems, which are not well understood from an approximability standpoint. Our results also lead to many other interesting directions for future research, some of which we outline below.

10.1 Algorithms for good Average-case models

Improved approximations for Realistic average-case models

The broad research direction of looking beyong the rigorous demands of worst-case guarantees, in our search for good approximation algorithms seems like a direction with much promise for progress. The challenge is to identify average-case models which are realistic and natural, and yet, design improved algorithms for them.

Question 8. *Can we design good models and algorithms for "real-world" instances of important optimization problems?*

One example is the case of ordering problems, the simplest of which is the Feedback Arc Set problem [84], for which we know only $O(\log n \log\log n)$ approximation in the worst-case. An appropriate average-case could be one where we view the backward edges in the optimal ordering as random noise. Another problem which would be interesting to study is minimum (balanced) Vertex Separator problem. A model where the vertices involved in the separator are chosen randomly may be a natural setting, and also pose a challenge to our techniques because of the limited randomness in such a model.

Semi-random Graph Partitioning

In Chapter 4 and Chapter 5 we gave $O(1)$ approximations for semi-random instances for various graph partitioning problems. We believe that our model can capture fairly general instances that come up in practice — in this light, it would be very propitious to have practical algorithms that achieve small constant factor approximation, or better still a Polynomial Time Approximation Scheme.

Question 9. *Can we design practical near-linear time algorithms achieve $O(1)$ approximations for semi-random instances of graph partitioning?*

Another obvious direction in the context of graph partitioning is to develop good approximations for more general or other realistic models for graph partitioning. For instance, it would be very interesting to design a $O(1)$ approximation under weak stability assumptions in the spirit of [17, 16]. The most ambitious question in this mould would be to design such approximations for an appropriate smoothed version of graph partitioning problems (please refer to [81] for the challenges in designing smoothed models for discrete optimization problems).

10.2 Translating Average Case Algorithms to Worst Case

In Chapter 8, we saw how we could systematically translate the counting algorithms for a natural average-case model for Densest k-subgraph to the worst case using linear programming hierarchies. It would be very interesting if this can be generalized to find use in other problems.

A concrete problem where such an effort can be fruitful is the classic problem of approximate 3-coloring. The current best algorithm [29] combines combinatorial techniques of Blum[23] with the SDP-based approaches using three levels of the Lasserre hierarchy (see also [10] for more details). Blum's algorithm tries to find a large independent set in the distance-2 neighborhood of some vertex : this approach is motivated by a similar algorithm for the random model.

Question 10. *Can an algorithm that uses* k-rounds of the Lasserre hierarchy *(inspired by the length-k path counting algorithms of [24] for the semi-random model) give an* $n^{O(1/k)}$-coloring of a 3-colorable graph*?*

Another problem with a Densest k-subgraph flavor, that could benefit by a study of a suitable average-case version is the Label Cover problem (and its variants): its approximation factor dictates the best inapproximability results for several problems.

These questions are a part of the following broader direction:

Question 11. *Can we characterize a class of algorithms (say, local-counting algorithms) and problems, for which we can systematically translate algorithms from average-case to the worst-case, by rounding a suitable lift-and-project relaxation?*

10.3 Integrality Gaps and Evidence for Barriers

Densest k-subgraph

The most interesting question left open by this thesis on the Densest k-subgraph problem is whether we can break the "log density barrier". This is captured by the following simple question about random graphs:

Question 12. *Can we distinguish between the following two distributions:*

- $\mathcal{D}_1$*: graph* G *picked from* $G(n, n^{-1/2})$.
- $\mathcal{D}_2$*: graph* G *picked from* $G(n, n^{-1/2})$ *with the induced subgraph on* $\sqrt{n}$ *vertices replaced with* $G(\sqrt{n}, n^{-(1/4+\epsilon)})$.

This setting and other average-case variants of Densest k-subgraph seems to present a barrier for algorithmic progress in many optimization problems with strict budgets. An advantage with using these barriers as average-case assumptions is that we have a candidate distribution of hard instances (unlike the Unique Games conjecture, for instance). Stronger evidence for these average-case assumptions (building on our work in [21]) would place them on a firmer footing.

An important research direction in this context, is to identify a unifying hardness assumption about finding small dense subgraphs:

Question 13. *Can we identify a broad family of problems, that become intractable based on an assumption about the hardness of Densest k-subgraph?*

Candidate Hard Distribution for Graph Partitioning

A complementary research question that comes up from our work on average-case algorithms for graph partitioning problems is identifying candidate hard distributions for basic problems like Balanced Cut or Small Set Expansion. Our algorithms in Chapter 4 and Chapter 6 show that sufficient randomness either within the clusters or between clusters to generate instances make them easy. However, there is a strong belief that these problems may not have constant factor approximations in the worst-case — in that case, it would be more satisfying to have a candidate distribution of hard instances, with some evidence viz. integrality gaps.

Question 14. *Is there a natural distribution of instances for Balanced Cut or Small Set Expansion which give* $\omega(1)$ *for strong SDP hierarchies?*

Bibliography

[1] Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. $O(\sqrt{\log n})$ approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, STOC '05, pages 573–581, New York, NY, USA, 2005. ACM.

[2] N. Alon and V. Milman. Isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, Series B*, 38(1):73–88, February 1985.

[3] Noga Alon, Michael Krivelevich, and Benny Sudakov. Finding a large hidden clique in a random graph. *Random Structures & Algorithms*, 13(3-4):457–466, 1998.

[4] Noga Alon and Joel Spencer. *The Probabilistic Method*. John Wiley, 1992.

[5] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[6] Matthew Andrews, Mohammad Taghi Hajiaghayi, Howard J. Karloff, and Ankur Moitra. Capacitated metric labeling. In Dana Randall, editor, *SODA*, pages 976–995. SIAM, 2011.

[7] Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In Leonard J. Schulman, editor, *STOC*, pages 171–180. ACM, 2010.

[8] Sanjeev Arora, Boaz Barak, Markus Brunnermeier, and Rong Ge. Computational complexity and information asymmetry in financial products. In *Proceedings of the First Symposium on Innovations in Computer Science (ICS)*, 2010.

[9] Sanjeev Arora, Béla Bollobás, László Lovász, and Iannis Tourlakis. Proving integrality gaps without knowing the linear program. *Theory of Computing*, 2(1):19–51, 2006.

[10] Sanjeev Arora and Rong Ge. New tools for graph coloring. In Leslie Ann Goldberg, Klaus Jansen, R. Ravi, and José D. P. Rolim, editors, *APPROX-RANDOM*, volume 6845 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2011.

[11] Sanjeev Arora, Subhash A. Khot, Alexandra Kolla, David Steurer, Madhur Tulsiani, and Nisheeth K. Vishnoi. Unique games on expanding constraint graphs are easy: extended abstract. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC '08, pages 21–28, New York, NY, USA, 2008. ACM.

[12] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[13] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, STOC '04, pages 222–231, New York, NY, USA, 2004. ACM.

[14] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of np. *J. ACM*, 45(1):70–122, 1998.

[15] Yuichi Asahiro, Refael Hassin, and Kazuo Iwama. Complexity of finding dense subgraphs. *Discrete Appl. Math.*, 121(1-3):15–26, 2002.

[16] Maria-Florina Balcan. Better guarantees for sparsest cut clustering. In *COLT*, 2009.

[17] Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. Approximate clustering without the approximation. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 1068–1077, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[18] Nikhil Bansal, Uriel Feige, Robert Krauthgamer, Konstantin Makarychev, Viswanath Nagarajan, Joseph Naor, and Roy Schwartz. Min-max graph partitioning and small set expansion. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 17–26. IEEE, 2011.

[19] Boaz Barak, Prasad Raghavendra, and David Steurer. Rounding semidefinite programming hierarchies via global correlation. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 472–481. IEEE, 2011.

[20] 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 *Proceedings of the 42nd ACM symposium on Theory of computing (STOC '10)*, pages 201–210, New York, NY, USA, 2010. ACM.

[21] Aditya Bhaskara, Moses Charikar, Aravindan Vijayaraghavan, Venkatesan Guruswami, and Yuan Zhou. Polynomial integrality gaps for strong SDP relaxations of Densest k-subgraph. In *Proceedings of the Twenty-Third Annual ACM-SIAM*

Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012, pages 388–405. SIAM, 2012.

[22] Yonatan Bilu and Nathan Linial. Are stable instances easy? In *ICS'10*, pages 332–341, 2010.

[23] A. Blum. Some tools for approximate 3-coloring. *Proceedings on the Annual IEEE Symposium on Foundations of Computer Science.*, 0:554–562 vol.2, 1990.

[24] Avrim Blum and Joel Spencer. Coloring random and semi-random k-colorable graphs. *J. Algorithms*, 19:204–234, September 1995.

[25] Ravi B. Boppana. Eigenvalues and graph bisection: An average-case analysis. In *28th Annual Symposium on Foundations of Computer Science.*, pages 280 –285, oct. 1987.

[26] Thang Nguyen Bui, F. Thomson Leighton, Soma Chaudhuri, and Michael Sipser. Graph bisection algorithms with good average case behavior. *Combinatorica*, 7:171–191, June 1987.

[27] 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.

[28] Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Integrality gaps for Sherali-Adams relaxations. In Michael Mitzenmacher, editor, *STOC*, pages 283–292. ACM, 2009.

[29] E. Chlamtac. Approximation algorithms using hierarchies of semidefinite programming relaxations. In *Proceedings of the 48th Annual IEEE Symposium on the Foundations of Computer Science. FOCS '07.*, pages 691 –701, oct. 2007.

[30] Eden Chlamtac and Gyanit Singh. Improved approximation guarantees through higher levels of SDP hierarchies. In Ashish Goel, Klaus Jansen, José D. P. Rolim, and Ronitt Rubinfeld, editors, *APPROX-RANDOM*, volume 5171 of *Lecture Notes in Computer Science*, pages 49–62. Springer, 2008.

[31] Eden Chlamtac and Madhur Tulsiani. Convex relaxations and integrality gaps. *Handbook on Semidefinite, Cone and Polynomial Optimization*, 2011.

[32] Julia Chuzhoy, Yury Makarychev, Aravindan Vijayaraghavan, and Yuan Zhou. Approximation algorithms and hardness of the k-route cut problem. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 780–799. SIAM, 2012.

[33] Amin Coja-Oghlan. A spectral heuristic for bisecting random graphs. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '05, pages 850–859, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[34] Anne Condon and Richard Karp. Algorithms for graph partitioning on the planted partition model. In Dorit Hochbaum, Klaus Jansen, Jos Rolim, and Alistair Sinclair, editors, *Randomization, Approximation, and Combinatorial Optimization. Algorithms and Techniques*, volume 1671 of *Lecture Notes in Computer Science*, pages 221–232. Springer Berlin / Heidelberg, 1999.

[35] Tassos Dimitriou and Russell Impagliazzo. Go with the winners for graph bisection. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, SODA '98, pages 510–520, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.

[36] M. E. Dyer and A. M. Frieze. Fast solution of some random np-hard problems. In *27th Annual Symposium on Foundations of Computer Science (FOCS).*, pages 331 –336, oct. 1986.

[37] U. Feige and J. Kilian. Heuristics for finding large independent sets, with applications to coloring semi-random graphs. In *Proceedings of 39th Annual Symposium on Foundations of Computer Science.*, pages 674 –683, nov 1998.

[38] 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.

[39] Uriel Feige and Joe Kilian. Heuristics for semirandom graph problems. *J. Comput. Syst. Sci.*, 63:639–673, December 2001.

[40] Uriel Feige, Guy Kortsarz, and David Peleg. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[41] Uriel Feige and Robert Krauthgamer. Finding and certifying a large hidden clique in a semirandom graph. *Random Struct. Algorithms*, 16:195–208, March 2000.

[42] Uriel Feige and Michael Seltser. On the densest k-subgraph problem. Technical Report CS97-16, Weizmann Institute of Science, Rehovot, Israel, 1997.

[43] 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.

[44] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC '93, pages 698–707, New York, NY, USA, 1993. ACM.

[45] Konstantinos Georgiou, Avner Magen, Toniann Pitassi, and Iannis Tourlakis. Integrality gaps of 2-o(1) for vertex cover SDPs in the Lovász–Schrijver hierarchy. *SIAM J. Comput.*, 39(8):3553–3570, 2010.

[46] Konstantinos Georgiou, Avner Magen, and Madhur Tulsiani. Optimal Sherali-Adams gaps from pairwise independence. In Irit Dinur, Klaus Jansen, Joseph Naor, and José D. P. Rolim, editors, *APPROX-RANDOM*, volume 5687 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2009.

[47] Doron Goldstein and Michael Langberg. The dense k subgraph problem. *CoRR*, abs/0912.5327, 2009.

[48] Venkatesan Guruswami and Ali Kemal Sinop. Lasserre hierarchy, higher eigenvalues, and approximation schemes for graph partitioning and quadratic integer programming with PSD objectives. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 482–491. IEEE, 2011.

[49] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematics Society (N.S*, 43:439–561, 2006.

[50] Haiyan Hu, Xifeng Yan, Yu Huang, Jiawei Han, and Xianghong Jasmine Zhou. Mining coherent dense subgraphs across massive biological networks for functional discovery. *Bioinformatics*, 21 Suppl 1:i213–21, Jun 2005.

[51] Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001.

[52] M. Jalali, N. Mustapha, A. Mamat, and M.N.B. Sulaiman. A new clustering approach based on graph partitioning for navigation patterns mining. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1 –4, dec. 2008.

[53] Mark Jerrum and Gregory Sorkin. Simulated annealing for graph bisection. In *in Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, pages 94–103, 1993.

[54] George Karypis and Vipin Kumar. *METIS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices*, September 1998.

[55] S. Khot. Improved inapproximability results for maxclique, chromatic number and approximate graph coloring. In *Proceedings on the 42nd IEEE Symposium on Foundations of Computer Science, FOCS '01.*, pages 600 – 609, oct. 2001.

[56] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC '02, pages 767–775, New York, NY, USA, 2002. ACM.

[57] Subhash Khot, Guy Kindler, Elchanan Mossel, and Ryan O'Donnell. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM J. Comput.*, 37(1):319–357, 2007.

[58] Subhash Khot and Rishi Saket. SDP integrality gaps with local ℓ_1-embeddability. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 565–574. IEEE Computer Society, 2009.

[59] Subhash Khot and Nisheeth K. Vishnoi. The unique games conjecture, integrality gap for cut problems and embeddability of negative type metrics into ℓ_1. In *Proceedings of the 44th Annual IEEE Symposium on the Foundations of Computer Science*, FOCS'04, pages 53–62. IEEE, 2005.

[60] Pascal Koiran and Anastasios Zouzias. On the certification of the restricted isometry property. *CoRR*, abs/1103.4984, 2011.

[61] Alexandra Kolla, Konstantin Makarychev, and Yury Makarychev. How to play unique games against a semi-random adversary. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*. IEEE, 2011.

[62] Ludek Kucera, Alberto Marchetti-Spaccamela, Marco Protasi, and Maurizio Talamo. Near optimal algorithms for finding minimum steiner trees on random graphs. In *Mathematical Foundations of Computer Science 1986*, pages 501–511, London, UK, UK, 1986. Springer-Verlag.

[63] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the web for emerging cyber-communities. In *Proceedings of the eighth international conference on World Wide Web*, WWW '99, pages 1481–1493, New York, NY, USA, 1999. Elsevier North-Holland, Inc.

[64] Ravi Kumar and D. Sivakumar. On polynomial approximation to the shortest lattice vector length. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, SODA '01, pages 126–127, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[65] Lap Chi Lau, Joseph (Seffi) Naor, Mohammad R. Salavatipour, and Mohit Singh. Survivable network design with degree or order constraints. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, STOC '07, pages 651–660, New York, NY, USA, 2007. ACM.

[66] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46:787–832, November 1999.

[67] Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.*, 46:259–271, 1990.

[68] Konstantin Makarychev and Yury Makarychev. How to play unique games on expanders. In Klaus Jansen and Roberto Solis-Oba, editors, *WAOA*, volume 6534 of *Lecture Notes in Computer Science*, pages 190–200. Springer, 2010.

[69] Konstantin Makarychev, Yury Makarychev, and Aravindan Vijayaraghavan. Approximation algorithms for semi-random graph partitioning problems. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 367–384. ACM, 2012.

[70] F. McSherry. Spectral partitioning of random graphs. In *Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, FOCS '01, pages 529–, Washington, DC, USA, 2001. IEEE Computer Society.

[71] Mark E. J. Newman. *Random graphs as models of networks*, pages 35–68. Wiley-VCH Verlag GmbH and Co. KGaA, 2005.

[72] Mike Perkowitz and Oren Etzioni. Towards adaptive web sites: conceptual framework and case study. *Artif. Intell.*, 118:245–275, April 2000.

[73] Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, STOC'08, pages 255–264, New York, NY, USA, 2008. ACM.

[74] Prasad Raghavendra. Optimal algorithms and inapproximability results for every CSP? In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, STOC '08, pages 245–254, 2008.

[75] Prasad Raghavendra and David Steurer. Integrality gaps for strong SDP relaxations of unique games. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 575–585. IEEE Computer Society, 2009.

[76] Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 755–764, New York, NY, USA, 2010. ACM.

[77] Prasad Raghavendra, David Steurer, and Madhur Tulsiani. Reductions between expansion problems. In *To appear in the proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, 2012.

[78] Prasad Raghavendra and Ning Tan. Approximating csps with global cardinality constraints using sdp hierarchies. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 373–387. SIAM, 2012.

[79] Grant Schoenebeck. Linear level Lasserre lower bounds for certain k-CSPs. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 593–602, Washington, DC, USA, 2008. IEEE Computer Society.

[80] Grant Schoenebeck, Luca Trevisan, and Madhur Tulsiani. A linear round lower bound for Lovász-Schrijver SDP relaxations of vertex cover. In *IEEE Conference on Computational Complexity*, pages 205–216. IEEE Computer Society, 2007.

[81] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis (motivation and discrete models). In Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Michiel H. M. Smid, editors, *WADS*, volume 2748 of *Lecture Notes in Computer Science*, pages 256–270. Springer, 2003.

[82] Anand Srivastav and Katja Wolf. Finding dense subgraphs with semidefinite programming. In *Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 181–191, 1998.

[83] 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.

[84] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 1 edition, April 2011.