Vertex Sparsification and Universal Rounding Algorithms

by

Ankur Moitra

B.S., Cornell University (2007)
S.M., Massachusetts Institute of Technology (2009)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2011

Vertex Sparsification and Universal Rounding Algorithms

by

Ankur Moitra

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 2011, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

Suppose we are given a gigantic communication network, but are only interested in a small number of nodes (clients). There are many routing problems we could be asked to solve for our clients. Is there a much smaller network - that we could write down on a sheet of paper and put in our pocket - that approximately preserves all the relevant communication properties of the original network? As we will demonstrate, the answer to this question is YES, and we call this smaller network a vertex sparsifier.

In fact, if we are asked to solve a sequence of optimization problems characterized by cuts or flows, we can compute a good vertex sparsifier ONCE and discard the original network. We can run our algorithms (or approximation algorithms) on the vertex sparsifier as a proxy - and still recover approximately optimal solutions in the original network. This novel pattern saves both space (because the network we store is much smaller) and time (because our algorithms run on a much smaller graph).

Additionally, we apply these ideas to obtain a master theorem for graph partitioning problems - as long as the integrality gap of a standard linear programming relaxation is bounded on trees, then the integrality gap is at most a logarithmic factor larger for general networks. This result implies optimal bounds for many well studied graph partitioning problems as a special case, and even yields optimal bounds for more challenging problems that had not been studied before. Morally, these results are all based on the idea that even though the structure of optimal solutions can be quite complicated, these solution values can be approximated by crude (even linear) functions.

Thesis Supervisor: F. Thomson Leighton
Title: Professor

Acknowledgments

I think a doctoral thesis is more a reflection of your personality, your tastes and your passions than of what you know. In writing this thesis, I've come to realize just how much I owe to the strong influences in my life. Both of my parents are computer scientists, but have never pushed computer science on me and have encouraged me in every endeavor. When I eventually settled on computer science, I realized that the reason computer science classes resonated with me is because the ideas and perspectives had been instilled in me through every dinner conversation and every discussion. I owe my parents a tremendous amount for fostering creativity in me, and teaching me to ask the right questions. And if anything, this thesis is about asking just one good question.

I would like to thank my doctoral advisor, Tom Leighton. Tom amazes me on a number of levels and I could not have asked for a better role model, teacher and friend. He has this remarkable way about him. I think it is impossible to be in the same room with Tom and not be in a great mood. Every problem seems solvable, and every idea is easy to communicate. He has always had this unwavering faith in me (well-founded or not), and my best interests have always been his primary concern.

Also I'm grateful to Madhu Sudan and Piotr Indyk for reading my thesis and serving on my committee. Madhu especially has been a great friend to me during my time in graduate school. I remember, during the first conference I attended no one made time to talk to me except Madhu. I can't imagine anyone else at a conference wearing his name tag and yet managing to be so humble and welcoming. I would also like to thank my undergraduate advisor, Eva Tardos, for introducing me to theoretical computer science and for her constant encouragement, advice and candor.

Thank you to all my friends during graduate school - especially Craig Bonnoit, Matt Edwards, Emilio Nanni, Matt Johnston, Mike Petr, Jelani Nelson, Krzysztof Onak, Debmalya Panigrahi, Arnab Bhattacharyya, Nikhil Srivastava, Moritz Hardt, Shaddin Dughmi, Greg Valiant, Rotem Oshman, Adam Kalai and Yael Kalai. Your friendship has meant a lot to me and has made MIT (and MSR) a fantastic experience. Thank you also to my Cornell friends who made my undergraduate years some of the best of my life. Thank you especially to Diana Ding (obviously), Yi Xu, Jeremy Miller, Rahul Malik, Dan Lee, Rob Zimmerman, Liwei Chen, Kevin Moore, Joseph Andrews, Dave Levine and Ian Murray.

Most importantly, I'd like to thank my girlfriend and soulmate, Diana Ding. Everyone I've met seems to find me tough to read; but you understand me instantly, like my thoughts are written on my forehead. In trying times, you help me remember who I am and quite often you know what I want even before I do. Your love and support made this thesis possible. And I'd also like to thank your parents for raising such a caring, self-less and thoughtful daughter. I dedicate my thesis to you, Diana.

This thesis has been supported by a generous Fannie and John Hertz Foundation Fellowship, and an Akamai Presidential Fellowship. I benefitted greatly from being a part of the Hertz community, and having the freedom to pursue my passions.

Contents

List of Figures

Chapter 1

Introduction

The only certainties in life are death, taxes and that communication networks grow and become increasingly complex over time. This natural law gives us, as computer scientists, a mandate. We should develop methods to store these networks, and develop algorithms (for various optimization problems we may want to solve) that still run in some reasonable amount of time on a massive network. There is another possibility. Maybe these networks are not actually all that large.

As a motivating example, suppose we are a content provider, say Akamai, and we have a small number of data centers distributed throughout some gigantic communication network. These data centers store large files, and we may periodically want to transfer some of these files between different data centers. This is a communication problem - our goal is to find some routing scheme in the original network, that achieves large throughput and transfers the contents of these files as quickly as possible. Over time, we may be asked to solve a number of communication problems of this type.

Question 1. *Is there a much smaller communication network that approximates all the relevant communication properties of the original network?*

Can we reduce the number of edges in the communication network? The celebrated results of Benczur and Karger give a positive answer [15]. In general, we can reduce the number of edges so that the total number of edges is nearly-linear in the total number of nodes in the communication network. However, most communication networks are already sparse in this sense. A typical node in the internet backbone connects to only a constant number of neighbors. See Figure 1.2.

We could ask a more bold question: Can we reduce the number of nodes? In our example, we are only interested in transferring files between different data centers. The structure of the original network is only relevant to us in so far as it impacts how we can communicate between data centers. Is there a high-throughput set of paths to transfer the contents of a file from data center A to data center B? Can we simultaneously send a large file from data center A to data center B and from data center C to data center D?

We will ask for something even stronger. Our goal will be to find a communication network *on just the data centers* that approximates all the relevant communication properties of the original network. We will call this graph a vertex sparsifier. For any communication problem we could ever be asked to solve, we want this vertex sparsifier to be a good approximation to the original network. In our example, this vertex sparsifier would be a graph on just *four* nodes. Consequently, we could write this network down on a sheet of paper, and put it in our pocket.

Throughout this paper, we will use G to refer to the original network and H to refer to the corresponding vertex sparsifier. Additionally, we will refer to the data centers as terminals, and we will denote this set as K. For now, we will not address the question of whether a good vertex sparsifier should *exist*. As it turns out, these graphs do exist. In fact, we do not need to make any assumptions at all about the structure of the original network, and for more realistic models of networks, we obtain even stronger results.

What if good vertex sparsifiers do exist? What if there is indeed some much smaller network that captures all the relevant communication properties of a massive communication network? We could hope that knowing a good vertex sparsifier would save us **space** - because we would only need to store a much smaller network. We could also hope to save **time** - because we could run our algorithms on a much smaller network as a proxy for running directly on the original network.

Let us formalize the types of routing problems that we will consider, so that we can state our results more precisely. We will be interested in are minimum congestion routing problems. In general, we will represent a routing problem as a vector $\vec{r}$ of numbers - one number for each pair of terminals. We will call $\vec{r}$ a demand vector. Let a and b be terminals. The number $\vec{r}_{a,b}$ corresponding to the pair (a, b) represents the *rate* at which we would like to transfer information from terminal a to terminal b. A valid routing solution is a choice of flows - one for each pair of terminals - so that for each pair of terminals (a, b) the flow value from a to b is equal to the target rate $\vec{r}_{a,b}$.

Of course, there are many ways to map rates to flows in a communication network. Our goal will be to find a mapping that does not overload the bandwidth of any link by too much. To this end, we can define a notion of the congestion on a link as the total flow traversing the link, divided by the intrinsic bandwidth of the link. Our goal in the minimum congestion routing problem is to find a a valid routing solution that minimizes the worst case congestion on any link. We can intuitively think of the most congested link in a network as being the "bottleneck". So if we are interested in sending large amounts of data, the minimum congestion routing solution will achieve the largest throughput (in steady state).

We ask: Is there a graph H on just the set of terminals, so that (simultaneously) for every demand vector $\vec{r}$, the minimum congestion of routing $\vec{r}$ in G is approximately the same as the minimum congestion of routing $\vec{r}$ in H? We will refer to the multiplicative factor by which H approximates G as the "quality" of H as a vertex sparsifier. See Chapter 2 for formal definitions.

1.1 Our Results.

We can now state our main results:

Good quality vertex sparsifiers exist! In fact, for any capacitated graph $G = (V, E)$ on n nodes, and for any set of k terminals $K \subset V$, there is always a (capacitated) vertex sparsifier *on just the set of terminals* $H = (K, E_H)$ that has quality $O(\log k / \log \log k)$. Surprisingly, this factor is independent of the size of the original network. No matter how massive the original network is, in our example, we can always find a graph on just *four* nodes that captures all relevant communication properties of the original network within a small constant factor. Moreover, this factor depends sub-logarithmically on the number of terminals (or data centers, in our example).

To phrase our results another way: we can imagine that the original communication network will grow over time. Yet there is always some small, faithful representation of the relevant communication properties. This representation will also change over time, but does not *grow*. In our example, this representation is always a network on just four nodes!

For realistic models of networks, our results are even stronger. For "simple" graphs there is always a *constant* quality vertex sparsifier. Hence, the factor by which our vertex sparsifier approximates the original network is actually independent of the number of nodes in the network, and is even independent of the number of terminals. We will give the technical definition of "simple" later, but we note that many common types of networks meet this condition. Planar networks (which can arise in some road networks), graphs that exclude a fixed minor (the internet graph is well-known to exclude a small, fixed minor) and even graphs that have small separators at every scale (as is typical in social networks) all fit this technical condition and hence have a constant quality vertex sparsifier.

Additionally, we can compute a good quality vertex sparsifier quickly - in time quadratic in the size of the original network. This is interesting because the running time of computing a good quality vertex sparsifier matches the running time of solving just a single minimum congestion routing problem. Yet a good vertex sparsifier will help us solve not just one but actually a *sequence* of routing problems quickly. We can compute a good vertex sparsifier just *once* and for any routing problem we are asked to solve, we can compute a near-optimal solution by running a minimum congestion routing algorithm on our vertex sparsifier, as opposed to running directly on the original network. Hence, the amortized time to compute a good routing solution is actually *independent* of the size of the original network, and depends polynomially on the number of terminals.

Finally, approximation really is necessary. Vertex sparsification is an example of a ubiquitous theme in combinatorial optimization, where the goal is to preserve some combinatorial family of parameters on a smaller graph. Vertex sparsification departs from this agenda by asking to preserve a family of parameters that captures *computationally* hard parameters too. A good vertex sparsifier will approximately preserve the value of the sparsest cut and the values of a whole laundry list of other NP-hard graph partitioning problems.

Indeed, Makarychev and Makarychev [48] prove (unconditionally) that we cannot in general hope for a vertex sparsifier with quality better than $\tilde{\Omega}(\sqrt{\log k})$. This lower bound is polynomially related to the upper bound, and is based on exciting connections to functional analysis.

1.2 Applications to Routing.

Given these results, let us return to minimum congestion routing. How does a good vertex sparsifier help us solve a sequence of routing problems quickly? We have only required a good vertex sparsifier to preserve *values*. But what does approximately knowing the minimum congestion we can achieve really tell us about *finding* a good routing?

In fact, there is a canonical way to map good routing schemes in H back to good routing schemes in G. Hence, knowing a good vertex sparsifier not only helps us (approximately) determine the value of a minimum congestion routing in G (without looking at G), we can also determine an approximately optimal routing (again, without looking at G). To explain this principle, let us assume that a good vertex sparsifier is always a better communication network than the original network. Formally, we want H to be such that for any demand vector $\vec{r}$, the minimum congestion of routing $\vec{r}$ in H is always at most the minimum congestion of routing $\vec{r}$ in G. This condition is easy to satisfy – we could just as well set each capacity in H to be infinite. The challenge is in ensuring that H is a faithful representation of the communication properties of G. In fact, the condition that H now be a good vertex sparsifier means that for any demand vector $\vec{r}$, the minimum congestion of routing $\vec{r}$ in H is smaller but must never be too much smaller than the minimum congestion of routing $\vec{r}$ in G.

We can choose H itself as a demand vector - i.e. for each pair of terminals (a, b), we can set the target rate $\vec{r}_{a,b}$ equal to the bandwidth of the link from a to b in H. The resulting demand vector $\vec{r}$ has a good routing solution in H: for each pair of terminals, we can send a flow of value $\vec{r}_{a,b}$ directly on the link (a, b) in H. This explicit routing scheme is a low-congestion routing of $\vec{r}$ in H and since H is a good quality vertex sparsifier, $\vec{r}$ must also have a low-congestion routing in G. We can interpret a low-congestion routing of $\vec{r}$

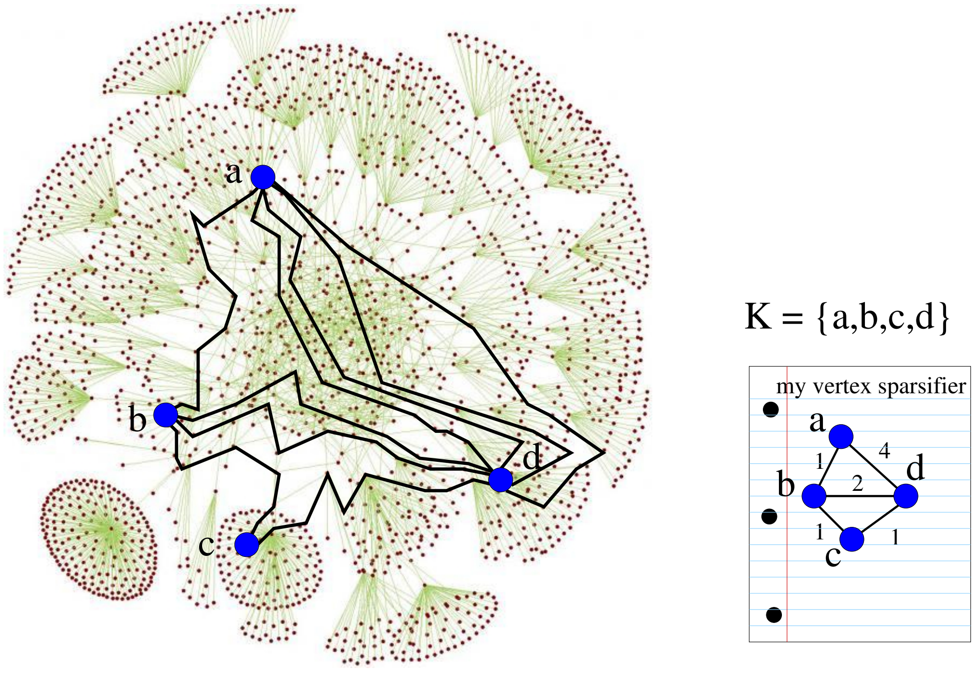

Figure 1-1: Vertex sparsification, applied to the internet backbone.

in G as a method to simulate H in G without too much slow-down (after all, H is a better communication network but not too much better). See Figure 1.2.

For each pair of terminals (a, b), this simulation is literally a flow of value $c_H(a, b)$ from a to b in G (where $c_H(a, b)$ is the bandwidth of the link (a, b) in H). We can regard these flow paths from a to b as the image of the link (a, b) that is present in H. Then we can correspondingly map any path in H to a concatenation of these images in G, and this gives us a fixed method to map a good routing solution in H to a pretty-good routing solution in G. The multiplicative factor by which the congestion increases, when mapping a routing solution back to G, is equal to the quality of H as a vertex sparsifier.

It is not surprising that this is a routing strategy: we can always fix paths between each pair of terminals, and restrict our routing schemes to using only these paths. What is surprising is that there is a method to fix these paths in advance (just by computing a good simulation of H in G) so that whatever routing problem we are asked to solve, restricting ourselves to using these paths is still a near-optimal routing strategy.

Once we have computed a good quality vertex sparsifier H, the incremental running time to compute a (near-optimal) routing of $\vec{r}$ in G is just the time to run a minimum congestion routing algorithm on H. In our example, the amortized running time needed to compute a good routing scheme in the original network can actually be done in *constant* time and is independent of the size of the original network!

1.3 Applications to Graph Partitioning.

Additionally, we apply the ideas of vertex sparsification to obtain a master theorem for graph partitioning problems - as long as there is a good approximation algorithm when the input network is a tree, there is a good approximation algorithm whose approximation ratio is at most a logarithmic (in $|K|$) factor larger for general networks. Often, it is much easier to design an approximation algorithm for a graph partitioning problem that works well on trees than it is to design one for general networks.

Moreover, the actual design of these approximation algorithms (for the general case) is often sensitive to seemingly minor details in the definition of the optimization problem. For example, the goal in generalized sparsest cut is to cut few edges, and disconnect many pairs of demands. Yet the goal in the multi-cut problem is to cut few edges, and disconnect all pairs. In the case of sparsest cut, the optimal rounding algorithm for rounding the standard linear programming relaxation is based on low distortion embeddings into ℓ_1. Yet the optimal rounding algorithm for multi-cut is based on the purely combinatorial technique of *region growing*. Curiously, the answer to how *good* the natural linear programming relaxation is, is the same in both of these cases. The integrality gap is logarithmic in the number of terminals.

Question 2. *Why does a slight variation on the constraints of a graph partitioning problem lead to completely different rounding algorithms (for the standard linear programming relaxation), but ultimately the same approximation factor?*

Here we are able to give a universal rounding algorithm for graph partitioning problems. Given any rounding algorithm (for the standard linear programming relaxation) when the input network is a tree as a black-box, we can apply the ideas of vertex sparsification to obtain a rounding algorithm that works well for general networks. Hence we obtain a robust answer to the above question – these problems all have the same integrality gap, precisely because the integrality gap is constant on trees. We can use this paradigm to give optimal rounding algorithms for many well-studied graph partitioning problems (such as generalized sparsest cut and mult-cut) as a special case and we can even give optimal results for more challenging problems that had not been previously studied.

The central idea behind this result is that minimum congestion routing is mathematically *expressive*. We will call the set of demand vectors $\vec{r}$ that can be routed in G with congestion at most one the *unit congestion polytope*. Then minimum congestion routing is expressive in the sense that the unit congestion polytope exactly determines the values of a number of combinatorial parameters of G. For example, does G have a sparse cut? Does G have a small multi-cut? The values of each of these optimization problems (and many more) can be determined by the values of optimizing (exponentially many) linear functions over the unit congestion polytope. Hence, a good vertex sparsifier H not only approximately preserves the unit congestion polytope but also the values of these combinatorial parameters.

So we can compute a good vertex sparsifier as pre-processing step before running an approximation algorithm for these various problems. In fact, we can take this idea further and ask not only for H to be small (i.e. a graph on just the set of terminals) but also simple in the sense that H can be realized as a convex combination of "contraction-based" trees. This result will allow us to algorithmically reduce a graph partitioning problem to a tree on just the set of terminals. In doing-so, we pay a multiplicative factor in the approximation guarantee which is logarithmic in the number of terminals (which corresponds to the quality of the best vertex sparsifier that can be achieved as a convex combination of "contraction-based" trees). Hence, we obtain our master theorem – and an explanation for why $O(\log |K|)$ always seems to be the right answer for how well a standard linear programming relaxation approximates a myriad of different NP-hard graph partitioning problems.

Next, we give an outline of the organization of this thesis.

1.4 Outline

In Chapter 2 we will state the main definitions and results in this thesis formally. In Chapter 3 we prove that good vertex sparsifiers exist, and we give a fast algorithm for computing good vertex sparsifiers based on multiplicative weights. In Chapter 4 we give an algorithm with improved guarantees, and also we give a universal rounding algorithm for graph partitioning problems. In Chapter 5 we give lower bounds on the quality of a vertex sparsifier, and we give a number of separation results that show that certain restrictions of vertex sparsifiers can be have asymptotically worse quality than others. We also demonstrate that directed vertex sparsification is impossible. In Chapter 6 we give additional applications of vertex sparsification. Specifically, we give optimal results for a certain oblivious routing problem, and we also demonstrate how graph layout problems can also be solved using our framework. In Chapter 7 we highlight open questions.

The results in this thesis appear in [49], [44], and [20]. We note that the results of [48] and [25], which are also on the topic of vertex sparsification, are not included in this thesis, but see the results of Makarychev and Makarychev [48] for the best known lower bound of $\tilde{\Omega}(\sqrt{\log k})$ for vertex sparsification.

Chapter 2

Main Results, Formally

Here we state the main results and introduce notation that will be used throughout this thesis.

2.1 Cut Sparsifiers

We begin by asking for a weaker condition to be met: namely, we want a graph on just the set of terminals that approximately preserves the value of minimum cuts separating *subsets* of terminals.

Suppose we are given an undirected, capacitated graph $G = (V, E)$ and a set $K \subset V$ of terminals of size k. Let $c : E \to \mathbb{R}^+$ be the capacity function of this graph. Let $h : 2^V \to \mathbb{R}^+$ denote the cut function of G:

$$h(U) = \sum_{e \in \delta(U)} c(e)$$

where $\delta(U)$ denotes the set of edges crossing the cut $(U, V - U)$. We define the function $h_K : 2^K \to \mathbb{R}^+$ which we refer to as the terminal cut function on K:

$$h_K(A) = \min_{U \subset V \text{ s.t. } U \cap K = A} h(U)$$

We will also use $\delta_K(A)$ to denote the set of pairs $(a, b) \in K$ which cross the partition

$(A, K - A)$. The combinatorial interpretation of the terminal cut function is that $h_K(A)$ is just the minimum edge cut separating A from $K - A$ in G. Note that A is required to be a subset of K and that we can compute the value and cut achieving $h_K(A)$ using any max-flow algorithm.

Definition 1. *H is a* cut-sparsifier *for the graph $G = (V, E)$ and the terminal set K if H is a graph on just the terminal set K (i.e. $H = (K, E_H)$) and if the cut function $h' : 2^K \rightarrow \mathbb{R}^+$ of H satisfies (for all $A \subset K$)*

$$h_K(A) \leq h'(A).$$

The goal of a cut-sparsifier is to everywhere approximate the terminal cut function. So then we can define a notion of quality for any particular cut-sparsifier, which captures how faithfully the cut function of H *everywhere* approximates the terminal cut function:

Definition 2. *The* quality *of a cut-sparsifier H is defined as*

$$max_{A \subset K} \frac{h'(A)}{h_K(A)}.$$

We will abuse notation and define $\frac{0}{0} = 1$ so that when A is disconnected from $K - A$ in G or if $A = \emptyset$ or $A = K$, the ratio of the two cut functions is 1 and we ignore these cases when computing the worst-case ratio and consequently the quality of a cut-sparsifier.

Theorem 1. *For any capacitated graph $G = (V, E)$, for any set $K \subset V$ of $|K| = k$ terminals, there is an $O(\log k / \log \log k)$-quality cut-sparsifier $H = (K, E_H)$. If G excludes $K_{r,r}$ as a minor, then there is an $O(r^2)$-quality cut-sparsifier.*

This result appears in Section 3.1.

Many naturally occurring networks do in fact exclude a small, fixed minor: road networks are often (close to) planar, the internet graph is well-known to have small treewidth, and even social networks have the so-called padded decomposition property which is also enough to guarantee that constant quality cut-sparsifiers exist.

2.2 Comparison

There are many results in combinatorial optimization that demonstrate that certain graph connectivity properties can be approximated on a smaller graph. For example, if we define the *local connectivity* of two terminals $a, b \in K$ as the minimum cut in G separating a and b, then Mader's Theorem (see [43]) implies that there is a graph $H = (K, E_H)$ so that for all pairs of terminals $a, b \in K$, H exactly preserves the local connectivity.

Yet results of this form only guarantee that H preserves minimum cuts separating subsets of terminals for *small* subsets of terminals. Here, preserving the local connectivity only requires preserving the minimum cuts separating single pairs of terminals from each other. Consider, for example, the graph G which is the complete bipartite graph with the k terminals on one side and 2 nodes on the other. The local connectivity between any pair of terminals is 2, and applying the splitting-off operation in Mader's Theorem iteratively results in the graph $H = (K, E_H)$ which is a cycle. This preserves the local connectivity exactly, and yet if we bisect the cycle we get an edge cut of size 2 in H that cuts the graph into two $\Omega(k)$-sized sets of terminals A and $K - A$. But the capacity of the minimum cut separating A from $K - A$ in G is $\Omega(k)$. So the cut function of H does not well approximate the terminal cut function everywhere. And in general, results in combinatorial optimization about preserving minimum cuts separating small subsets of terminals will be useless for our purposes.

We also note that many graph partitioning problems (for example generalized sparsest cut, or generalized bisection type problems) depend on minimum cuts separating A and $K - A$ for *large* sized sets A. So if we are given a graph H which approximately preserves just the small terminal cuts, we cannot guarantee that mapping, say, a generalized sparsest cut problem to H approximately preserves the value of the optimum. So if we want to perform reductions that are oblivious to the particular optimization problem, we really do need to preserve all minimum cuts separating every subset of terminals, and the above question really is the right question in this context.

There is also an extensive line of work on edge sparsification - where the goal is to find a (capacitated) graph $G' = (V, E')$ so that for any cut in G, the corresponding cut in G' has value within an $1 \pm \epsilon$ factor. Benczur and Karger give a fast, random sampling approach which achieves $|E'| = O(|V| \log |V|)$ [15]. Generalizing these results, Spielman and Teng construct a nearly-linear sized graph G' that preserves the spectrum of the Laplacian [58]. Batson, Spielman and Srivastava recently proved that in fact, only $O(|V|)$ edges are needed to preserve the spectrum of the Laplacian [13]. However, we underscore that these results do not reduce the number of nodes in a graph.

Additionally, Räcke proves that a capacitated graph can be replaced by a distribution on ”decomposition trees” that approximately preserves the value of the minimum congestion routing for each demand vector $\vec{r}$ within an $O(\log |V|)$ factor [53], [54]. This decomposition has been useful in designing approximation algorithms for a wide range of graph partitioning problems. The results of Räcke can also be used to obtain a reasonable quality cut (or flow) sparsifier, but unlike the results in this thesis the approximation guarantee will depend (logarithmically) on the size of the original graph. Moreover, the results in [54] do not improve in the case of graphs that exclude a small, fixed minor. By comparison, in this case we are able to compute a cut (or flow) sparsifier whose quality is a constant independent of the size of the original graph, and independent of the number of terminals. We are also able to get stronger guarantees for other problems from our results. For example, in Section 6.1 we are able to obtain an oblivious routing scheme (for demand vectors restricted to a set $K \subset V$ of clients) whose competitive ratio is $O(\log |K|)$ while the results in [54] only yield an $O(\log |V|)$ competitive ratio.

2.3 Flow Sparsifiers

Next, we ask for a stronger condition to be met: we want a graph on just the set of terminals that approximately preserves minimum congestion routing. Let us first define the maximum concurrent flow problem:

An instance of the maximum concurrent flow problem consists of an undirected graph $G = (V, E)$, a capacity function $c : E \to \mathbb{R}^+$ that assigns a non-negative capacity to each edge, and a set of demands $\{(s_i, t_i, f_i)\}$ where $s_i, t_i \in V$ and f_i is a non-negative demand. We denote $K = \cup_i \{s_i, t_i\}$. The maximum concurrent flow question asks, given such an instance, what is the largest fraction of the demand that can be simultaneously satisfied? This problem can be formulated as a polynomial-sized linear program, and hence can be solved in polynomial time. However, a more natural formulation of the maximum concurrent flow problem can be written using an exponential number of variables.

For any $a, b \in V$ let $P_{a,b}$ be the set of all (simple) paths from a to b in G. Then the maximum concurrent flow problem and the corresponding dual can be written as :

$$\begin{array}{ll} \max & \lambda \\ \text{s.t.} & \\ & \sum_{P \in P_{s_i,t_i}} x(P) \geq \lambda f_i \\ & \sum_{P \ni e} x(P) \leq c(e) \\ & x(P) \geq 0 \end{array} \qquad \begin{array}{ll} \min & \sum_e d(e)c(e) \\ \text{s.t.} & \\ & \forall_{P \in P_{s_i,t_i}} \sum_{e \in P} d(e) \geq D(s_i, t_i) \\ & \sum_i D(s_i, t_i) f_i \geq 1 \\ & d(e) \geq 0, D(s_i, t_i) \geq 0 \end{array}$$

For a maximum concurrent flow problem, let λ^* denote the optimum.

Let $|K| = k$. Then for a given set of demands $\{s_i, t_i, f_i\}$, we associate a vector $\vec{f} \in \mathbb{R}^{\binom{k}{2}}$ in which each coordinate corresponds to a pair $(x, y) \in \binom{K}{2}$ and the value $\vec{f}_{x,y}$ is defined as the demand f_i for the terminal pair $s_i = x, t_i = y$.

Definition 3. *We denote $cong_G(\vec{f}) = \frac{1}{\lambda^*}$*

Or equivalently $cong_G(\vec{f})$ is the minimum C s.t. $\vec{f}$ can be routed in G and the total flow on any edge is at most C times the capacity of the edge.

Definition 4. *$H = (K, E_H)$ is a flow-sparsifier if for all $\vec{f} \in \mathbb{R}^{\binom{k}{2}}$, $cong_H(\vec{f}) \leq cong_G(\vec{f})$*

So a flow-sparisifer is a ”better” flow network than the original graph, in the sense that all demands (with endpoints supported in K) that are routable in G (with congestion at most 1) can also be routed in H.

Definition 5. *The quality of a flow-sparsifier H is $\max_{\vec{f} \in \mathbb{R}^{\binom{k}{2}}} \frac{cong_G(\vec{f})}{cong_H(\vec{f})}$*

Theorem 2. *For any capacitated graph $G = (V, E)$, for any set $K \subset V$ of $|K| = k$ terminals, there is an $O(\log k / \log \log k)$-quality flow-sparsifier $H = (K, E_H)$. If G excludes $K_{r,r}$ as a minor, then there is an $O(r^2)$-quality flow-sparsifier.*

This result appears in Section 3.2. See also Section 3.2.2 for a comparison between cut and flow-sparsification. In particular, flow-sparsification is no easier than cut-sparsification and conversely flow-sparsifiers can be obtained from cut-sparsifiers, with some degradation in quality corresponding to how well multicommodity flows are approximated by sparsest cuts.

We also give a fast algorithm for computing a good flow-sparsifier (and hence the graph that this algorithm returns is also a good cut-sparsifier):

Theorem 3. *There is an algorithm that for any capacitated graph $G = (V, E)$, for any set $K \subset V$ of $|K| = k$ terminals, computes an $O(\log k/ \log \log k)$-quality flow-sparsifier $H = (K, E_H)$ in time $\tilde{O}(km^2)$ where $m = |E|$.*

Theorem 4. *There is an algorithm that for any capacitated graph $G = (V, E)$ that excludes $K_{r,r}$ as a minor and for any set $K \subset V$ of $|K| = k$ terminals, computes an $O(r^2)$-quality flow-sparsifier $H = (K, E_H)$ in time $\tilde{O}(km^2)$ where $m = |E|$.*

Here, the hidden constant depends polynomially on r. These results appear in Section 3.2.4, and in Section 4.1.2 we give another algorithm for computing a good quality flow-sparsifier, and this algorithm has the additional benefit that the output is guaranteed to have quality that is at least as good as the best quality that can be achieved through "contractions". Hence, if the integrality gap for the semi-metric relaxation for the 0-extension problem (see Section 3.1.1) is $O(\sqrt{\log k})$, this algorithm already computes an $O(\sqrt{\log k})$-quality flow-sparsifier.

2.4 Universal Rounding Algorithms

An important application of vertex sparsification is that given a good quality vertex sparsifier H, an approximation algorithm can be run on H as a proxy for running directly on G. Because the size (number of nodes) of H is $|K|$, any approximation algorithm that achieves a $poly(\log |V|)$-approximation guarantee in general will achieve a $poly(\log |K|)$ approximation guarantee when run on H (provided that the quality α is also $poly(\log |K|)$). Feasible solutions in H can also be mapped back to feasible solutions in G for many of these problems, so polynomial time constructions for good cut-sparsifiers yield black box techniques for designing approximation algorithms with guarantees $poly(\log |K|)$ (and independent of the size of the graph).

Hence, we pay an additional price in the approximation guarantee that corresponds to how well H approximates G. We can ask:

Question 3. *Do we really need to pay a price (in the approximation guarantee) when applying vertex sparsification to an optimization problem?*

In many cases, we can combine the steps of computing a good quality vertex sparsifier, and computing a near-optimal solution to an optimization problem (on the vertex sparsifier). We can simultaneously ask for a vertex sparsifier that has good quality and also on which our optimization problem is easy.

In particular, we observe that we can compute $O(\log k)$-quality flow-sparsifiers (and hence have a slightly worse quality factor) but which can be realized as a convex combination of (in a technical sense) contraction-based trees. A similar idea appears in [25].

Definition 6. *We call an optimization problem D a fractional graph partitioning problem if it can be written as (for some monotone increasing function f):*

$$\begin{aligned}\min \quad & \textstyle\sum_{(u,v)\in E} c(u,v)d(u,v) \\ \textit{s.t.} \quad & \\ & d : V \times V \to \Re^+ \textit{ is a semi-metric} \\ & f(d\big|_K) \geq 1\end{aligned}$$

We refer to this as a graph partitioning problem because the goal is to separate pairs of terminals, without stretching too many high capacity edges. We will call the dual problem P a fractional packing problem. To make this definition seem more natural, we demonstrate that a number of well-studied problems fit into this framework for an appropriate choice of the monotone increasing function f.

Let ID[1] denote the integral dual graph partitioning problem. We will refer to the relaxation in Definition 6 D for ID as the semi-metric relaxation.

Example 1. *[45], [46], [8] P: maximum concurrent flow; ID: generalized sparsest cut*

Here we are given some demand vector $\vec{r} \in \mathbb{R}^{\binom{K}{2}}$, and the goal is to maximize the value C such that $C\vec{r}$ is feasible in G. Then the dual to this problem corresponds to minimizing the total distance $\times$ capacity units, subject to the constraint that $\sum_{(a,b)} \vec{r}_{a,b} d(a,b) \geq 1$, where d is the induced semi-metric on K. This function is clearly a monotone increasing function of the distances between pairs of terminals, and hence is an example of what we call a graph packing problem.

Example 2. *[31] P: maximum multiflow; ID: multicut*

Here we are given some pairs of terminals $T \subset \binom{K}{2}$, and the goal is to find a demand vector $\vec{r}$ that can be routed in G that maximizes $\sum_{(a,b) \in T} \vec{r}_{a,b}$. The dual to this problem corresponds to minimizing the total distance $\times$ capacity units, subject to the constraint that $\min_{(a,b) \in T} \{d(a,b)\} \geq 1$, and this again is a monotone increasing function of the distances between pairs of terminals, and hence is also an example of what we call a graph packing problem.

Example 3. *ID: Steiner multi-cut*

Example 4. *ID: Steiner minimum-bisection*

Example 5. *[51] P: multicast routing; ID: requirement cut*

This is another partitioning problem, and the input is again a set of subsets $\{R_i\}_i$. Each subset R_i is also given at requirement r_i, and the goal is to minimize the total capacity removed from G, in order to ensure that each subset R_i is contained in at least r_i different components. Similarly to the Steiner multi-cut problem, the standard relaxation for this problem is to minimize the total amount of distance $\times$ capacity units allocated in G, s.t. for each i the minimum spanning tree T_i (on the induced metric on K) on every subset R_i has total distance at least r_i. This again can be cast into the above framework, by writing this constraint as $\min_i \{ \frac{\min_{T \in \Pi_i} \sum_{(a,b) \in T} d(a,b)}{r_i} \} \geq 1$ (where Π_i denotes the set of trees on R_i). The dual problem P is often solved in order to find good multicast routing schemes.

Hence the notion of a fractional graph partitioning problem is quite general, and captures many interesting, well-studied optimization problems as a special case. This definition even captures much more expressive and challenging optimization problems that had not been studied before. Yet there is still enough structure to this definition to define a universal rounding algorithm, that simultaneously works well for this entire class of optimization problems. No matter what we choose for the monotone increasing function f, the value of the semi-metric relaxation on depends on the set of demand vectors that can be routed in G with congestion at most one. So we can use the ideas of vertex sparsification to algorithmically reduce the input graph to a tree, while still approximately preserving the integrality gap of the semi-metric relaxation:

Theorem 5. *For any fractional graph partitioning problem D, the maximum integrality gap of the semi-metric relaxation is at most $O(\log k)$ times the maximum integrality gap restricted to trees.*

We can make this algorithmic:

Theorem 6. *Given a polynomial time rounding algorithm for the semi-metric relaxation on trees that achieves an approximation ratio of C, there is a polynomial time rounding algorithm for the semi-metric relaxation on general graphs that achieves an approximation ratio of $O(C \log k)$.*

Hence, designing a rounding algorithm that works for general graphs is no harder than designing one that works on trees. These results appear in Section 4.2.

2.5 Lower Bounds and Separations

In many interesting cases, constant quality cut-sparsifiers exist. Yet in general, approximation is necessary and there are families of graphs that have no constant quality cut-sparsifiers. This implies that there are no constant quality flow-sparsifiers for these families either. The best lower bound is due to Makarychev and Makarychev [48]) who give a family of graphs for which no cut-sparsifier has quality better than $\tilde{\Omega}(\sqrt{\log k})$. This result

[1]The notion of what constitutes an integral solution depends on the problem. In some cases, it translates to the requirement that all distances be 0 or 1, and in other cases it can mean something else. The important point is that the notion of integral just defines a class of admissible metric, as opposed to arbitrary metrics which can arise in the packing problem.

is based on exciting connections to the Lipschitz extendability problem which has been studied in functional analysis since the 1950s.

Here, we give an elementary $\Omega(\log^{1/4} k)$ lower bound that relies only on duality and estimates for binomial coefficients.

Theorem 7. *There is an infinite family of graphs for which no cut-sparsifier of quality better than* $\Omega(\log^{1/4} k)$.

This result appears in Section 5.1.1.

We can show that for this particular family of graphs, no contraction based cut-sparsifier can have quality better than $\Omega(\sqrt{\log k})$. Yet for this family of graphs, we demonstrate a family of cut-sparsifiers based on random walks that achieve quality $o(\sqrt{\log k})$. Interestingly, the upper bounds that we present (and in fact, all known upper bounds for general graphs) produce vertex sparsifiers through contractions. Yet, this result allows us to demonstrate that the best quality cut-sparsifiers do not always result from contractions:

Theorem 8. *There is an infinite family of graphs so that the ratio of the best quality cut-sparsifier to the best quality cut-sparsifier that can be achieved through a distribution on contractions is* $o(1) = O(\frac{\log\log\log\log k}{\log^2 \log\log k})$

This result appears in Section 5.1.

Yet another wrinkle to this result is that flow-sparsifiers that are based on contractions actually preserve more than just the congestion of multicommodity flows. In fact, contraction based flow-sparsifiers of quality α also preserve network coding rates within the same α factor as well. In contract, a flow-sparsifier that is not contraction based that has quality α is not known to also preserve network coding rates within this factor - in part because the relationship between network coding rate and multicommodity flow rate is poorly understood and in fact has implications for circuit lower bounds.

So requiring a flow-sparsifier to be generated from a convex combination of contractions actually requires it to do more than just approximately preserve the congestion of every multicommodity flow problem. And this problem is asymptotically harder, in some cases, than the problem of just preserving multicommodity flow rates. See Section 3.2.5.

We also give another separation, which demonstrates that flow-sparsification can be asymptotically harder than cut-sparsification:

Theorem 9. *There is an infinite family of graphs so that the best quality flow-sparsifier has quality* $\Omega(\log \log k)$ *and yet the best quality cut-sparsifier has quality* $O(1)$.

The quality of the best flow-sparsifier and the quality of the best cut-sparsifier are always related within a poly-logarithmic (in k) factor. These results appear in Section 5.2.

Finally, we prove that flow-sparsification is impossible in directed graphs.

Theorem 10. *There is an infinite family of directed graphs so that the quality of the best flow-sparsifier is* $\Omega(k)$

This result appears in Section 5.3. Hence, the results that we obtain in this thesis cannot be generalized to directed graphs.

Chapter 3

Vertex Sparsifiers

Here we prove that there are $O(\frac{\log k}{\log \log k})$-quality vertex sparsifiers (both for approximating cuts and multicommodity flows). This is a global structural result, but we prove this by introducing a zero-sum game between an extension player and a quality player. The extension player constructs a graph H that is a vertex sparsifier, and the quality player checks the quality of this graph. We define this game in such a way that bounding the game value of this game that good quality vertex sparsifiers exist.

We can then bound the game value of this game by proving that there is a good response for the extension player for every distribution on checks that the quality player makes. We use a rounding procedure due to Fakcharoenphol, Harrelson, Rao and Talwar [26] for the 0-extension problem to produce such a good response.

Thus, the intuition for why good quality vertex sparsifiers should exist *does not* come from looking at a graph G, and a set $K \subset V$ of terminals, and determining that there is some way to approximately preserve all these exponentially many minimum cuts (or, for that matter all the exponentially many multicommodity flows) on a smaller graph. Rather, the intuition comes from imagining an adversary trying to disprove the statement that G has a good quality vertex sparsifier, and showing that this adversary in fact cannot disprove this statement. The beauty of the Min-Max Theorem is that this is enough to imply that good quality vertex sparsifiers exist.

3.1 Cut Sparsifiers

The results in this section are based on [49].

3.1.1 0-Extensions

We begin by examining the operation of contracting edges in G. We observe that contracting an edge (whose endpoints are not both terminals) cannot decrease the value of the minimum cut separating A from $K - A$ for $A \subset K$. This is because if we contract an edge $e = (u, v)$ in G to obtain a graph G/e, the minimum cut in G/e separating A from $K - A$ is just the minimum cut in G separating A from $K - A$ that also contains u and v on the same side of the cut.

Hence, if we contract edges in G until the only remaining nodes are the set of terminals K, the resulting graph H will in fact be a cut-sparsifier. Obtaining a cut-sparsifier is easy – after all, we could just choose a complete weighted graph H whose edges all have infinite weight, but this would be a terrible *quality* cut-sparsifier. Similarly any graph H on just the set of terminals, obtained through contractions will not necessarily be a good quality cut-sparsifier either.

Nevertheless contractions and distributions on contractions will be our method of producing not just cut-sparsifiers, but good cut-sparsifiers. Actually, we will use a slightly more general type of operation which is still guaranteed to make the values of minimum cuts (separating subsets of terminals) only increase. This operation is referred to as a 0-extension and was originally formulated in [39] by Karzanov who used this operation to define a type of graph partitioning problem that generalizes minimum multiway cut problem.

Suppose we are given a semi-metric D on the terminals (in addition to a weighted graph G and $K \subset V$). Then the goal of the 0-extension problem is to assign each node in V to a terminal in K (and each terminal $t \in K$ must be assigned to itself) such that the sum over all edges (u, v) of $c(u, v)$ times the distance between u and v under the metric D is minimized.

Definition 7. *We will call a function $f : V \rightarrow K$ such that $f(t) = t$ for all $t \in K$ a*

0-*extension function*

So the goal of the 0-extension problem is to minimize

$$\sum_{(u,v)\in E} c(u,v)D(f(u),f(v))$$

over all 0-extension functions f. When D is just the uniform metric on the terminals K, this exactly the minimum multiway cut problem. Karzanov gave a (semi)metric relaxation of the 0-extension problem [39]:

$$\begin{aligned} \min \quad & \textstyle\sum_{(u,v)\in E} c(u,v)\beta(u,v) \\ \text{s.t.} \quad & \\ & \beta \text{ is a semi-metric on } V \\ & \forall_{t,t'\in K}\beta(t,t') = D(t,t'). \end{aligned}$$

Note that the semi-metric β is defined on V while D is defined only on K. Let OPT^* denote the value of an optimal solution to the above linear programming relaxation of the 0-extension problem. Clearly $OPT^* \leq OPT$. Calinescu, Karloff and Rabani [18] gave a randomized rounding procedure to round any feasible solution β of value C to a 0-extension that has expected value at most $O(\log k)C$. Fakcharoenphol, Harrelson, Rao and Talwar [26] gave an improved randomized rounding procedure that achieves an $O(\frac{\log k}{\log\log k})$ approximation ratio:

Theorem 11. *[26]*

$$OPT^* \leq OPT \leq O\left(\frac{\log k}{\log\log k}\right) OPT^*$$

If G excludes $K_{r,r}$ as a minor, then Calinescu, Karloff and Rabani [18] gave an improved rounding algorithm that achieves an $O(r^2)$ approximation ratio. The bound of $O(r^3)$ is implicit in [18], and can be immediately improved to $O(r^2)$ using the results of [28] (which provide improved bounds for padded decompositions for graphs excluding $K_{r,r}$ over those in [42]). Then this immediately implies as a corollary:

Theorem 12. *[18]*

$$OPT^* \leq OPT \leq O(r^2)OPT^*$$

This relaxation, and these rounding algorithms for the (semi)metric relaxation will be a key ingredient in our proof that good quality vertex sparsifiers exist.

As promised, we can use 0-extension functions to generate a graph on just the terminals that is a cut-sparsifier: Given a 0-extension function f, we can define the capacitated graph G_f on K that results from the function f as:

$$c_f(a,b) = \sum_{u,v | f(u)=a, f(v)=b} c(u,v)$$

We will abuse notation and refer to the graph G_f generated by f as a 0-extension of the graph G. We will use h_f to denote the cut function of the resulting graph G_f. Then in fact, for any 0-extension f, the graph G_f is a cut-sparsifier:

Claim 1. *For any subset of terminals $U \subset K$,*

$$h_K(U) \leq h_f(U).$$

Proof: For any $u \in K$, let $f^{-1}(u) = \{a \mid f(a) = u\}$. Let $A = \cup_{u \in U} f^{-1}(u)$. By definition, $h_f(U) = h(A)$ and because $A \cap K = U$ this implies that $h(A) \geq h_K(U)$, because the cut $A, V - A$ is a cut separating U from $K - U$ so it is at least the minimum cut-separating U from $K - U$. $\square$

As we will see later, G_f is also necessarily a flow-sparsifier – as the 0-extension function not only makes the values of minimum cuts separating subsets of terminals increase, but also makes the congestion of multicommodity flows with demands in K decrease.

3.1.2 Intuition

We now have a method to generate a cut-sparsifier - for any 0-extension f, $G_f = (K, E_f)$ is a cut-sparsifier. Of course, it is easy to construct a cut-sparsifier. We could just as well set all capacities in $H = (K, E_H)$ to be infinite, and certainly the value of any cut in H will be at least as large as the value of the corresponding minimum cut in G. However, we are interested in finding a good *quality* cut-sparsifier.

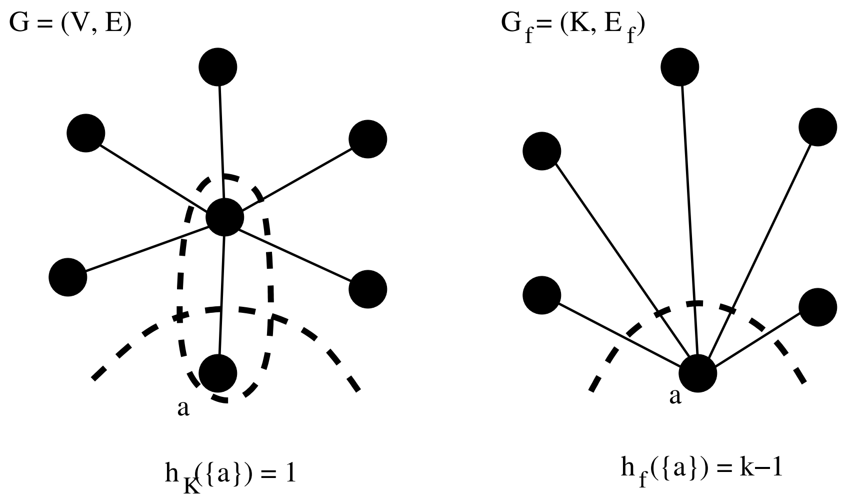

Figure 3-1: A single 0-extension f produces a cut-sparsifier, but not necessarily a good quality cut-sparsifier.

We could hope that a single, appropriately chosen 0-extension f will result in not just a cut-sparsifier but actually a good quality cut-sparsifier. Unfortunately this is not the case. Consider the example in Figure 3-1. All graphs G_f generated by a 0-extension f are isomorphic - the only choice we can make in choosing f is where to map the single non-terminal node, and all choices are the same up to a permutation of the labels of the terminals. Yet G_f has terrible quality. Let u (the single non-terminal) be assigned to terminal a. Then the value of the minimum cut separating $\{a\}$ from $K - \{a\}$ in G is one, and yet the value of the corresponding cut in G_f is $k - 1$. Hence G_f has quality $k - 1$ as a cut-sparsifier, and out goal is to give an $O(\frac{\log k}{\log \log k})$-quality cut-sparsifier. Here we are losing a union bound over each terminal!

We will need to use a distribution on 0-extensions. In this example, we can choose each of the k 0-extensions f with equal probably (i.e. with probability $\frac{1}{k}$). Let γ be the resulting distribution on 0-extensions. Consider the graph $H = \sum_f \gamma(f) G_f$ and let h' be the corresponding cut function. See Figure 3-2. This graph is a cut-sparsifier, and has quality at most 2 because for any cut $(A, K - A)$ the value of the minimum cut in G is $\min(|A|, |K - A|)$ and the value of the cut in H is $\frac{2}{k}|A||K - A| \leq 2\min(|A|, |K - A|)$.

In general, the distribution γ on 0-extensions that we choose must depend heavily on which minimum cuts in G are large and which are small, and how these cuts overlap.

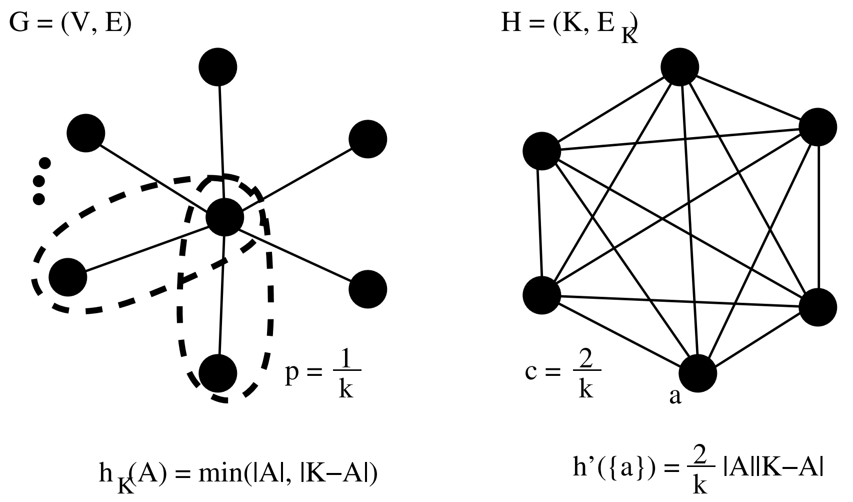

Figure 3-2: Constructing a good quality cut-sparsifier via a distribution on 0-extensions.

Consider the example in Figure 3-3. Here there are two non-terminals. If we choose a 0-extension f that sends both of these non-terminals to the same terminal with probability $\frac{1}{k}$, then we obtain the same graph H as in Figure 3-2. This graph is a cut-sparsifier for the graph G in Figure 3-3, however it is not a good quality cut-sparsifier. The graph G has a sparse cut that bisects the set of terminals while cutting only a single edge. Yet any cut that bisects the set of terminals has value at least $\Omega(k)$ in the graph H in Figure 3-2.

Hence, we need to choose some distribution on 0-extensions that preserves this sparse cut. Suppose we map the non-terminal on the left to a randomly chosen terminal on the left, and we map the non-terminal on the right to a randomly chosen terminal on the right. Any graph G_f that we obtain in this way does indeed preserve this sparse cut that bisects the set of terminals. If we then average over all such choices of f, we obtain a cut-sparsifier that has good quality.

Hence, in order to prove that a good quality cut-sparsifier exists (that can be realized as a distribution on 0-extensions), the distribution γ that we choose must depend on the minimum cut-structure of G. Every time we choose some 0-extension f to have positive support in γ, there are some minimum cuts in G that are preserved after contraction and some that are not. How can we find a good distribution γ without explicitly remembering how well we are currently approximating each one of the exponentially many minimum cuts that we are interested in? We will instead use a zero-sum game to prove that good

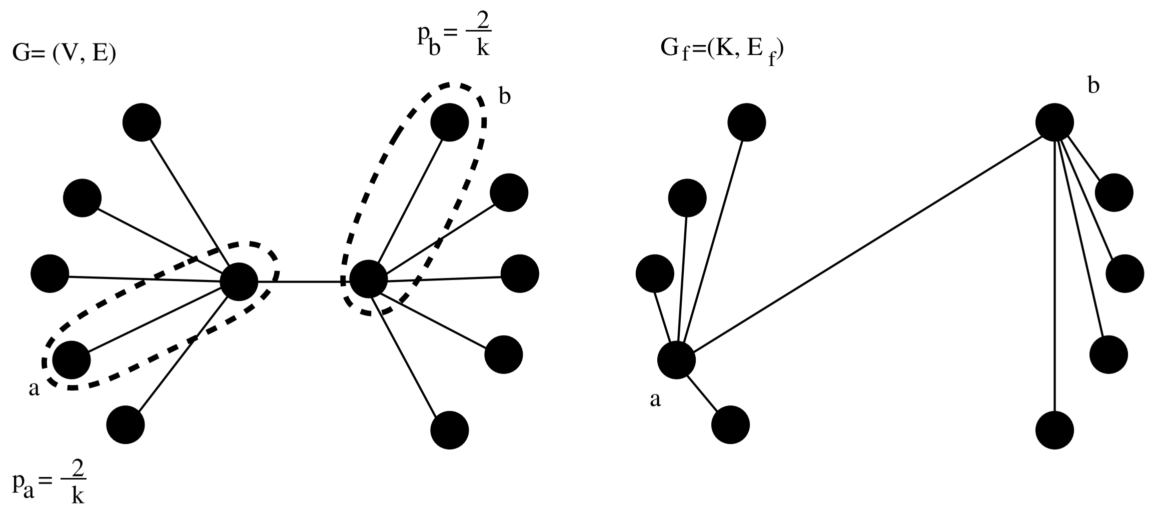

Figure 3-3: The distribution on 0-extensions that we choose depends crucially on the minimum-cut structure in G.

cut-sparsifiers exist. Hence, playing against an adversary reveals to us the right distribution γ that approximately preserves all exponentially many minimum cuts in G.

3.1.3 A Zero-Sum Game

Here we introduce and analyze an appropriately chosen zero-sum game, so that a bound on the game value of this game will imply the desired structural graph theory result.

Given an undirected, capacitated graph $G = (V, E)$ and a set $K \subset V$ of terminals, an extension player (P1) and a quality player (P2) play the following zero-sum game that we will refer to as the extension-quality game:

The **extension player** (P1) chooses a 0-extension f

The **quality player** (P2) chooses a cut from 2^K

Given a strategy f for P1 and a strategy A for P2, P2 wins $\frac{1}{h_K(A)}$ units for each unit of capacity crossing the cut $(A, K - A)$ in P1's 0-extension. Also, we restrict P2 to play only strategies A for which $h_K(A) \neq 0$. So if P1 plays a strategy f and P2 plays a strategy A then P2 wins:

$$N(f, A) = \sum_{(u,v)\in E} \frac{1_{(f(u),f(v))\in\delta_K(A)}c(u, v)}{h_K(A)}$$

This completely specifies the game. Both players have exponential-sized strategy spaces in this game. The extension player can choose any one of the exponentially many 0-extension functions, and there are k^{n-k} such functions. The quality player can choose any one of the exponentially many cuts from 2^K.

Despite being asymptotically large, this game is still finite for any fixed G and K. Hence the celebrated minmax theorem of von Neumann implies that this game has a *game value* – which roughly means that two different mechanisms to play this game are equivalent:

We could either force P1 to play first and choose a randomized strategy, which in this case is a distribution on 0-extension functions, and then allow P2 to choose any strategy knowing full-well what randomized strategy P1 has chosen. We could instead force P2 to play first and choose a randomized strategy, which would be a distribution on cuts to check, and then allow P1 to choose any strategy knowing what randomized strategy P2 chose.

Von Neumann's minmax theorem states that the best (expected) value that P1 can achieve in both these scenarios is equivalent, and this quantity is called the game value. The intuition for why the game value should be useful for our purposes is that a bound on the game value implies that there is some randomized strategy that P1 can choose, so that P2 cannot do well (can do at most as well as the game value) even knowing P1's randomized strategy. Hence there is some distribution on 0-extension functions which, no matter what cut P2 chooses, P2 cannot find a cut to check that in expectation is multiplicatively too much larger than the value of the corresponding minimum cut in G.

Definition 8. *Let ν denote the game value of the extension-quality game.*

We can bound the game value by bounding the cost of P1's best response to any fixed, randomized strategy for P2. So consider any randomized strategy μ for P2. μ is just a probability distribution on 2^K. We can define an ℓ_1 metric on K:

$$D_\mu(t,t') = \sum_{A \subset K} \mu(A) \frac{1_{(t,t') \in \delta_K(A)}}{h_K(A)}$$

D_μ is a weighted sum of cut-metrics on K. Given D_μ we can define a semi-metric β that is roughly consistent with D_μ. This semi-metric will serve as a feasible solution to the linear programming relaxation for the 0-extension problem. A bound on the cost of this feasible solution and the rounding algorithm due to [26] will together imply that there is a 0-extension that has not too much cost, and this will imply that the extension player has a good response to the strategy μ. We define β as:

Initially set all edge distances $d(u, v)$ to zero. Then for each $A \subset K$, if there is no unique minimum cut separating A and $K - A$, choose one such minimum cut arbitrarily. For this minimum cut, for each edge (u, v) crossing the cut, increment the distance $d(u, v)$ by $\frac{\mu(A)}{h_K(A)}$.

Then let β be the semi-metric defined as the shortest path metric on G when distances are $d(u, v)$.

Claim 2. *$\beta(t, t') \geq D_\mu(t, t')$ for all terminals t, t'.*

Proof: Consider any particular pair of terminals t, t'. Consider the contribution of any particular $A \subset K$ to the semi-metric D_μ. A only contributes if t and t' are separated with respect to A - i.e. if exactly one of $\{t, t'\}$ is in A. We will show that any set A that contributes to D_μ, has at least as large a contribution to β. So consider a set A that contains t but not t'. Then any path in G from t to t' must cross the cut in G that actually achieves the minimum $h_K(A)$. But when considering the set A, we increased the distance on all edges crossing this cut by $\frac{\mu(A)}{h_K(A)}$, so the contribution to $\beta(t, t')$ of the set A is at least as large as the contribution of A to $D_\mu(t, t')$ and this implies the claim. $\square$

Claim 3. *β is an ℓ_1 semi-metric*

Proof: Using the description of β, one can easily write this semi-metric as a linear combination of cut-metrics on the set V, and this implies the claim. $\square$

Claim 4. $\sum_{(u,v) \in E} \beta(u, v) c(u, v) = 1$

Proof: Again, for each set $A \subset K$ consider the total distances $\times$ capacity units that are allocated when we increase the distances of all edges crossing the cut that achieves $h_K(A)$ by $\frac{\mu(A)}{h_K(A)}$. We know that the total capacity crossing this cut is $h_K(A)$ and for each such edge, the distance on that edge (according to β) is incremented by $\frac{\mu(A)}{h_K(A)}$. So this implies that the total contribution of the set $A \subset K$ to the total distance $\times$ capacity units is $\mu(A)$ and if we sum over all A we get the desired claim, because μ is a probability distribution. $\square$

Theorem 13.
$$\nu \leq O\left(\frac{\log k}{\log \log k}\right)$$

Proof: Using Theorem 11, there exists a 0-extension $f : V \to K$ (such that $f(t) = t$ for all terminals t) and such that

$$\sum_{(u,v) \in E} c(u,v)\beta(f(u), f(v)) \leq O\left(\frac{\log k}{\log \log k}\right)$$

Then suppose P1 plays such a strategy f:

$$\begin{aligned}
&E_{A \leftarrow \mu}[N(f, A)] \\
&= \sum_{(u,v) \in E} \sum_{A} \frac{1_{(f(u),f(v)) \in \delta_K(A)} c(u,v)\mu(A)}{h_K(A)} \\
&= \sum_{(u,v) \in E} c(u,v) D_\mu(f(u), f(v)) \\
&\leq \sum_{(u,v) \in E} c(u,v)\beta(f(u), f(v)) \leq O\left(\frac{\log k}{\log \log k}\right)
\end{aligned}$$

$\square$

We can improve this result by noticing that the semi-metric D_μ corresponding to the cost function associated with the randomized strategy μ for P2 and the feasible solution β that we produced are both ℓ_1. In particular, let $G = (V, E)$ be an undirected, capacitated graph and let $K \subset V$ be a subset of terminals. Let η denote a bound on the maximum integrality gap of the semi-metric relaxation when the semi-metric D is ℓ_1 and the feasible solution to the linear program β is also ℓ_1.

Corollary 1. $\nu \leq \eta$

We can instead use Theorem 12 in the case in which G excludes $K_{r,r}$ as a minor, and this immediately implies:

Corollary 2. *For any capacitated graph $G = (V, E)$ that excludes $K_{r,r}$ as a minor,*

$$\nu \leq O(r^2).$$

We can immediately use any bound on the game value to demonstrate that good quality cut-sparsifiers exist:

Theorem 14. *For any capacitated graph $G = (V, E)$, for any set $K \subset V$ of $|K| = k$ terminals, there is an ν-quality cut-sparsifier $H = (K, E_H)$ and in fact such a graph can be realized as a convex combination of graphs G_f generated by 0-extensions f.*

Proof: We can again apply von Neumann's minmax theorem, and get that there exists a distribution γ on 0-extensions ($f : V \to K$ s.t. $f(t) = t$ for all $t \in K$) such that for all $A \subset K$: $E_{f \leftarrow \gamma}[N(f, A)] \leq \nu$.

For any 0-extension f, let G_f be the corresponding 0-extension of G generated by f, and let $h_f : 2^K \to \mathbb{R}^+$ be the cut function defined on this graph. Further, let $H = \sum_{f \in supp(\gamma)} \gamma(f) G_f$. Then for any $A \subset K$:

$$h_K(A) \leq h_f(A) \text{ and } h_K(A) \leq h'(A) = \sum_{f \in supp(\gamma)} \gamma(f) h_f(A)$$

Also because $E_{f \leftarrow \gamma}[N(f, A)] \leq \nu$:

$$\sum_{(u,v) \in E} \sum_{f \in supp(\gamma)} \gamma(f) \frac{1_{(f(u), f(v)) \in \delta_K(A)} c(u, v)}{h_K(A)} = \frac{1}{h_K(A)} \sum_{f \in supp(\gamma)} \gamma(f) h_f(A) = \frac{h'(A)}{h_K(A)}$$

and so for any $A \subset K$: $h'(A) \leq \nu h_K(A)$. □

In particular, in general there are $O(\frac{\log k}{\log \log k})$-quality cut-sparsifiers. If G excludes $K_{r,r}$ as a minor, then there is an $O(r^2)$-quality cut-sparsifier. And if η is an upper bound on the maximum integrality gap of the 0-extension LP when both the semi-metric D and the feasible solution β are required to be ℓ_1, then there is an η-quality cut-sparsifier.

3.1.4 Quality-Revealing Linear Program

Here we give a linear program whose integrality gap is exactly the quality of the best cut-sparsifier that can be achieved through 0-extension functions. This result appears in [20].

Definition 9. *Let $\alpha_0^c(G, K)$ be the infimum quality of any cut-sparsifier $H = \sum_f \gamma(f) G_f$ and γ is a distribution over 0-extension functions.*

We will call $\alpha_0^c(G, K)$ the contraction quality of G, K.

Definition 10. *Let Δ_U denote the cut-metric in which $\Delta_U(u, v) = 1_{|U \cap \{u,v\}|=1}$.*

Suppose we are given a semi-metric D as an input to a 0-extension problem that is also ℓ_1. Then we can write a stronger (exponentially-sized) linear program whose integrality gap is no worse than the semi-metric relaxation. The advantage is that for this linear program, the maximum integrality gap over any ℓ_1 semi-metric D will be exactly the contraction quality (of G, K).

$$\begin{array}{ll} \min & \sum_U \delta(U) h(U) \\ \text{s.t.} & \\ & \forall_{t,t' \in K} \sum_U \delta(U) \Delta_U(t, t') = D(t, t'). \end{array}$$

We will call this the cut-cut relaxation.

Definition 11. *Let $\eta(G, K)$ be the maximum integrality gap of the cut-cut relaxation over all ℓ_1 semi-metrics (for a fixed G, K).*

We note that in Corollary 16 we were able to demonstrate that $\eta(G, K)$ is an upper bounds for the game value of the extension-cut game, and hence an $\eta(G, K)$-quality cut-sparsifier $H = \sum_f \gamma(f) G_f$ exists (which can be written as a convex combination of 0-extension graphs G_f).

Theorem 15. $\alpha_0^c(G, K) = \eta(G, K)$

Proof: Suppose γ is a distribution on 0-extensions such that $H = \sum_f \gamma(f) G_f$ is a α-quality cut sparsifier. Given any ℓ_1 semi-metric D on K, we can solve the cut-cut relaxation given above. Let R be the optimal value on D. Notice that cut $(U, V-U)$ that is assigned positive weight in an optimal solution must be the minimum cut separating $U \cap K = A$ from $K - A = (V-U) \cap K$ in G. If not, we could replace this cut $(U, V-U)$ with the minimum cut separating A from $K-A$ without affecting the feasibility and simultaneously reducing the cost of the solution. So for all U for which $\delta(U) > 0$, $h(U) = h_K(U \cap K)$.

Consider then the cost of the semi-metric D against the cut-sparsifier H which is defined to be $\sum_{(a,b)} c_H(a,b) D(a,b) = \sum_f \gamma(f) \sum_{(a,b)} c_f(a,b) D(a,b)$ which is just the average cost of D against G_f where f is sampled from the distribution γ. The cut-cut relaxation gives a decomposition of D into a weighted sum of cut-metrics - i.e. $D(a,b) = \sum_U \delta(U) \Delta_U(a,b)$. Also, the cost of D against H is linear in D so this implies that

$$\sum_{(a,b)} c_H(a,b) D(a,b) = \sum_{(a,b)} \sum_U c_H(a,b) \delta(U) \Delta_U(a,b) = \sum_{(a,b)} c_H(a,b) \delta(U) h'(U \cap K)$$

In the last line, we use $\sum_{(a,b)} c_H(a,b) \Delta_U(a,b) = h'(U \cap K)$. Then

$$\sum_{(a,b)} c_H(a,b) D(a,b) \leq \sum_U \delta(U) \alpha h_K(U \cap K) = \alpha R$$

In the inequality, we have used the fact that H is an α-quality cut-sparsifier, and in the last line we have used that $\delta(U) > 0$ implies that $h(U) = h_K(U \cap K)$. So the average cost of D against G_f where f is sampled from γ is at most α times R, so there must be some f such that the cost against D is at most α times R. Hence α is an upper bound on the integrality gap of the cut-cut relaxation for any D that is ℓ_1. $\square$

We will use this result in Section 5.1.1 to establish a (tight) lower bound on the quality of the best cut-sparsifier that can be achieved through contractions for a particular example. We will then give a cut-sparsifier (not based on contractions) that has asymptotically between quality - thus demonstrating an asymptotic separation (for a particular family of graphs) between the quality of the best cut-sparsifier and the quality of the best cut-sparsifier that can be achieved through a distribution on contractions.

3.2 Flow Sparsifiers

In Section 3.1, we proved that $O(\frac{\log k}{\log \log k})$-quality cut-sparsifiers exist. Using the celebrated approximate min-cut max-flow theorem for multicommodity flows due to Linial, London and Rabinovich [46], we observe that scaling up the weights in such in a good cut-sparsifier by an appropriate factor yields a good flow-sparsifier. In fact, an $O(\frac{\log k}{\log \log k})$-quality cut-sparsifier can be easily scaled up to obtain a poly-logarithmic (in k) quality flow-sparsifier.

However, we can do better. We prove that $O(\frac{\log k}{\log \log k})$-quality flow-sparsifiers exist and furthermore, if G excludes $K_{r,r}$ as a minor, then $O(r^2)$-quality flow-sparsifiers exist. We also prove that flow-sparsfication is no easier than cut-sparsification and in fact any α-quality flow-sparsifier is also an α-quality cut-sparsifier. Hence the results in this section, while technically more involved, subsume those in Section 3.1.

The additional benefit of working with flow-sparsifiers as opposed to cut-sparsifiers is that if we are given a graph $H = (K, E_H)$ which is guaranteed to be a flow-sparsifier (for example, this graph may be given to us as a distribution on 0-extension graphs), the problem of computing the quality of H is easy. The problem of determining the quality of H can be formulated as a single minimum congestion routing problem. However, if we are given a graph $H = (K, E_H)$ and promised that this graph is a cut-sparsifier (or even a flow-sparsifier), determining the quality of H as a cut-sparsifier is an instance of the generalized sparsest cut problem. Our algorithms for efficiently constructing good quality flow-sparsifiers (and cut-sparsifiers, for that matter) will be based on this property.

Additionally, this property allows us to (in an amortized sense) compute approximate minimum congestion routings for a sequence of demand vectors $\vec{f}$ in G in time polynomial in the number of terminals, and independent of the size of the original graph. The approximation factor will be exactly the quality of a good flow-sparsifier and the preprocessing time will be just the time to compute a good flow-sparsifier, and the time to compute a single minimum congestion routing in G.

The results in this section are based on [44].

3.2.1 A Hardest Flow

Suppose we are given a graph $H = (K, E_H)$ that is a flow-sparsifier. The quality of H as a flow-sparsifier is the worst case ratio over all demand vectors $\vec{f}$ with endpoints in K of the congestion of the minimum congestion routing in G and H respectively. We can clearly restrict attention to only demand vectors $\vec{f}$ that are routable with congestion at most one in H - we will call such a demand vector "routable" in H. Given that H is a flow-sparsifier, there is a natural notion of a "hardest" flow (routable in H) to route in G. Hence we can express the quality of H as a minimum congestion routing problem.

Definition 12. *Let $H = (K, E_H)$ be a capacitated graph. Then let $\vec{H} \in \mathbb{R}^{\binom{k}{2}}$ be the demand vector in which each coordinate (which corresponds to a pair $a, b \in K$) is set to $c_H(a, b)$ - i.e. the capacity of the edge $(a, b) \in E_H$ (if it exists, and is zero if not).*

Lemma 1. *If $H = (K, E_H)$ is a flow-sparsifier for G and K, then the quality of H as a flow-sparsifier is $cong_G(\vec{H})$.*

Proof: $cong_G(\vec{H})$ is clearly a lower bound for the quality of H as a flow-sparsifier, because the flow $\vec{H}$ is feasible in H by saturating all edges. To prove the reverse inequality, given any flow $\vec{f}$ routed in H, we can compose this routing of $\vec{f}$ with the embedding of H into G. If the routing of $\vec{f}$ has congestion at most 1 in H, then composing the routing with the embedding of H into G will result in congestion at most $cong_G(\vec{H})$ in G:

In particular, consider the flow $\vec{f}$ in H, and consider the path decomposition of this flow. If $\vec{f}$ assigns δ units of flow to a path $P_{a,b}$ in H, then construct a set of paths in G that in total carry δ units of flow as follows: Let $P_{a,b} = (a, p_1), (p_1, p_2), ...(p_l, b)$. Let $p_0 = a$ and $p_{l+1} = b$. Then consider an edge (p_i, p_{i+1}) contained in this path and suppose that $c_H(p_i, p_{i+1})$ is α in H. Then for each flow path P connecting p_i to p_{i+1} in the low-congestion routing of H in G, add the same path and multiply the weight by $\frac{\delta}{\alpha}$. The union of these flow paths sends δ units of flow from a to b in G. If we consider any edge in H, every flow path in $\vec{f}$ that traverses this edge uses exactly the paths in G to which (a, b) is mapped in the embedding of H into G. The total flow through any edge (a, b) in H is at most $c_H(a, b)$ because $\vec{f}$ is feasible in H, and so we have not scaled up the amount of flow transported on each path in an optimal routing the (demand) graph H into

G at all. Consequently the congestion of this (sub-optimal) routing of $\vec{f}$ in G is at most $cong_G(\vec{H})$. □

In fact, this proof reveals an important property of a good quality flow-sparsifier: Given a good quality flow-sparsifier H, we can compute an embedding of H into G of congestion α once, as a preprocessing step. We can use this routing so that given any subsequent demand vector $\vec{f}$, we can compute a near optimal routing of $\vec{f}$ in G without using any additional information about G except knowledge of H and the low-congestion routing of H in G.

Corollary 3. *Let H be a flow-sparsifier of quality α. We can compute an optimal routing of $\vec{H}$ in G as a preprocessing step. For each demand vector $\vec{f}$, we can then compute a routing of $\vec{f}$ in G that has congestion at most $\alpha cong_G(\vec{f})$ by running a minimum congestion routing algorithm on H.*

The running time of the best known $(1+\epsilon)$-approximate minimum congestion routing runs in time $\tilde{O}(rmn)$ where $m = |E|$ and $n = |V|$ and r is the number of demand pairs. Hence, if we run a minimum congestion routing algorithm on H, the algorithm runs in time polynomial in the number of terminals and has no dependence on the size of the original graph G. So given a sequence of routing problems that we are asked to solve in G (all of which have endpoints only in the set K), we an compute a near-optimal routing for each problem in time polynomial in $|K|$ and independent of the size of G!

3.2.2 Relations to Cut Sparsifiers

Next, we prove that flow-sparsification is no-easier than cut-sparsification:

Lemma 2. *If $H = (K, E_H)$ is an α-quality flow-sparsifier for G and K, then H is also an α-quality cut-sparsifier (for G and K).*

Proof: Let $H = (K, E_H)$ be an α-quality flow-sparsifier, and let h' be the cut function of H. The proof of this lemma will be composed of two claims. The first establishes that H is a cut-sparsifier, and the second establishes that the quality of H as a cut-sparsifier is at most α:

Claim 5. *For all $A \subset K$: $h_K(A) \leq h'(A)$*

Proof: Suppose for some $A \subset K$: $h_K(A) > h'(A)$. Then the min-cut max-flow theorem implies that there is a flow $\vec{r}$ feasible in G such that the total demand crossing the (demand) cut $(A, K - A)$ is exactly $h_K(A)$. But this flow $\vec{r}$ cannot be feasible in H because the cut $(A, K - A)$ has capacity $h'(A)$ in H which is strictly smaller than the demand crossing the cut. So this implies $cong_G(\vec{r}) \leq 1 < cong_H(\vec{r})$ which is a contradiction. □

Claim 6. *For all $A \subset K$: $h'(A) \leq \alpha h_K(A)$*

Proof: Consider the flow $\vec{H}$. This flow is feasible in G' - i.e. $cong_H(\vec{H}) = 1$. Now suppose that there is a cut $A \subset K$ such that $h'(A) > \alpha h_K(A)$. Choose the set $U \subset V, U \cap K = A$ such that $h(U) = h_K(A)$. The capacity crossing the cut $(U, V - U)$ is $h_K(A)$ but the total demand crossing the cut is $h'(A)$ so

$$cong_G(\vec{H}) \geq \frac{h'(A)}{h_K(A)} > \alpha$$

and $cong_G(\vec{H}) > \alpha cong_H(\vec{H})$ which is a contradiction. □

□

We also note that not only is a graph G_f obtained from a 0-extension function a cut-sparsifier, but it is also a flow-sparsifier as well.

Claim 7. *For any 0-extension f, and for any demand vector $\vec{r} \in \mathbb{R}^{\binom{k}{2}}$, $cong_{G_f}(\vec{r}) \leq cong_G(\vec{r})$*

Proof: We can equivalently think of a contraction operation as setting the capacity of an edge to be infinite. Then a comparison argument implies the claim: Suppose we set the capacity of an edge $e = (u, v)$ to be infinite in the graph G to obtain a graph G'. Then any demand vector $\vec{r}$ that can be routed with congestion at most C in G can also be routed with congestion at most C in G' - the same routing for $\vec{r}$ that is used in G has congestion at most C in G' as well. □

3.2.3 A Larger Zero-Sum Game

To prove the existence of good flow-sparsifiers, we again use an exponentially-sized zero-sum game. However, here we will need to add another layer of indirection based on duality in order to prove this result.

We first associate a convex subset of $\mathbb{R}^{\binom{k}{2}}$ with G, K which characterizes the value of the minimum congestion routing of any demand vector $\vec{f}$ in G.

Definition 13.

$$P_G = \{\vec{f} \in \mathbb{R}^{\binom{k}{2}} | cong_G(\vec{f}) \leq 1\}$$

We will call P_G the unit congestion polytope of G. Note that $cong_G(\vec{f})$ can be computed as the minimum α so that $\alpha \vec{f}$ is contained in P_G. Similarly, we will define S_G to be the boundary of P_G:

Definition 14.

$$S_G = \{\vec{f} \in \mathbb{R}^{\binom{k}{2}} | cong_G(\vec{f}) = 1\}$$

The zero-sum game that we introduce here, to prove the existence of (better) flow-sparsifiers will actually be based on the dual the the maximum concurrent flow problem. Hence, another interesting convex set is the polar dual to P_G, which we will refer to as D_G.

Definition 15. *We will call a metric space d on K realizable in G if there is some extension of d to V so that $\sum_{(u,v) \in E} c(u, v)d(u, v) \leq 1$.*

The set D_G of realizable metric spaces is clearly convex. We will encounter a minor technicality, that if we were to define a zero-sum game using D_G as the strategy space for one of the players, then D_G is uncountably infinite (but still compact). To avoid this technical annoyance, we can simply choose the set of vertices of D_G to be a strategy space and this set will then be finite. Not only is D_G convex, but we can explicitly write it as the projection of a higher dimensional polytope, and hence D_G is also a polytope and has a finite (albeit exponentially many) number of vertices.

Claim 8. *The number of vertices in D_G is finite.*

Proof: To prove this claim, consider the polytope P_G. This polytope can be realized as the projection of a higher dimensional polytope (that actually finds the realization of each multicommodity flow, and uses flow-conservation constraints as opposed to an explicit decomposition into flow paths). This higher dimensional polytope has a (polynomial in n, k) number of constraints, and hence a finite number of vertices and this implies that the projected polytope P_G also has a finite number of vertices. Each distinct facet of P_G contains a distinct subset of the vertices of P_G, and hence the number of facets of P_G is also finite. The facets of P_G are exactly the polar duals to the vertices of D_G, and this implies the claim. $\square$

Given an undirected, capacitated graph $G = (V, E)$ and a set $K \subset V$ of terminals, a (metric) extension player (P1) and a quality player (P2) play the following zero-sum game that we will refer to as the extension-quality game. We will add an additional generalization that will prove useful later:

Definition 16. *Let $\mathcal{H}$ be the set of flow-sparsifiers for G and K - i.e.*

$$\mathcal{H} = \{H | H \textit{ is a flow-sparsifier for } G, K\}$$

Let $\mathcal{H}' \subset \mathcal{H}$ be an arbitrary, finite subset. For example, we can consider $\mathcal{H}' = \{G_f\}_f$. In this case, $\mathcal{H}'$ is the set of all 0-extensions.

Suppose that we want to impose additional structure on a flow-sparsifier. For example, in Section 4.2 we will want a flow-sparsifier that not only has good quality but also can be realized as a convex combination of "contraction-based" trees. In fact, we can re-use the much of the machinery in this section for a different choice of $\mathcal{H}'$ and prove that good (restricted) flow-sparsifiers exist provided $\mathcal{H}'$ meets a technical condition. On a first reading, the reader is advised to think of $\mathcal{H}'$ as the set of all 0-extensions.

The extension player (P1) chooses a $H \in \mathcal{H}'$
The quality player (P2) chooses a vertex $d_{\vec{r}}$ of D_G

We write $d_{\vec{r}}$ for a vertex of D_G because we will use $\vec{r}$ to denote a vertex of P_G that the half-space perpendicular to $d_{\vec{r}}$ contains. Note that this is well-defined because D_G is the polar dual to P_G and hence if $d_{\vec{r}} \in D_G$ is not perpendicular to some vertex of P_G, then $d_{\vec{r}}$ is not a vertex in D_G. Equivalently, $\vec{r}$ is a demand vector that is feasible in G and for which $\sum_{(a,b)} \vec{r}_{a,b} d_{\vec{r}}(a,b) = 1$ and if there is no such tight demand vector, we could increase coordinates in $d_{\vec{r}}$ and still obtain a vector that using strong duality must be contained in D_G.

Given a strategy H for P1 (and suppose that the capacity function of this graph is c_H) and a strategy $d_{\vec{r}}$ for P2, P2 wins

$$M(H, d_{\vec{r}}) = \sum_{a,b \in K} d_{\vec{r}}(a,b) c_H(a,b)$$

The advantage of restricting the congestion player's strategies to be vertices of D_G is that the joint strategy space of the game is finite, so we can apply the standard minmax theorem for finite strategy space games, rather than more general minmax theorems for arbitrary convex spaces that are not necessarily characterized as the convex hull of a finite number of points.

Definition 17. *Let ν denote the game value of the extension-quality game*

Given a metric space d (which we can assume after scaling is in realizable), consider the optimization problem of finding a graph $H \in \mathcal{H}'$ that minimizes $\sum_{a,b \in K} d(a,b) c_H(a,b)$. We will call this the $\mathcal{H}'$-extension problem. If we consider the semi-metric relaxation given for the 0-extension problem, this is also a relaxation for the $\mathcal{H}'$-problem. Hence, the semi-metric relaxation has a natural notion of an integrality gap with respect to the set $\mathcal{H}'$.

Definition 18. *We will call the following quantity the integrality gap of $\mathcal{H}'$ (on G and K):*

$$\sup_{d \in D_G} \inf_{H \in \mathcal{H}'} \sum_{a,b \in K} d(a,b) c_H(a,b)$$

Note that the integrality gap of $\mathcal{H}'$ is defined relative to G and K.

Claim 9. *The integrality gap of $\mathcal{H}' = \{G_f\}_f$ is at most $O(\frac{\log k}{\log \log k})$*

Proof: If $d \in D_G$, the theorem due to Fakcharoenphol, Harrelson, Rao and Talwar [26] implies that there is a 0-extension f such that

$$\sum_{(u,v)\in E} c(u,v)d(f(u),f(v)) \leq O(\frac{\log k}{\log\log k})$$

Then consider the graph G_f and let c_f be the corresponding capacity function.

$$\sum_{a,b\in K} d(a,b)c_f(a,b) = \sum_{a,b\in K}\sum_{u,v\in E} d(f(u),f(v))c(u,v) \leq O(\frac{\log k}{\log\log k})$$

□

We can instead use Theorem 12 in the case in which G excludes $K_{r,r}$ as a minor, and this immediately implies:

Corollary 4. *If $G = (V,E)$ excludes $K_{r,r}$ as a minor, the integrality gap of $\mathcal{H}' = \{H | H = \sum_f \gamma(f)G_f\}$ is at most $O(r^2)$*

Lemma 3. *ν is at most the integrality gap of $\mathcal{H}'$*

Proof: Using von Neumann's minmax theorem, we can bound the game value by bounding the cost of P1's best response to any fixed, randomized strategy for P2. So consider any randomized strategy μ for P2. μ is just a probability distribution on vertices of D_G. We can define a semi-metric on V: $\delta(a,b) = \sum_{d_{\vec{r}}} \mu(d_{\vec{r}})d_{\vec{r}}(a,b)$. Note that

$$\begin{aligned}\sum_{(a,b)\in E} \delta(a,b)c(a,b) &= \sum_{(a,b)\in E}\sum_{d_{\vec{r}}} \mu(d_{\vec{r}})d_{\vec{r}}(a,b)c(a,b) \\ &= \sum_{d_{\vec{r}}} \mu(d_{\vec{r}}) \sum_{(a,b)\in E} d_{\vec{r}}(a,b)c(a,b) \\ &= \sum_{d_{\vec{r}}} \mu(d_{\vec{r}}) = 1\end{aligned}$$

Suppose the integrality gap of $\mathcal{H}'$ is at most C. Then there exists an $H \in \mathcal{H}'$ such that

$$\sum_{a,b\in K} \delta(a,b)c_H(a,b) \leq C$$

Suppose that P1 plays such a strategy f:

$$\begin{aligned} E_{d_{\vec{r}} \leftarrow \mu}[M(H, d_{\vec{r}})] &= \sum_{d_{\vec{r}}} \mu(d_{\vec{r}}) \sum_{a,b \in K} d_{\vec{r}}(a,b) c_H(a,b) \\ &= \sum_{a,b \in K} \delta(a,b) c_H(a,b) \leq C \end{aligned}$$

□

We can immediately use any bound on the game value to demonstrate that better quality flow-sparsifiers exist:

Theorem 16. *For any capacitated graph $G = (V, E)$, for any set $K \subset V$ of $|K| = k$ terminals, there is an ν-quality flow-sparsifier $H = (K, E_H)$ and in fact such a graph can be realized as a convex combination of graphs G_f generated by 0-extensions f.*

Proof: We can again apply von Neumann's minmax theorem, and get that there exists a distribution γ on 0-extensions such that $E_{f \leftarrow \gamma}[M(f, d_{\vec{r}})] = \nu$ for all $d_{\vec{r}} \in D_G$. Then let

$$H = \sum_{f \in supp(\gamma)} \gamma(f) G_f$$

Let $d_{\vec{r}} \in D_G$ be arbitrary. Then

$$\begin{aligned} E_{f \leftarrow \gamma}[M(f, d_{\vec{r}})] &= \sum_{f} \gamma(f) \sum_{(a,b) \in E} d_{\vec{r}}(f(a), f(b)) c(a,b) \\ &= \sum_{f} \gamma(f) \sum_{u,v \in K} d_{\vec{r}}(u,v) c_f(a,b) \\ &= \sum_{u,v \in K} d_{\vec{r}}(u,v) \sum_{f} \gamma(f) c_f(a,b) \\ &= \sum_{u,v \in K} d_{\vec{r}}(u,v) c'(u,v) \end{aligned}$$

So for all $d_{\vec{r}} \in D_G$, we have that $\sum_{u,v \in K} d_{\vec{r}}(u,v) c'(u,v) \leq \nu$ And using strong duality (and the dual for maximum concurrent flow given in Section 2):

$$cong_G(\vec{H}) = \sup_{d_{\vec{r}} \in D_G} \sum_{u,v \in K} d_{\vec{r}}(u,v) c'(u,v) \leq \nu$$

$\square$

As we will see, flow-sparsifiers that arise as a convex combination of 0-extension graphs preserve more than just multicommodity flow rates. Roughly, any type of ”rate” that both can only increase when increasing the capacity of an edge, and is closed under composition with flow-like re-routings will also be preserved. An important type of rate that falls into this category, but is still poorly understood is the network coding rate.

3.2.4 Multiplicative Weights

Here we apply the results in [54] to give an efficient construction for a good quality flow-sparsifier. See also [4] for a more general exposition of the results in [54]. The highlight of this approach is that any efficient rounding algorithm for the semi-metric relaxation that achieves an approximation ratio of C (relative to the value of the relaxation) will immediately give rise to an efficient construction for a $2C$-quality flow-sparsifier. This approach has the benefit that the running time of a rounding algorithm is typically linear in the size of G (and has a polynomial dependence on the number of terminals), and we will be able to give fast algorithms for computing a good quality flow-sparsifier that run in roughly quadratic (in $|E|$) time.

This is interesting because the running time of computing a good quality flow-sparsifier matches (within logarithmic factors) the running time of solving just a single minimum congestion routing problem. Yet a good flow-sparsifier will help us solve a sequence of routing problems quickly. We can compute a good flow-sparsifier just *once* and for any routing problem we are asked to solve, we can compute a near-optimal solution by running a minimum congestion routing algorithm on our flow-sparsifier, as opposed to running directly on the original network.

In Section 4.1.2 we present another algorithm for computing a good quality flow-sparsifier. The algorithm presented in this section is faster. However, the algorithm presented in Section 4.1.2 will be able to achieve a stronger quality guarantee. Here, all we are able to construct is a flow-sparsifier whose quality matches the worst-case guarantee of an efficient rounding algorithm. The algorithm in Section 4.1.2 will be able to construct a flow-sparsifier whose quality is at least as good as the quality of the best flow-sparsifier that can be achieved through contractions. Conceivably, this could be much stronger. For example, the maximum integrality gap of the semi-metric relaxation is only known to be bounded between $O(\frac{\log k}{\log \log k})$ and $\Omega(\sqrt{\log k})$ and if this latter bound is correct, then the algorithm in Section 4.1.2 already constructs an $O(\sqrt{\log k})$-quality flow-sparsifier.

The zero-sum game given in Section 3.2.3 used to prove the existence of a good quality flow-sparsifier can be immediately used, in conjunction with the proof in [54] to give a fast algorithm for computing a good quality flow-sparsifier. We will not repeat the proof here, but rather we will just note the notational differences and state the result in the context of vertex sparsification.

Räcke considers the problem of finding a good hierarchical decomposition for a graph. To this end, for each ”decomposition tree” T_i he defines a notion of the load $load_{T_i}(e)$ and relative load $rload_{T_i}(e) = \frac{load_{T_i}(e)}{c(e)}$ on each edge. His goal is to find a packing that does not have too large a relative load on any edge:

$$\begin{array}{rl} \max & \sum_i \lambda_i \\ \text{s.t.} & \\ \forall e \in E & \sum_i \lambda_i rload_{T_i}(e) \leq \beta. \end{array}$$

The goal in [54] is to find a choice of $\vec{\lambda}$ so that $\sum_i \lambda_i = 1$ and yet $\beta = O(\log n)$. He then uses such a distribution $\vec{\lambda}$ to construct an $O(\log n)$-competitive oblivious routing scheme. To prove this result, Räcke considers a smooth potential function. Let $lmax(\vec{x}) = \ln\left(\sum_e e^{x_e}\right) \geq \max_e x_e$. Räcke considers the stronger goal of finding a $\vec{\lambda}$ such that $lmax(M\vec{\lambda}) \leq \beta$ where M is the constraint matrix - namely the value

$$(M\vec{\lambda})_e = \sum_i \lambda_i rload_{T_i}(e)$$

is the relative load on edge e.

Here, instead, our goal is to construct a good quality flow-sparsifier. We replace the notion of a "decomposition tree" with a choice of a flow sparsifier $H \in \mathcal{H}'$ (as in the game in Section 3.2.3) and a set of paths in G - one connecting each pair of terminals (a, b). Let $\mathcal{R}$ denote the possible choices for these paths - i.e. $\mathcal{R}$ denotes the set of all sets of $\binom{k}{2}$ simple paths in G so that each path connects a distinct pair of terminals. Let $R \in \mathcal{R}$ be a particular choice for these $\binom{k}{2}$ paths. We can consider routing H as a flow in G, using the paths in R. Let $R_{a,b}$ be the path in R connecting a and b in G. For each edge (a, b) in H (of capacity $c_H(a, b)$) we can send $c_H(a, b)$ units of flow along the corresponding path $R_{a,b}$. If we do this for all pairs of terminals, we have a routing of $\vec{H}$ in G. Moreover, we can define the load of $H \times R$ on an edge e as the total amount of flow traversing the edge e:

$$load_{H\times R}(e) = \sum_{a,b \in K} c_H(a, b) 1_{e \in R_{a,b}}$$

and similarly the relative load $rload_{H\times R}(e) = \frac{load_{H\times R}(e)}{c(e)}$. Then the goal of constructing a β-quality flow-sparsifier is equivalent to finding a distribution $\vec{\lambda}$ so that for all $e \in E$, $\sum_{H\times R} \lambda_{H\times R} rload_{H\times R}(e) \leq \beta$. As in [54], we can consider a smooth potential function and ask for a $\vec{\lambda}$ that satisfies a stronger guarantee:

$$\begin{aligned} lmax(M\vec{\lambda}) &\leq \beta \\ \textstyle\sum_{H\times R} \lambda_{H\times R} &\geq 1 \\ \lambda_{H\times R} &\geq 0 \qquad \text{for all } H, R \end{aligned}$$

Räcke reduces the problem of finding a solution to this convex program to the problem of always being able to find an index $H \times R$ so that the partial of $lmax(M\vec{\lambda})$ with respect to $\lambda_{H\times R}$ is at most $\beta/2$. The intuition here is that if we can always find such an index, throughout the course of the algorithm we will be able to maintain the invariant that the potential function $lmax(M\vec{\lambda})$ is at most, say, a β-factor larger than the ℓ_1-norm of $\vec{\lambda}$.

For any $H \times R$, the partial derivative of $lmax(M\vec{\lambda})$ with respect to $\lambda_{H\times R}$ is

$$\frac{\partial lmax(M\vec{\lambda})}{\partial \lambda_{H\times R}} = \sum_e load_{H\times R}(e) \frac{e^{(M\vec{\lambda})_e}}{c(e) e^{(M\vec{\lambda})_e}}$$

Set $d(e) = \frac{e^{(M\vec{\lambda})_e}}{c(e)e^{(M\vec{\lambda})_e}}$ and let D be the resulting shortest path metric on G. Then we have that $\sum_{u,v} c(u,v)D(u,v) \leq 1$.

Suppose we are given access to a rounding algorithm that always returns an element $H \in \mathcal{H}'$ whose cost is at most a C factor larger than the cost of the semi-metric relaxation. Hence, given the semi-metric D the algorithm returns an H for which

$$\sum_{a,b \in K} c_H(a,b)D(a,b) \leq C$$

For each pair of terminals $a, b \in K$, let $R_{a,b}$ be the shortest path according to the metric D between a and b in G (and break ties arbitrarily). Let R be the set of these shortest paths, one for each pair of terminals. Then

$$\sum_e load_{H \times R}(e) \frac{e^{(M\vec{\lambda})_e}}{c(e)e^{(M\vec{\lambda})_e}} = \sum_{a,b \in K} c_H(a,b) \sum_{e \in R_{a,b}} D(e) = \sum_{a,b \in K} c_H(a,b)D(a,b) \leq C$$

Räcke also proves that the number of iterations (of repeatedly finding some $H \times R$ for which the partial is small, and increasing the corresponding index in $\vec{\lambda}$) is at most $O(|E| \log n)$ where $|V| = n$. Hence we obtain:

Theorem 17. *Let $\mathcal{H}' \subset \mathcal{H}$ be any finite set of flow-sparsifiers (for G and K), and suppose that we are given a rounding algorithm ALG that achieves an approximation ratio C relative to the semi-metric relaxation. Let T be the running time of this algorithm. Then there is an algorithm to construct a $2C$-quality flow-sparsifier that can be realized as a convex combination of flow-sparsifiers in $\mathcal{H}'$. The running time of this algorithm is $O(k|E|^2 \log n + |E|T \log n)$.*

Hence, given a *fast* rounding algorithm for the semi-metric relaxation, we can quickly compute a good quality flow-sparsifier. The rounding algorithm due to Fakcharoenphol, Harrelson, Rao and Talwar [26] (Theorem 11) runs in time $O(k|E|)$ and hence:

Corollary 5. *There is an algorithm to compute an $O(\frac{\log k}{\log \log k})$-quality flow-sparsifier that runs in time $O(k|E|^2 \log n)$.*

Also, the rounding algorithm due to Calinescu, Karloff and Rabani [18] (using an im-

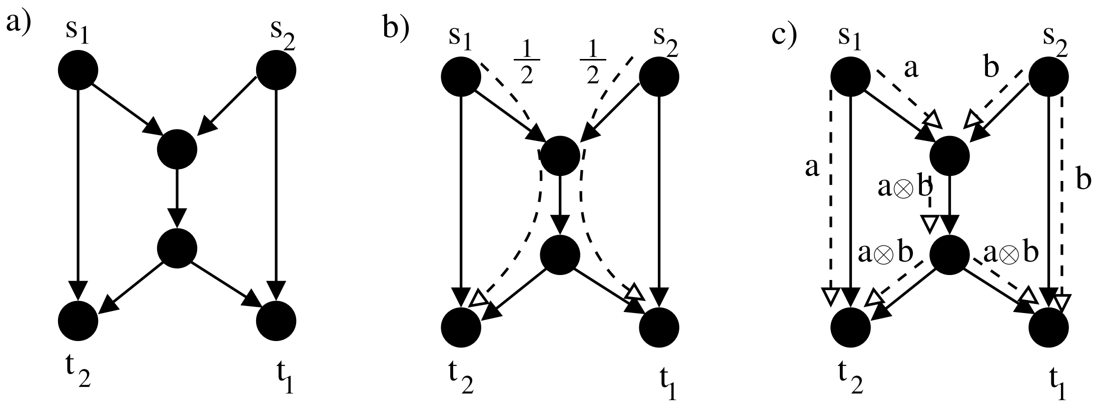

Figure 3-4: a. Communication problem on a directed graph b. Multicommodity solution c. Network coding solution

proved padded decomposition procedure due to Fakcharoenphol and Talwar [28]) runs in time $O(r^2|E| + k|E|)$ (assuming G excludes $K_{r,r}$ as a minor). Hence

Corollary 6. *Suppose G excludes $K_{r,r}$ as a minor. Then there is an algorithm to compute an $O(r^2)$-quality flow-sparsifier that runs in time $O(k|E|^2 \log n + r^2|E|^2 \log n)$.*

3.2.5 Network Coding

Here we demonstrate that an α-quality flow-sparsifier that is generated through a distribution on contractions also preserves network coding rates within an α-factor as well.

Suppose we are given the communication problem in Figure 3-$4a$ - s_1 wants to send a message to t_1 and s_2 wants to send a message to t_2. We can phrase this communication problem as a multicommodity problem - i.e as a maximum concurrent flow problem, in which case the optimal rate is $\frac{1}{2}$ as in Figure 3-$4b$. However, if we allow messages to be XORed, then the optimal rate can be made to be 1 as in Figure 3-$4c$. This later solution to the communication problem is a network coding solution. This example is due to Ahlswede et al [3].

We do not define the precise notion of a network coding problem or solution here because we will not use these definitions apart from noticing that a flow-sparsifier that is generated as a convex combination of 0-extension graphs preserves network coding rates (within the same factor) in addition to multicommodity flow rates. See [36] for a precise definition of network coding rate.

Definition 19. *Given a demand vector $\vec{r} \in \mathbb{R}^{\binom{k}{2}}$, suppose that R is the network coding rate for the network coding problem. We will call $cong_G^n(\vec{r}) = \frac{1}{R}$ the network congestion.*

We observe that $cong_G^n(\vec{r}) \leq cong_G(\vec{r})$ because from any maximum concurrent flow solution we can construct a network coding solution that is at least as good. Also, from the above example we can have a strict inequality in the case of a directed graph - i.e. the network coding solution performs strictly better. However, it is conjectured [1], [36] that the network coding rate is always equal to the maximum concurrent flow rate for undirected graphs. Despite much work on this problem, researchers have only been able to improve upon the sparsest cut upper bound for the network coding rate in specific cases [1], [36], [37].

This conjecture has far-reaching implications. The results of Ahlswede et al [1] imply that even proving an $o(\log k)$ bound - i.e. $cong_G(\vec{r}) \leq o(\log k) cong_G^n(\vec{r})$ for all undirected graphs would imply the first super-linear lower bound for oblivious matrix transposition in the bit-probe model, and this would be the first super-linear lower bound in this powerful model for *any* problem.

In fact, any flow-sparsifier generated from a convex combination of 0-extensions will necessarily approximately preserve many types of "congestion" functions, which we formalize below as proper congestion functions.

Definition 20. *We will call a congestion function $cong_G^* : \mathbb{R}^{\binom{k}{2}} \rightarrow \mathbb{R}^+$ a proper congestion function if for all demand vectors $\vec{r}$*

- *increasing the capacity of an edge in G can only decrease $cong_G^*(\vec{r})$*
- $cong_G^*(\vec{r}) \leq cong_G(\vec{H}) cong_H^*(\vec{r})$

Note that the second point requires that the congestion function be "compose-able" with minimum congestion routings. For example, in the case of network codes, if there is a good network coding solution in H and H can embed well into G as a flow, then there is a good network coding solution in G as well.

Claim 10. *Network congestion is a proper congestion function.*

This follows immediately from the precise definition of network coding - again see [36] for a precise definition. Given any (proper) congestion function $cong^*_G$, we can define the notion of an α-quality vertex sparsifier as a graph $H = (K, E_H)$ for which for all $\vec{r} \in \mathbb{R}^{\binom{k}{2}}$:

$$cong^*_H(\vec{r}) \leq cong^*_G(\vec{r}) \leq \alpha cong^*_H(\vec{r})$$

Lemma 4. *Let $H = \sum_f \gamma(f) G_f$ be a flow-sparsifier of quality α that is generated from a convex combination of 0-extension functions. Then H is also an α-quality vertex sparsifier with respect to* any *proper congestion function.*

Proof: Using the first property of a proper congestion function, $cong^*_H(\vec{r}) \leq cong^*_G(\vec{r})$. Similarly, using the second property of a proper congestion function we have that for all demand vectors $\vec{r}$,

$$cong^*_G(\vec{r}) \leq cong_G(\vec{H}) cong^*_H(\vec{r})$$

and using Lemma 1 this implies the result. □

Later, we will demonstrate a separation - that requiring a vertex sparsifier to be generated as a convex combination of 0-extensions can make the problem asymptotically harder.

Chapter 4

Algorithms

4.1 Computing a Vertex Sparsifier

Here we give efficient algorithms for constructing flow-sparsifiers (and hence also cut-sparsifiers) that achieve the existential bounds given in Chapter 3. We give an algorithm that runs in time polynomial in n and k that can construct a flow-sparsifier whose quality matches the quality of the best flow-sparsifier that can be achieved through 0-extension functions for the given G and K. Thus this algorithm always constructs a flow-sparsifier of quality at most $O(\frac{\log k}{\log \log k})$. And if G excludes $K_{r,r}$ as a minor then this algorithm constructs a flow-sparsifier of quality at most $O(r^2)$. If there happens to be an even better quality flow-sparsifier (that can still be achieved through 0-extensions), the algorithm can always find a flow-sparsifier whose quality is at least as good– although the flow-sparsifier that is constructed is not necessarily realizable as a convex combination of 0-extensions. Our algorithm makes novel use of Earth-mover constraints. Note that in Section 3.2.4, we gave a faster algorithm that achieves the same worst-case guarantees, but does not necessarily find an even better flow-sparsifier if one happens to exist for the particular G and K.

4.1.1 A Geometric Interpretation

We can naturally represent any (candidate) vertex sparsifier using a polynomial number of variables. Any (candidate) graph $H = (K, E_H)$ is defined by $\binom{k}{2}$ non-negative variables

$$\vec{X} = \{x_{a,b}\}_{a,b \in K}$$

which each are assigned a value equal to the capacity of the corresponding edge in H (here we think of H as being a complete, capacitated graph - where edge weights can be zero).

Consider the predicates:

- H is a cut-sparsifier
- H is a flow-sparsifier
- H has quality at most α as a cut-sparsifier
- H has quality at most α as a flow-sparsifier

Each of these predicates can be written as a set of linear inequalities on the variables X. Consider the first and third predicate, which will be easier to write as a set of linear inequalities as these constraints only pertain to cuts. Consider the polytope:

$$\begin{array}{lll} \textbf{Type 1.c:} & h_K(A) \leq \sum_{(a,b) \in \delta_K(A)} x_{a,b} & \text{for all } A \subset K \\ \textbf{Type 2.c:} & \sum_{(a,b) \in \delta_K(A)} x_{a,b} \leq \alpha h_K(A) & \text{for all } A \subset K \\ & 0 \leq x_{a,b} & \text{for all } a, b \in K \end{array}$$

Claim 11. *H is a cut-sparsifier if and only if all* ***Type 1.c*** *inequalities are satisfied. Also H has quality at most α as a cut-sparsifier if and only if all* ***Type 2.c*** *inequalities are satisfied.*

Hence, the above polytope represents the set of all α-quality cut-sparsifiers for G, K. Apriori we do not know if this polytope is non-empty, but the results in Chapter 3 imply that this polytope is non-empty for $\alpha = O(\frac{\log k}{\log \log k})$. If G excludes $K_{r,r}$ as a minor, then this polytope is non-empty for $\alpha = O(r^2)$.

Notice that if we are content with an algorithm that runs in time polynomial in n but exponential in k, we can afford to explicitly write down each of **Type 1.c** and **Type 2.c** inequalities. Then we can run any polynomial time algorithm for solving a linear program to obtain not just a good quality cut-sparsifier, but also the best quality cut sparsifier for the given G and K.

However, an algorithm that runs in time exponential in k is insufficient for many applications - for many NP-hard graph partitioning problems, we can compute a good quality cut-sparsifier as a preprocessing step. We can run an off-the-shelf approximation algorithm directly on H as a proxy for running on G in order to save both in both time and the approximation guarantee (by replacing a dependence on the size of G with a slightly worse factor that only depends on the size of H). Yet if we require time exponential in k to compute a good quality cut-sparsifier, we can solve a graph partitioning problem involving the terminals directly by brute force in G simply by enumerating the minimum cut in G separating A from $K - A$ for all $A \subset K$.

Our goal will be to give an algorithm for computing a good quality vertex sparsifier that runs in time polynomial in both n and k. Our constructive results will be predicated on the existential result – once we are guaranteed that the polytope is non-empty, we can design a separation oracle for this polytope and use the Ellipsoid algorithm to find a point inside this polytope in polynomial time – even though there are exponentially many constraints.

Already, we see that if we were to run the Ellipsoid algorithm (assuming a polynomial time separation oracle for the **Type 1.c** and **Type 2.c** inequalities, we would obtain a cut-sparsifier of good quality but this cut-sparsifier would not necessarily be realizable as a convex combination of 0-extensions, even though we know that good cut-sparsifiers realizable through 0-extensions exist.

Using this observation, there is in fact an even tighter polytope contained inside the above polytope, which the existential results in Chapter 3 also imply is non-empty.

Definition 21. $Z = \{\vec{H} \in \mathbb{R}^{\binom{k}{2}} | \vec{H} = \sum_f \gamma(f)\vec{G}_f\}_\gamma$

Consider the polytope:

Type Z: $\vec{X} \in Z$

Type 2.c: $\sum_{(a,b)\in\delta_K(A)} x_{a,b} \le \alpha h_K(A)$ for all $A \subset K$

$0 \le x_{a,b}$ for all $a, b \in K$

This polytope is contained inside the polytope formed using **Type 1.c** and **Type 2.c** inequalities because any graph representable as a convex combination of 0-extensions is necessarily a cut-sparsifier too. However, even this tighter polytope is non-empty for $\alpha = O(\frac{\log k}{\log \log k})$ because the existential results in Chapter 3 not only state that $O(\frac{\log k}{\log \log k})$-quality cut-sparsifiers exist, but that furthermore these cut-sparsifiers are realizable as a convex combination of 0-extensions.

We note that the 0-extension problem [39] is NP-hard, and hence we cannot hope to optimize a linear function over the polytope Z in polynomial time assuming $P \neq NP$: we could encode an arbitrary metric D on K into a linear function so that the minimum value of this linear function on the domain Z is precisely the value of the minimum cost 0-extension for the 0-extension problem defined by G, K and D.

Next, we give equivalent sets of linear inequalities for each of the remaining predicates (the second and fourth predicate above). Recall that in Section 3.2.3, we defined the set P_G as the set of all demand vectors routable in G with congestion at most one. Let $V(P_G)$ be the vertices of this polytope, which using Claim 8 is finite.

Definition 22. $\Pi_G = \cup_{\vec{r}\in V(P_G)}\{$*all semi-metrics* d *s.t.* $\sum_{a,b\in K} \vec{r}_{a,b} d(a,b) = 1\}$

Recall that in Section 3.2.3, we also defined D_G as the set of semi-metrics that are realizable in G using total distance $\times$ capacity units at most one. Then consider the polytope:

Type 1.f: $\sum_{a,b\in K} x_{a,b} d(a,b) \ge 1$ for all $d \in \Pi_G$

Type 2.f: $\sum_{a,b\in K} x_{a,b} d(a,b) \le \alpha$ for all $\vec{d} \in D_G$

$0 \le x_{a,b}$ for all $a, b \in K$

Lemma 5. *H is a flow-sparsifier if and only if all **Type 1.f** inequalities are satisfied.*

Proof: Suppose there is some $d \in \Pi_G$ for which $\sum_{a,b\in K} x_{a,b} d(a,b) < 1$. Hence there is some $\vec{r} \in V(P_G)$ for which $\sum_{a,b\in K} \vec{r}_{a,b} d(a,b) = 1$, and so using weak duality the demand vector $\vec{r}$ is not routable with congestion one in H. However $\vec{r}$ is in P_G and hence is routable with congestion one in G, and so H is not a flow-sparsifier.

To prove the reverse direction, suppose that H is not a flow-sparsifier. Then there is some $\vec{r} \in P_G$ that is not routable with congestion one in H. If $\vec{r}$ is not on the boundary of P_G, we can scale up $\vec{r}$ so that it is - i.e. $\vec{r} \in S_G$, and this maintains the invariant that $\vec{r}$ is not routable in H with congestion one. We can apply strong duality, and there must be some semi-metric d such that the ratio

$$\frac{\sum_{a,b \in K} \vec{r}_{a,b} d(a,b)}{\sum_{a,b \in K} x_{a,b} d(a,b)} > 1$$

Since $\vec{r} \in S_G$, we can write $\vec{r}$ as a convex combination of points in $V(P_G)$. This implies that there is some $\vec{s} \in V(P_G)$ for which the ratio is also strictly larger than one:

$$\frac{\sum_{a,b \in K} \vec{s}_{a,b} d(a,b)}{\sum_{a,b \in K} x_{a,b} d(a,b)} > 1$$

So we can scale the semi-metric d to find a violated **Type 1.f** inequality. $\square$

Claim 12. *H has quality at most α as a flow-sparsifier if and only if all* ***Type 2.f*** *inequalities are satisfied.*

Proof: This follows immediately using strong duality and Lemma 1. $\square$

We will at times mix and match these various families of inequalities to create different polytopes – while all these polytopes that we create will be non-empty using the existential results on vertex sparsifiers, some of these families of inequalities will be easier to enforce than others. We have already seen hints of this - the set Z is NP-hard to optimize over, and actually it is even NP-hard to implement a separation oracle for the set of **Type 2.c** inequalities as these inequalities are equivalent to the generalized sparsest cut problem (where $\vec{X}$ are interpreted as demands to be routed in G).

4.1.2 Lifting (via an Earth-Mover Relaxation)

Here we give an algorithm that runs in time polynomial in n and k that returns a flow-sparsifier whose quality is at least as good as the quality of the best flow-sparsifier that can be achieved through a distribution on 0-extensions. The results in this section appear in [20].

Our algorithm makes novel use of Earth-mover constraints. Consider the polytope Z defined in Definition 21. Optimizing an arbitrary linear function over this polytope is NP-hard. Given this observation, it is doubtful that we can efficiently implement a separation oracle for the polytope:

$$\begin{array}{lll} \textbf{Type Z:} & \vec{X} \in Z & \\ \textbf{Type 2.f:} & \sum_{a,b \in K} x_{a,b} d(a,b) \leq \alpha & \text{for all } \vec{d} \in D_G \\ & 0 \leq x_{a,b} & \text{for all } a,b \in K \end{array}$$

Similarly, consider the polytope:

$$\begin{array}{lll} \textbf{Type 1.f:} & \sum_{a,b \in K} x_{a,b} d(a,b) \geq 1 & \text{for all } d \in \Pi_G \\ \textbf{Type 2.f:} & \sum_{a,b \in K} x_{a,b} d(a,b) \leq \alpha & \text{for all } \vec{d} \in D_G \\ & 0 \leq x_{a,b} & \text{for all } a,b \in K \end{array}$$

While we can implement a separation oracle for the **Type 2.f** inequalities using minimum congestion routing, we do not know of an efficient separation oracle for the **Type 1.f** inequalities. We will return to the question of designing an efficient separation oracle for the **Type 1.f** inequalities in Section 4.1.3 where we will give an efficient approximate separation oracle based on oblivious routing.

Our approach is to define a set $Z' \supset Z$ for which we *can* give an efficient separation oracle. Furthermore, any point in Z' will satisfy the **Type 1.f** inequalities. This set Z' is based on the Earth-mover relaxation for the 0-extension problem – see [21] and [5].

Definition 23. *Let* $Z' = \{\vec{H} | \textit{for all } a,b \in K, c_H(a,b) = \sum_{u,v \in V} c(u,v) p_{a,b}^{u,v}\}$ *where* p *is* *required to satisfy the conditions:*

$$\begin{array}{rcll} p^{u,v}_{a,b} & = & p^{v,u}_{a,b} & \textit{for all } u, v \in V, u \neq v, \textit{ and } a, b \in K \\ \sum_{b \in K} p^{u,v}_{a,b} & = & P^u_a & \textit{for all } u, v \in V, u \neq v, \textit{ and } a, b \in K \\ \sum_{a \in K} P^u_a & = & 1 & \textit{for all } u \in V \\ P^a_a & = & 1 & \textit{for all } a \in K \\ p^{u,v}_{a,b} & \geq & 0 & \textit{for all } u, v \in V, u \neq v, \textit{ and } a, b \in K \end{array}$$

We first note that Z' is a relaxation for Z:

Claim 13. $Z' \supset Z$

Proof: The vertices of Z are $\{\vec{G}_f\}_f$ where f ranges over all 0-extensions. Consider an arbitrary vertex $\vec{G}_f$. We can set $P^u_a = 1_{f(u)=a}$ and similarly we can set $p^{u,v}_{a,b} = 1_{f(u)=a} 1_{f(v)=b}$. These values satisfy the conditions in Definition 23. We can check that the equality $c_f(a, b) = \sum_{u,v \in V} c(u, v) p^{u,v}_{a,b}$ holds, where c_f is the capacity function of G_f. Hence $\vec{G}_f \in Z'$ and since this is true for each vertex of Z, we conclude that $Z' \supset Z$. □

The more surprising property of Z' that our algorithm will use is that the constraints of Z' enforce the **Type 1.f** inequalities.

Lemma 6. *Let $\vec{X} \in Z'$. Then $\vec{X}$ satisfies the **Type 1.f** inequalities.*

Proof: Consider an arbitrary demand vector $\vec{r}$ that can be routed with congestion at most one in G - i.e. $\vec{r} \in P_G$. Let δ' be any semi-metric on K. Then

$$\begin{aligned} \sum_{a,b \in K} \delta'(a, b) c_H(a, b) &= \sum_{a,b \in K} \delta'(a, b) \sum_{u,v \in V} c(u, v) p^{u,v}_{a,b} = \sum_{u,v \in V} c(u, v) \sum_{a,b \in K, a \neq b} p^{u,v}_{a,b} \delta'(a, b) \\ &\geq \sum_{u,v \in V} c(u, v) EMD_{\delta'}(p^u, p^v) \end{aligned}$$

where $EMD_{\delta'}(p^u, p^v)$ is the Earth-mover distance between the distributions p^u and p^v under the semi-metric δ' - i.e. the minimum cost under δ' to move the distribution p^u to p^v.

Let $\delta(u,v) = EMD_{\delta'}(p^u, p^v)$. Clearly δ is a semi-metric on V. Since $\vec{r}$ is routable in G with congestion at most one, using weak duality we obtain that

$$\sum_{u,v \in V} c(u,v)\delta(u,v) \geq \sum_{a,b \in K} \vec{r}_{a,b}\delta(a,b)$$

which is in fact true (for any flow $\vec{r} \in P_G$) for any semi-metric δ on V. We observe that $\delta(a,b) = \delta'(a,b)$ for all $a,b \in K$ and combining these equations we obtain that

$$\sum_{a,b \in K} \delta'(a,b)c_H(a,b) \geq \sum_{a,b \in K} \vec{r}_{a,b}\delta'(a,b)$$

and this is true for any semi-metric δ' on K.

Hence, by strong duality, $\vec{r}$ must also be routable in G with congestion at most one. $\square$

Recall Definition 9 in which we defined the contraction quality $\alpha_0^c(G,K)$ of G,K – which is the quality of the best cut-sparsifier that can be achieved through a convex combination of 0-extensions. We can analogously define the quantity $\alpha_0^f(G,K)$ as the quality of the best flow-sparsifier that can be achieved through a convex combination of 0-extensions.

Theorem 18. *There is an algorithm that runs in time polynomial in n and k that returns a flow-sparsifier whose quality is at most $\alpha_0^f(G,K)$, however this flow-sparsifier is not necessarily realizable as a convex combination of* 0*-extensions.*

Proof: Consider the polytope

$$\begin{array}{lll}
\textbf{Type Z':} & \vec{X} \in Z' & \\
\textbf{Type 2.f:} & \sum_{a,b \in K} x_{a,b} d(a,b) \leq \alpha & \text{for all } \vec{d} \in D_G \\
 & 0 \leq x_{a,b} & \text{for all } a,b \in K
\end{array}$$

We can implement an efficient separation algorithm for the **Type Z'** inequalities using linear programming (and because the number of Earth-mover constraints in the definition of the set Z' is polynomial in n and k). We can also implement an efficient separation algorithm for the **Type 2.f** inequalities using minimum congestion routing. Hence we can run the Ellipsoid algorithm and find a point $\vec{X}$ in this polytope, if one exists. Let α^* be the minimum α for which the above polytope is non-empty – using binary search, we can determine this value.

Using Claim 13, the polytope

$$\begin{array}{lll}
\textbf{Type Z':} & \vec{X} \in Z' & \\
\textbf{Type 2.f:} & \sum_{a,b \in K} x_{a,b} d(a,b) \leq \alpha_0^f(G,K) & \text{for all } \vec{d} \in D_G \\
 & 0 \leq x_{a,b} & \text{for all } a, b \in K
\end{array}$$

is non-empty. Hence $\alpha^* \leq \alpha_0^f(G,K)$. Let $\vec{H}$ be the point in $\mathbb{R}^{\binom{k}{2}}$ found for the value $\alpha = \alpha^*$. Using Lemma 6, $\vec{H}$ is a flow-sparsifier and because $\vec{H}$ satisfies the **Type 2.f** inequalities, $\vec{H}$ has quality at most $\alpha^* \leq \alpha_0^f(G,K)$. □

4.1.3 The Max-Min Congestion Problem

Here we give an efficient $O(\log k)$-approximate separation oracle for the **Type 1.f** inequalities. The results in this section appear in [49]. If we could efficiently implement an exact separation oracle for these inequalities, we could efficiently compute a flow-sparsifier whose quality matches the quality of the best flow-sparsifier for the given G and K. In comparison, the algorithm in Section 4.1.2 computes a flow-sparsifier whose quality is at least as good as the quality of the best flow-sparsifier that can be achieved through 0-extensions.

Let us introduce the max-min congestion problem, which is the natural optimization problem associated with finding a violated **Type 1.f** if such an inequality exists.

Definition 24. *The input to the max-min congestion problem is a capacitated graph $G = (V,E)$, a set of terminals $K \subset V$ and a capacitated graph on just the terminals $H = (K, E_H)$. The goal is to compute a non-negative vector (i.e. a demand vector) $\vec{r} \in \mathbb{R}^{\binom{k}{2}}$ that maximizes the ratio*

$$\frac{cong_H(\vec{r})}{cong_G(\vec{r})}$$

Notice that the problem of determining if H is a flow-sparsifier (i.e. if H satisfies all **Type 1.f** inequalities) is equivalent to the problem of whether or not the optimal value to the max-min congestion problem is at most one. We also observe that if $K = V$, then this optimization problem is easy because $\vec{G}$ will be the demand vector that maximizes the above ratio – in much the same manner that there is a single demand vector $\vec{H}$ whose congestion we need to compute to determine the quality of H as a flow-sparsifier (see Lemma 1). The challenging aspect of this optimization problem is that if $K \neq V$, there is no longer a notion of a "hardest" flow routable in G (with congestion at most one) to route in H.

We will give an $O(\log k)$-approximation algorithm for the max-min congestion problem based on a novel use of oblivious routing. Roughly, the strategy is to compute a good oblivious routing scheme in H and then to compute a demand vector $\vec{r}$ that is routable in G (with congestion at most one) that maximizes the congestion of our fixed oblivious routing scheme in G. We can accomplish this goal by writing a separate linear program for each edge (a, b) in H and attempting to find an $\vec{r}$ that is routable in G that maximizes the congestion of the oblivious routing scheme on e. We can always find an oblivious routing scheme in H that is $O(\log k)$-competitive (since H has only k nodes). Hence if we find a vector $\vec{r}$ that is routable in G but which results in congestion C when using the oblivious routing scheme for H, this vector $\vec{r}$ must also have $cong_H(\vec{r})$ at least $\frac{C}{O(\log k)}$ in H.

As outlined above, our first step is to compute an $O(\log k)$-competitive oblivious routing scheme f for H. Such an oblivious routing scheme is guaranteed to exist, and can be found in polynomial time using the results due to Räcke [54] because the graph H has only k nodes. For each $a, b \in K$, f specifies a unit flow from a to b in G' and we will let $f^{a,b} : E_H \to \mathbb{R}^+$ be the corresponding assignment of flows to edges for this unit flow. Since f is $O(\log k)$-competitive, f has the property that for any demand vector $\vec{r}$, the congestion that results from routing according to f is within an $O(\log k)$ factor of the optimal congestion for routing $\vec{r}$ in H.

For any edge $(a, b) \in E_H$, we consider the following linear program $LP(a, b)$:

$$\begin{aligned} \max C_{a,b} &= \frac{\sum_{s,t} \vec{r}_{s,t} f^{s,t}(a,b)}{c_H(a,b)} \\ \text{s.t.} & \\ & \vec{r} \in P_G \end{aligned}$$

Note the set P_G is a polytope, and the condition $\vec{r} \in P_G$ just enforces that the demand vector $\vec{r}$ is routable in G withe congestion at most one. In fact, we could write out the condition $\vec{r} \in P_G$ explicitly using a polynomial sized set of constraints that enforce that $\vec{r}$ is realizable by a flow (that satisfies the standard conservation constraints) and for which the total flow on any edge in G does not exceed the capacity of this edge. Regardless, the linear program above can be solved efficiently (because the terms $f^{s,t}(a, b)$ are not variables, but are specified by a fixed, competitive oblivious routing scheme in H).

The interpretation of this linear program is that it finds a demand vector that can be routed with congestion at most one in G, and among all such flows maximizes the congestion of the oblivious routing scheme on the edge $(a, b) \in E_H$. We will solve the above linear program for all edges in H and output the demand vector $\vec{r^*}$ that achieves the maximum value $C_{a,b}$. Let (a, b) be the edge in H for which we obtain the maximum value to the above linear program.

Lemma 7. $cong_H(\vec{r^*}) \geq \frac{C_{a,b}}{O(\log k)}$

Proof: The congestion of the oblivious routing scheme for the demand vector $\vec{r^*}$ is at least $C_{a,b}$, since the oblivious routing scheme sends a total flow of $C_{a,b}c_H(a, b)$ on the edge (a, b) in H when routing the demand vector $\vec{r^*}$. Since the oblivious routing scheme f is $O(\log k)$-competitive, $cong_H(\vec{r^*}) \geq \frac{C_{a,b}}{O(\log k)}$. $\square$

Let OPT be the optimal value of the max-min congestion problem.

Lemma 8. $C_{a,b} \geq OPT$

Proof: Let $\vec{d}$ be the optimal demand vector, for the max-min congestion problem. We can assume $\vec{d} \in P_G$ by scaling this vector, and then $cong_H(\vec{r}) = OPT$. Consider the point $\vec{d}$ which is feasible in each linear program (one for each edge in H) above. The oblivious routing scheme specifies a routing of $\vec{d}$ in H. So there must be some edge (a', b') in H for which the total flow (that results from routing $\vec{d}$ according to the oblivious routing scheme f) is at least $cong_H(\vec{d})c_H(a', b') = OPTc_H(a', b')$. Hence the optimal value $C_{a',b'}$ for the linear program $LP(a', b')$ is at least OPT. $\square$

These lemmas immediately imply

Theorem 19. *There is an $O(\log k)$-approximation algorithm for the max-min congestion problem that runs in time polynomial in n and k.*

An interesting open question is whether the max-min congestion problem is even hard:

Open Question 1. *Is there an efficient algorithm for the max-min congestion problem?*

As we observed earlier, a positive answer to this question would immediately yield an efficient algorithm for computing the best quality flow-sparsifier for any given G and K.

4.2 Universal Rounding Algorithms

A good flow-sparsifier approximately preserves the congestion of every demand vector. Using the definition of the unit congestion polytope (see Definition 13), we can interpret an α-quality flow-sparsifier as a graph $H = (K, E_H)$ for which

$$P_G \subset P_H \subset \alpha P_G$$

Here we observe that many linear programming relaxations are based on multicommodity flows in the sense that the value of a relaxation (for, say, a graph partitioning problem) is only a function of the polytope P_G. Hence, a good flow-sparsifier approximately preserves the value of these many types of linear programming relaxations. For example, the linear programming relaxations for generalized sparsest cut, multi-cut and even the requirement cut problem each depend only on properties of P_G. In fact, in a technical sense, a good flow-sparsifier also approximately preserves the integrality gap of any linear programming relaxation that only depends upon P_G.

Here we impose additional structure on a good vertex sparsifier – we require a flow-sparsifier to not only have good quality, but also be decomposable into a convex combination of ”contraction-based” trees. We give an algorithm for computing an $O(\log k)$-quality flow-sparsifier that is explicitly represented as a convex combination of ”contraction-based” trees. This allows us to algorithmically reduce the input graph to a tree, while still approximately preserving the integrality gap of any relaxation (that depends only on P_G) within an $O(\log k)$ factor. Hence we can use any rounding algorithm for the relaxation on trees to get a rounding algorithm for general graphs.

This immediately implies a master theorem for graph partitioning problems - as long as the integrality gap of linear programming relaxation (that depends only on P_G) is bounded on trees, then the integrality gap is at most an $O(\log k)$ factor larger for general graphs. Additionally, if there is an efficient rounding algorithm for the relaxation when restricted to trees, then there is also an efficient rounding algorithm for general graphs whose approximation ratio that is at most an $O(\log k)$ factor larger. This result implies optimal bounds for many well studied graph partitioning problems (such as generalized sparsest cut, multi-cut and the requirement cut problem) as a special case, and even yields optimal bounds for more exotic problems that had not been studied before.

The results in this section appear in [20] and a similar approach is given in [25].

4.2.1 0-Decompositions

The notion of a 0-extension was introduced by Karzanov [39], and we made extensive use of this concept in Section 3. Here we introduce a related concept, that of a 0-decomposition. In a technical sense, the aim of this definition is to combine the useful properties of a 0-extension function and a decomposition tree [54].

Recall that given a 0-extension f, we can define a graph G_f on just the terminal set K that results from collapsing all node mapped to the same terminal. Again, recall that we let c_f denote the capacity function of this resulting graph.

Definition 25. *Given a tree T on K, and a 0-extension f, we can generate a 0-decomposition $G_{f,T} = (K, E_{f,T})$ as follows:*

1. *The only edges present in $G_{f,T}$ will be those in T, and for any edge $(a, b) \in E(T)$, let T_a, T_b be the subtrees containing a, b respectively that result from deleting (a, b) from T.*

2. *$c_{f,T}(a,b)$ (i.e. the capacity assigned to (a,b) in $G_{f,T}$ is defined as:*

$$c_{f,T}(a,b) = \sum_{u,v \in K \text{ and } u \in T_a, v \in T_b} c_f(u,v)$$

Hence a 0-decomposition is specified by a 0-extension f and a tree T on K. We first prove that any 0-decomposition is a flow-sparsifier. The proof of this claim is a generalization of the one given in Claim 7.

Claim 14. *$G_{f,T}$ is a flow-sparsifier*

Proof: We can think of forming a 0-decomposition through a series of operations that either contract an edge or re-route an edge (a,b) by deleting the capacity $c(a,b)$ associated with the edge, and choosing a path $p = a, p_1, p_2, ...p_s, b$ and incrementing the capacity of each edge in the path by $c(a,b)$. Call this latter operation a re-routing. The proof in Claim 7 demonstrates that a contraction operation cannot make the congestion of a demand vector $\vec{r}$ increase. A re-routing operation also cannot make the congestion of $\vec{r}$ increase: consider demand vector $\vec{r}$. We can assume, after scaling, that $\vec{r}$ has $cong_G(\vec{r}) = 1$. Consider the graph G' that results from re-routing the edge (a,b) along the path $p = a, p_1, p_2, ...p_s, b$. Given the minimum congestion routing of $\vec{r}$ in G, we can divert all flow through (a,b) to the path p and the resulting routing will still be a routing of $\vec{r}$ (in G') and will have congestion at most one. Hence $cong_{G'}(\vec{r}) \leq cong_G(\vec{r})$ and we can apply contraction and re-routing operations inductively to prove that $G_{f,T}$ is a flow-sparsifier. $\square$

Next we prove the main result in this section: we give an algorithm that can construct an $O(\log k)$-quality flow-sparsifier and a representation of this flow-sparsifier as a convex combination of 0-decompositions. This result is a generalization of the celebrated result of Räcke [54] in the sense that not only is the graph that we compute a convex combination of trees, but this graph also contains no non-terminals (and achieves an approximation ratio that is independent of the size of the original graph G).

Theorem 20. *There is an algorithm to construct a distribution γ on 0-decompositions such that $H = \sum_{f,T} \gamma(f,T) G_{f,T}$ is an $O(\log k)$-quality flow-sparsifier, and this algorithm runs in time polynomial in n and k.*

To prove this theorem, we can immediately use the machinery in Section 3.2.3, setting $\mathcal{H}' = \{G_{f,T}\}_{f,T}$. All that remains is to bound the integrality gap of $\mathcal{H}'$. Roughly, we need to show that for any metric $d \in D_G$ - i.e. is realizable in G with total capacity $\times$ distance units at most one - that there is a f, T such that $\sum_{a,b \in K} d(a,b) c_{f,T}(a,b) \leq O(\log k)$ where $c_{f,T}$ is the capacity function of $G_{f,T}$.

We can prove this by a randomized rounding procedure that is almost the same as the rounding procedure in [27]: We can scale up the distances in d to obtain a metric Δ in which the distances are all at least one and we assume 2^δ is an upper bound on the diameter of the metric space. Then we need to first choose a 0-extension f for which the cost against Δ is $O(\log k)$ times the cost of realizing Δ in G. We do this as follows:

```
Choose a random permutation $\pi(1), \pi(2), ..., \pi(k)$ of $K$
Choose $\beta$ uniformly at random from $[1, 2]$
$D_\delta \leftarrow \{V\}, i \leftarrow \delta - 1$
**while** $D_{i+1}$ has a cluster which contains more than one terminal **do**
  $\beta_i \leftarrow 2^{i-1}\beta$
  **for** $\ell = 1$ to $k$ **do**
    **for** every cluster $S$ in $D_{i+1}$ **do**
      Create a new cluster of all unassigned vertices in $S$ closer than $\beta_i$ to $\pi(\ell)$
    **end for**
  **end for**
  $i \leftarrow i - 1$
**end while**
```

Then, exactly as in [27], we can construct a decomposition tree from the rounding procedure. The root node is V corresponding to the partition D_δ, and the children of this node are all sets in the partition $D_{\delta-1}$. Each successive D_i is a refinement of D_{i+1}, so each set of D_i is made to be a child of the corresponding set in D_{i+1} that contains it. At each level i of this tree, the distance to the layer above is 2^i, and one can verify that this tree-metric associated with the decomposition tree dominates the original metric space Δ restricted to the set K. Note that the tree metric does not dominate Δ on V, because there are some nodes which are mapped to the same leaf node in this tree, and correspondingly have distance 0.

If we consider any edge (u, v), we can bound the expected distance in this tree metric from the leaf node containing u to the leaf-node containing v. In fact, this expected distance is only a function of the metric space Δ restricted to $K \cup \{u, v\}$. Accordingly, for any (u, v), we can regard the metric space that generates the tree-metric as a metric space on just $k+2$ points.

Formally, the rounding procedure in [27] is:

```
Choose a random permutation $\pi(1), \pi(2), ..., \pi(n)$ of $V$
Choose $\beta$ uniformly at random from $[1, 2]$
$D_\delta \leftarrow \{V\}, i \leftarrow \delta - 1$
while $D_{i+1}$ has a cluster that is not a singleton do
  $\beta_i \leftarrow 2^{i-1}\beta$
  for $\ell = 1$ to $n$ do
    for every cluster $S$ in $D_{i+1}$ do
      Create a new cluster of all unassigned vertices in $S$ closer than $\beta_i$ to $\pi(\ell)$
    end for
  end for
  $i \leftarrow i - 1$
end while
```

Formally, [27] proves a stronger statement than just that the expected distance (according to the tree-metric generated from the above rounding procedure) is $O(\log n)$ times the original distance. We will say that u, v are split at level i if these two nodes are contained in different sets of D_i. Let X_i be the indicator variable for this event.

Then the distance in the tree-metric Δ_T generated from the above rounding procedure is $\Delta_T(u, v) = \sum_i 2^{i+1} X_i$. In fact, [27] proves the stronger statement that this is true even if u, v are not in the metric space (i.e. $u, v \notin V$) but are always grouped in the cluster which they would be if they were in fact in the set V (provided of course that they can be grouped in such a cluster). More formally, we set $V' = V \cup \{u, v\}$ and if the step "Create a new cluster of all unassigned vertices in S closer than β_i to $\pi(\ell)$" is replaced with "Create a new cluster of all unassigned vertices in V' in S closer than β_i to $\pi(\ell)$", then [27] actually proves in this more general context that

$$\sum_i 2^{i+1} X_i \leq O(\log n)\Delta(u, v)$$

If we input the metric space Δ restricted to K into the above rounding procedure (but at each clustering stage we consider all of V) we get exactly our rounding procedure. So the main theorem in [27] (or rather our restatement of it) is

(If Δ_T is the tree-metric generated from the above rounding procedure)

Theorem 21. *[27] For all u, v, $E[\Delta_T(u, v)] \leq O(\log k)\Delta(u, v)$.*

At the end of the rounding procedure, we have a tree in which each leaf corresponds to a subset of V that contains at most 1 terminal. We are given a tree-metric Δ_T on V associated with the output of the algorithm, and this tree-metric has the property that $\sum_{(u,v)\in E} c(u, v)\Delta_T(u, v) \leq O(\log k)$.

We would like to construct a tree T' from T which has only leafs which contain exactly one terminal. We first state a simple claim that will be instructive in order to do this:

Claim 15. *Given a tree metric Δ_T on a tree T on K, $\sum_{a,b\in K} \Delta_T(a, b)c_{f,T}(a, b) = \sum_{a,b\in K} \Delta_T(a, b)c_f(a, b)$.*

Proof: The graph $G_{f,T}$ can be obtained from G_f by iteratively re-routing some edge $(a, b) \in E_f$ along the path connecting a and b in T and adding $c_f(a, b)$ capacity to each edge on this path, and finally deleting the edge (a, b). The original cost of this edge is $c(a, b)\Delta_T(a, b)$, and if $a = p_1, p_2, ..., p_r = b$ is the path connecting a and b in T, the cost after performing this operation is $c(a, b)\sum_{i=1}^{r-1} \Delta_T(p_i, p_{i+1}) = c(a, b)\Delta_T(a, b)$ because Δ_T is a tree-metric. $\square$

We can think of each edge (u, v) as being routed between the deepest nodes in the tree that contain u and v respectively, and the edge pays $c(u, v)$ times the distance according to the tree-metric on this path. Then we can perform the following procedure: each time we find a node in the tree which has only leaf nodes as children and none of these leaf nodes contains a terminal, we can delete these leaf nodes. This cannot increase the cost of the edges against the tree-metric because every edge (which we regard as routed in the tree) is routed on the same, or a shorter path. After this procedure is done, every leaf node that doesn't contain a terminal contains a parent p that has a terminal node a. Suppose that the deepest node in the tree that contains a is c We can take this leaf node, and delete it, and place all nodes in the tree-node c. This procedure only affects the cost of edges with one endpoint in the leaf node that we deleted, and at most doubles the cost paid by the edge because distances in the tree are geometrically decreasing. So if we iteratively perform the above steps, the total cost after performing these operations is at most 4 times the original cost.

And it is easy to see that this results in a natural 0-extension in which each node u is mapped to the terminal corresponding to the deepest node that u is contained in.

Each edge pays a cost proportional to a tree-metric distance between the endpoints of the edge. So we know that $\sum_{a,b \in K} \Delta_T(a,b) c_{f,T}(a,b) = O(\log k)$ because this cost increased by at most a factor of 4 from iteratively performing the above steps. Yet using the Claim 15, we get a 0-extension f and a tree T such that $\sum_{a,b \in K} \Delta_T(a,b) c_{f,T}(a,b) = O(\log k)$ and because Δ_T dominates Δ when restricted to K, this implies that

$$\sum_{a,b \in K} \Delta(a,b) c_{f,T}(a,b) \leq \sum_{a,b \in K} \Delta_T(a,b) c_{f,T}(a,b) = O(\log k)$$

and this implies an $O(\log k)$ bound on the integrality gap of $\mathcal{H}' = \{G_{f,T}\}_{f,T}$.

We can immediately apply the results in Section 3.2.4 and this implies the main theorem (stated above) in this section - that there is an efficient algorithm for computing an $O(\log k)$-quality flow-sparsifier that can be realized as a convex combination of contraction-based trees.

4.2.2 Applications

Recall the definition of a fractional graph partitioning problem that we gave in Section 2.4:

We call an optimization problem D a fractional graph partitioning problem if it can be written as (for some monotone increasing function f):

$$\begin{aligned} \min \quad & \textstyle\sum_{(u,v)\in E} c(u,v)d(u,v) \\ \text{s.t.} \quad & \\ & d : V \times V \to \Re^{+} \text{ is a semi-metric} \\ & f(d\Big|_{K}) \geq 1 \end{aligned}$$

We also gave a number of examples of NP-hard graph partitioning problems for which known linear programming relaxations fit into the above framework. Here we will demonstrate a universal rounding algorithm (that succeeds for any fractional graph partitioning problem) and reduces the input graph to a tree at the cost of an $O(\log k)$-factor.

We will let $D(G)$ denote the optimal value to the above linear program – note that this quantity also depends on the choice of K and the function f. The notion of an "integral" solution corresponds to some restriction on the set of admissible metrics d. The above linear program is a relaxation in that this restriction is removed. We let $ID(G)$ denote the optimal solution, when this restriction is enforced:

$$\begin{aligned} \min \quad & \textstyle\sum_{(u,v)\in E} c(u,v)d(u,v) \\ \text{s.t.} \quad & \\ & d : V \times V \to \Re^{+} \text{ is a semi-metric} \\ & d \text{ is "integral"} \\ & f(d\Big|_{K}) \geq 1 \end{aligned}$$

Recall that an "integral" solution will not necessarily coincide with the set of metric spaces that have only integer distances, but could alternatively be some other condition (as long as this condition is only a function of d restricted to K) – for example, the set of "integral" solutions could be the set of all metrics that when restricted to K are cut-metrics.

Our first step is to prove that graph partitioning "costs more" on a 0-decomposition $G_{f,T}$ than on the original graph G.

Lemma 9. *For any 0-decomposition $G_{f,T}$, we have that $D(G) \leq D(G_{f,T})$ and $ID(G) \leq ID(G_{f,T})$.*

Roughly, this lemma will follow because $G_{f,T}$ is a flow-sparsifier and hence makes routing problems easier and partitioning problems harder.

Proof: As in the proof of Claim 14, we can think of forming a 0-decomposition through a series of operations that either contract an edge or re-route an edge (a, b) by deleting the capacity $c(a, b)$ associated with the edge, and choosing a path $p = a, p_1, p_2, ...p_s, b$ and incrementing the capacity of each edge in the path by $c(a, b)$.

Consider first the operation of contracting an edge e in G to form G'. We can alternatively think of this operation as setting the capacity of e to be infinite. Then for any metric d (integral or otherwise) for which $f(d\Big|_K) \geq 1$ the quantity $\sum_{(u,v)\in E'} c'(u, v)d(u, v)$ is at least as large as $\sum_{(u,v)\in E} c(u, v)d(u, v)$ where E' and c' denote the edge set and capacity function for the graph G' respectively. Hence the contracting an edge cannot decrease the value of the optimal solution to either the graph partitioning problem or the relaxation.

Similarly, the operation of re-routing an edge e along a path p does not change the set of metrics d for which $f(d\Big|_K) \geq 1$. And if $c(e)$ is the capacity of edge e, the edge contributes $c(e)d(e)$ to $\sum_{(u,v)\in E} c(u, v)d(u, v)$ before re-routing and contributes $c(e)\sum_{e'\in P} d(e') \geq c(e)d(e)$ after re-routing (where the last inequality follows from the condition that d be a metric).

Hence the lemma follows by inductively applying contraction and re-routing operations to form the 0-decomposition $G_{f,T}$. $\Box$

Lemma 10. *Let $H = \sum_{f,T} \gamma(f, T)G_{f,T}$ be an α-quality flow-sparsifier. Then $D(G_{f,T}) \leq \alpha D(G)$.*

Proof: Let d be a metric on V such that $\sum_{(u,v)} c(u, v)d(u, v) = D(G)$ and $f(d\Big|_K) \geq 1$. Since H has quality α as a flow-sparsifier, there is a routing of H in G for which the congestion on any edge is at most α. Since $H = \sum_{f,T} \gamma(f, T)G_{f,T}$ we can regard this routing of H in G as a simultaneous routing of each capacitated $\gamma(f, T)G_{f,T}$ in G.

For each f, T in the support of γ, we will construct a semi-metric $d_{f,T}$ that dominates d on K and hence satisfies the condition $f(d_{f,T}\Big|_K) \geq 1$, and yet if we sample a f, T pair according to γ, the expected value of $\sum_{a,b\in K} c_{f,T}(a, b)d_{f,T}(a, b)$ will be at most $\alpha D(G)$.

So when constructing $d_{f,T}$, initialize all distances to zero. $d_{f,T}$ will be a shortest path metric on G restricted to K. Each edge $(a, b) \in E_{f,T}$ is routed to some distribution on paths connecting a and b in G. A total of $\gamma(f, T)c_{f,T}(a, b)$ flow is routed on some distribution on paths, and consider a path p that carries $C(p)$ total flow from a to b in the routing of $\gamma(f, T)G_{f,T}$. If the total distance along this path is $d(p)$, we increment the distance $d_{f,T}$ on the edge (a, b) in $G_{f,T}$ by $\frac{d(p)C(p)}{\gamma(f,T)c_{f,T}(a,b)}$, and we do this for all such paths. We do this also for each (a, b) in $G_{f,T}$.

The distance function $d_{f,T}$ dominates d when both metrics are restricted to K, because the distance that we allocate to the edge (a, b) in $G_{f,T}$ is a convex combination of the distances along paths connecting a and b in G, each of which is at least $d(a, b)$. Hence $d_{f,T}$ satisfies the condition $f(d_{f,T}\Big|_K) \geq 1$. We can also compute the expected value of $\sum_{a,b \in K} c_{f,T}(a, b)d_{f,T}(a, b)$ when we sample a f, T pair according to γ:

$$\sum_{f,T} \gamma(f, T) \sum_{(a,b) \in E_{f,T}} c_{f,T}(a, b)d_{f,T}(a, b) \leq \alpha \sum_{(a,b) \in E} c(a, b)d(a, b) = \alpha D(G)$$

Hence there is some f, T for which

$$\sum_{(a,b) \in E_{f,T}} c_{f,T}(a, b)d_{f,T}(a, b) \leq \alpha D(G)$$

and yet $f(d_{f,T}\Big|_K) \geq 1$.

□

We can now prove our main theorem in this section:

Theorem 22. *Given a polynomial time rounding algorithm for the semi-metric relaxation on trees that achieves an approximation ratio of C, there is a polynomial time rounding algorithm for the semi-metric relaxation on general graphs that achieves an approximation ratio of $O(C \log k)$.*

Proof: Given G and K, we can use Theorem 20 to construct an $O(\log k)$-quality flow sparsifier $H = \sum_{f,T} \gamma(f, T)G_{f,T}$ that can be realized as a convex combination of contraction-based trees in time polynomial in n and k. We can compute the value of the semi-metric relaxation for each f, T and select a pair f, T for which $D(G_{f,T}) \leq O(\log k)D(G)$ – such a pair is guaranteed to exist using Lemma 10.

$G_{f,t}$ is a tree, and hence we can apply the given rounding algorithm and obtain an "integral" solution d^* for which $\sum_{(a,b)\in E_{f,T}} c_{f,T}(a,b)d^*(a,b) \leq CO(\log k)D(G)$ and $f(d^*\Big|_K) \geq 1$. Recall that the condition that d^* be "integral" is only a function of d^* restricted to K. In $G_{f,T}$ each non-terminal u is mapped to a terminal $f(u)$. We can extend d^* to V such that $d(u, f(u)) = 0$ and this extension is still "integral". Then

$$\sum_{u,v} c(u,v)d^*(u,v) = \sum_{a,b\in K} c_f(a,b)d^*(a,b) \leq \sum_{(a,b)\in E_{f,T}} c_{f,T}(a,b)d^*(a,b)$$

where c_f is the capacity function of the 0-extension G_f. Then the last inequality follows because $G_{f,T}$ can be obtained from G_f by re-routing edges, and each such operation can only increase the total sum of distance $\times$ capacity units. Hence d^* is an "integral" solution that satisfies $\sum_{u,v} c(u,v)d^*(u,v) \leq CO(\log k)D(G)$ and $f(d^*\Big|_K) \geq 1$.

□

This rounding algorithm simultaneously gives optimal rounding algorithms (based on the standard linear programming relaxations) for both generalized sparsest cut and multicut. The previous techniques for rounding a fractional solution to generalized sparsest cut [46], [8] rely on metric embedding results, and the techniques for rounding fractional solutions to multicut [31] rely on purely combinatorial, region-growing arguments. Yet our rounding algorithm is based on producing both a combinatorially simple graph and a geometrically simple class of metrics.

Chapter 5

Lower Bounds and Separations

Morally, the results in this thesis are centered around the idea that the structure of near-optimal solutions is *simple* for routing and graph partitioning problems. We can find a near-optimal routing for a demand vector knowing only a good flow-sparsifier H, and a low-congestion embedding of H into the original graph G

Here we prove lower bounds for vertex sparsification. We can regard these results as answering the question: What if we consider optimal solutions (or solutions that are within a constant factor of optimal) – are the solutions still *simple*? The answer, in this case, is no – optimal solutions (or solutions within a constant factor of optimal) cannot be represented concisely on a vertex sparsifier. In some sense, this lower bound reinforces the efficacy of the methods that we have presented: We are getting something *new* when considering near-optimal solutions, because near-optimal solutions are simple in a way that optimal solutions are not.

Specifically, we prove that in general we cannot hope for a cut-sparsifier whose quality is better than $\Omega(\log^{1/4} k)$. Makarychev and Makarychev concurrently gave a better lower bound of $\tilde{\Omega}(\sqrt{\log k})$. As flow-sparsification is no easier than cut-sparsification, this immediately implies that in general we cannot hope for a flow-sparsifier whose quality is better than $\tilde{\Omega}(\sqrt{\log k})$ either.

We also prove a number of asymptotic separations - we give an example in which the quality of the best cut-sparsifier is asymptotically better than the quality of the best cut-sparsifier that can be achieved through contractions. To establish this result, we define a cut-sparsifier based on random walks and we use tools from Harmonic analysis to analyze the quality of this cut-sparsifier. We also give an example in which the quality of the best flow-sparsifier is asymptotically worse than the quality of the best cut-sparsifier.

5.1 The Best Cut Sparsifier Doesn't Always Come from Contractions

Here we give an infinite family of graphs for which the quality of the best cut-sparsifier is at least $\Omega(\log^{1/4} k)$. Our proof is elementary, and is only based on duality and estimates for binomial coefficients. Additionally, we use this example to demonstrate a separation - that the best quality cut-sparsifiers do not always come through contractions. In fact, we demonstrate that the the quality of the best cut-sparsifier is asymptotically better than the quality of the best cut-sparsifier that can be achieved through 0-extensions. The results in this section appear in [20].

Let us define the graph G that we will use to obtain these results: Let Y be the hypercube of size 2^d for $d = \log k$. To generate G from Y: for every node $y_s \in Y$ (i.e. $s \in \{0,1\}^d$), we add a terminal z_s and connect the z_s to y_s using an edge of capacity $\sqrt{d}$. Additionally we set the capacity of each edge in the hypercube to one.

5.1.1 Lower Bounds

First, we give an $\Omega(\sqrt{d})$ integrality gap for the semi-metric relaxation of the 0-extension problem on this graph, even when the semi-metric (actually on all of V) is ℓ_1. Hence this is a lower bound on the integrality gap of the cut-cut relaxation given in Section 3.1.4 and immediately implies that any cut-sparsifier that can be realized as a convex combination of 0-extensions has quality at least $\Omega(\sqrt{d}) = \Omega(\sqrt{\log k})$. Such a bound is actually implicit in the work of [38] too.

Consider the distance assignment to the edges: Each edge connecting a terminal to a node in the hypercube - i.e. an edge of the form (z_s, y_s) is assigned distance $\sqrt{d}$ and every other edge in the graph is assigned distance 1. Then let σ be the shortest path metric on V

given these edge distances.

Recall that given $U \subset V$, Δ_U is the corresponding cut metric – see Definition 10.

Claim 16. *σ is an ℓ_1 semi-metric on V, and in fact there is a weighted combination δ of cuts such that $\sigma(u,v) = \sum_U \delta(U)\Delta_U(u,v)$ and $\sum_U \delta(U)h(U) = O(kd)$*

Proof: We can take $\delta(U) = 1$ for any cut $(U, V-U)$ s.t. $U = \{z_s \cup y_s | s_i = 1\}$ - i.e. U is the axis-cut corresponding to the i^{th} bit. We also take $\delta(U) = \sqrt{d}$ for each $U = \{z_s\}$. This set of weights will achieve $\sigma(u,v) = \sum_U \delta(U)\Delta_U(u,v)$, and also there are d axis cuts each of which has capacity $h(U) = \frac{k}{2}$ and there are k singleton cuts of weight $\sqrt{d}$ and capacity $\sqrt{d}$ so the total cost is $O(kd)$. $\square$

Yet if we take D equal to the restriction of σ on K, then the optimal solution to the 0-extension problem defined by G, K and D is at least $\Omega(kd^{3/2})$:

Lemma 11. *For any 0-extension f, $\sum_{u,v \in V} D(f(u), f(v))c(u,v) \geq \Omega(kd^{3/2})$*

Proof: Consider any 0-extension f. And we can define the weight of any terminal a as $wt_f(a) = |f^{-1}(a)| = |\{v | f(v) = a\}|$. Then $\sum_a wt_f(a) = n$ because each node in V is assigned to some terminal. We can define a terminal as heavy with respect to f if $wt_f(a) \geq \sqrt{k}$ and light otherwise. Obviously, $\sum_a wt_f(a) = \sum_{a \text{ s.t. } a \text{ is light}} wt_f(a) + \sum_{a \text{ s.t. } a \text{ is heavy}} wt_f(a)$ so the sum of the sizes of either all heavy terminals or of all light terminals is at least $\frac{n}{2} = \Omega(k)$.

Suppose that $\sum_{a \text{ s.t. } a \text{ is light}} wt_f(a) = \Omega(k)$. For any pair of terminals a, b, $D(a,b) \geq \sqrt{d}$. Also for any light terminal a, $f^{-1}(a) - \{a\}$ is a subset of the Hypercube of at most $\sqrt{k}$ nodes, and the small-set expansion of the Hypercube implies that the number of edges out of this set is at least $\Omega(wt_f(a) \log k) = \Omega(wt_f(a)d)$. Each such edge pays at least $\sqrt{d}$ cost, because $D(a,b) \geq \sqrt{d}$ for all pairs of terminals. So this implies that the total cost of the 0-extension f is at least $\sum_{a \text{ s.t. } a \text{ is light}} \Omega(wt_f(a)d^{3/2})$.

Suppose that $\sum_{a \text{ s.t. } a \text{ is heavy}} wt_f(a) = \Omega(k)$. Consider any heavy terminal z_t, and consider any $y_s \in f^{-1}(z_t)$ and $t \neq s$. Then the edge (y_s, z_s) is capacity $\sqrt{d}$ and pays a total distance of $D(z_t, z_s) \geq \sigma(y_t, y_s)$. Consider any set U of $\sqrt{k}$ nodes in the Hypercube. If we attempt to pack these nodes so as to minimize $\sum_{y_s \in U} \sigma(y_s, y_t)$ for some fixed node y_t, then the packing that minimizes the quantity is an appropriately sized Hamming ball centered at y_t. In a Hamming ball centered at the node y_t of at least $\sqrt{k}$ total nodes, the average distance from y_t is $\Omega(\log k) = \Omega(d)$, and so this implies that $\sum_{y_s \in f^{-1}(z_t)} D(z_t, z_s) \geq \sum_{y_s \in f^{-1}(z_t)} D(y_t, y_s) \geq \Omega(wt_f(z_t)d)$. Each such edge has capacity $\sqrt{d}$ so the total cost of the 0-extension f is at least $\sum_{a \text{ s.t. } a \text{ is heavy}} \Omega(wt_f(a)d^{3/2})$ □

As noted, using the results in Section 3.1.4, this immediately implies that any cut-sparsifier that can be realized as a convex combination of 0-extensions has quality at least $\Omega(\sqrt{d}) = \Omega(\sqrt{\log k})$.

Next we prove an un-restricted lower bound – that for any cut-sparsifier (even one that is not necessarily a convex combination of 0-extensions) has quality at least $\Omega(d^{1/4}) = \Omega(\log^{1/4} k)$. Note that his lower bound is weaker than the lower bound for contraction-based cut-sparsifiers, but as well will demonstrate, there is actually a cut-sparsifier that has quality $o(\sqrt{\log k})$. Hence we will prove a separation: cut-sparsification through contraction can be asymptotically harder than unrestricted cut-sparsification.

The particular example G that we gave above has many symmetries, and we can use these symmetries to justify considering only symmetric cut-sparsifiers. The fact that these cut-sparsifiers can be assumed without loss of generality to have nice symmetry properties, translates to that any such cut-sparsifier H is characterized by a much smaller set of variables rather than one variable for every pair of terminals. In fact, we will be able to reduce the number of variables from $\binom{k}{2}$ to $\log k$. This in turn will allow us to consider a much smaller family of cuts in G in order to derive that the system is infeasible. In fact, we will only consider sub-cube cuts (cuts in which $U = \{z_s \cup y_s | s = [0, 0, 0, ...0, *, *, ..., *]\}$) and the Hamming ball $U = \{z_s \cup y_s | d(y_s, y_0) \leq \frac{d}{2}\}$.

Definition 26. *The operation J_s for some $s \in \{0,1\}^d$ which is defined as $J_s(y_t) = y_{t+s \mod 2}$ and $J_s(z_t) = z_{t+s \mod 2}$. Also let $J_s(U) = \cup_{u \in U} J_s(u)$.*

Definition 27. *For any permutation $\pi : [d] \rightarrow [d]$, $\pi(s) = [s_{\pi(1)}, s_{\pi(2)}, ...s_{\pi(d)}]$. Then the operation J_π for any permutation π is defined at $J_\pi(y_t) = y_{\pi(t)}$ and $T_\pi(z_t) = z_{\pi(t)}$. Also let $J_\pi(U) = \cup_{u \in U} T_\pi(u)$.*

Claim 17. *For any subset $U \subset V$ and any $s \in \{0,1\}^d$, $h(U) = h(J_s(U))$.*

Claim 18. *For any subset $U \subset V$ and any permutation $\pi : [d] \to [d]$, $h(U) = h(J_\pi(U))$.*

Both of these operations are automorphisms of the capacitated graph G and also send the set K to K.

Lemma 12. *If there is a cut-sparsifier H for G which has quality α, then there is a cut-sparsifier H' which has quality at most α and is invariant under the automorphisms of the capacitated graph G that send K to K.*

Proof: Given the cut-sparsifier H, we can apply an automorphism J to G, and because $h(U) = h(J(U))$, this implies that $h_K(A) = \min_{U \text{ s.t. } U \cap K = A} h(J(U))$. Also $J(U \cap K) = J(U) \cap J(K) = J(U) \cap K$ so we can re-write this last line as

$$\min_{U \text{ s.t. } U\cap K=A} h(J(U)) = \min_{U' \text{ s.t. } J(U')\cap K=J(A)} h(J(U'))$$

And if we set $U' = J^{-1}(U)$ then this last line becomes equivalent to

$$\min_{U' \text{ s.t. } J(U')\cap K=J(A)} h(J(U')) = \min_{U \text{ s.t. } U\cap K=J(A)} h(U) = h_K(J(A))$$

So the result is that $h_K(A) = h_K(J(A))$ and this implies that if we do not re-label H according to J, but we do re-label G, then for any subset A, we are checking whether the minimum cut in G re-labeled according to J, that separates A from $K - A$ is close to the cut in H that separates A from $K - A$. The minimum cut in the re-labeled G that separates A from $K - A$, is just the minimum cut in G that separates $J^{-1}(A)$ from $K - J^{-1}(A)$ (because the set $J^{-1}(A)$ is the set that is mapped to A under J). So H is an α-quality cut-sparsifier for the re-labeled G iff for all A:

$$h_K(A) = h_K(J^{-1}(A)) \leq h'(A) \leq \alpha h_K(J^{-1}(A)) = \alpha h_K(A)$$

which is of course true because H is an α-quality cut-sparsifier for G.

So alternatively, we could have applied the automorphism J^{-1} to H and not re-labeled G, and this resulting graph $H_{J^{-1}}$ would also be an α-quality cut-sparsifier for G. Also, since the set of α-quality cut-sparsifiers is convex (as it is defined by a system of inequalities), we can find a cut-sparsifier H' that has quality at most α and is a fixed point of the group of automorphisms, and hence invariant under the automorphisms of G as desired. $\square$

Corollary 7. *If α is the best quality cut-sparsifier for the above graph G, then there is an α quality cut-sparsifier H in which the capacity between two terminals z_s and z_t is only dependent on the Hamming distance $Hamm(s,t)$.*

Proof: Given any quadruple z_s, z_t and $z_{s'}, z_{t'}$ s.t. $Hamm(s,t) = Hamm(s',t')$, there is a concatenation of operations from J_s, J_π that sends s to s' and t to t'. This concatenation of operations J is in the group of automorphisms that send K to K, and hence we can assume that H is invariant under this operation which implies that $c_H(s,t) = c_H(s',t')$. $\square$

One can regard any cut-sparsifier (not just ones that result from contractions) as a set of $\binom{k}{2}$ variables, one for the capacity of each edge in H. Then the constraints that H be an α-quality cut-sparsifier are just a system of inequalities, one for each subset $A \subset K$ that enforces that the cut in H is at least as large as the minimum cut in G (i.e. $h'(A) \geq h_K(A)$) and one enforcing that the cut is not too large (i.e. $h'(A) \leq \alpha h_K(A)$). See also Section 4.1.1.

Then in general, one can derive lower bounds on the quality of cut-sparsifiers by showing that if α is not large enough, then this system of inequalities is infeasible meaning that there is not cut-sparsifier achieving quality α. Unlike the above argument, this form of a lower bound is much stronger and does not assume anything about how the cut-sparsifier is generated. In fact, this is a *canonical* approach to giving a lower bound. Applying Farka's Lemma: a good quality cut-sparsifier exists if and only if we cannot prove the corresponding system of inequalities is infeasible by adding and subtracting constraints to obtain a contradiction.

Theorem 23. *For $\alpha = \Omega(\log^{1/4} k)$, there is no cut-sparsifier H for G which has quality at most α.*

Proof: Assume that there is a cut-sparsifier H' of quality at most α. Then using the above corollary, there is a cut-sparsifier H of quality at most α in which the capacity from a to b is only a function of $Hamm(a,b)$. Then for each $i \in [d]$, we can define a variable w_i as the total capacity of edges incident to any terminal of length i. I.e. $w_i = \sum_{b \text{ s.t. } Hamm(a,b)=i} c_H(a,b)$.

For simplicity, here we will assume that all cuts in the sparsifier H are at most the cost of the corresponding minimum cut in G and at least $\frac{1}{\alpha}$ times the corresponding minimum cut. This of course is an identical set of constraints that we get from dividing the standard definition that we use in this paper for α-quality cut-sparsifiers by α.

We need to derive a contradiction from the system of inequalities that characterize the set of α-quality cut sparsifiers for G. As we noted, we will consider only the sub-cube cuts (cuts in which $U = \{z_s \cup y_s | s = [0,0,0,....0,*,*,...*]\}$) and the Hamming ball $U = \{z_s \cup y_s | d(y_s, y_0) \leq \frac{d}{2}\}$, which we refer to as the Majority Cut.

Consider the Majority Cut: There are $\Theta(k)$ terminals on each side of the cut, and most terminals have Hamming weight close to $\frac{d}{2}$. In fact, we can sort the terminals by Hamming weight and each weight level around Hamming weight $\frac{d}{2}$ has roughly a $\Theta(\frac{1}{\sqrt{d}})$ fraction of the terminals. Any terminal of Hamming weight $\frac{d}{2} - \sqrt{i}$ has roughly a constant fraction of their total capacity w_i crossing the cut in H, because choosing a random terminal Hamming distance i from any such terminal corresponds to flipping i coordinates at random, and throughout this process there are almost an equal number of 1s and 0s. Hence this process is well-approximated by a random walk starting at $\sqrt{i}$ on the integers, which equally likely moves forwards and backwards at each step for i total steps, and asking the probability that the walk ends at a negative integer.

In particular, for any terminal of Hamming weight $\frac{d}{2} - t$, the fraction of the capacity w_i that crosses the Majority Cut is $O(exp\{-\frac{t^2}{i}\})$. So the total capacity of length i edges (i.e. edges connecting two terminals at Hamming distance i) cut by the Majority Cut is $O(w_i |\{z_s | Hamm(s,0) \geq \frac{d}{2} - \sqrt{i}\}|) = O(w_i \sqrt{i/d})k$ because each weight class close to the boundary of the Majority cut contains roughly a $\Theta(\frac{1}{\sqrt{d}})$ fraction of the terminals. So the total capacity of edges crossing the Majority Cut in H is $O(k \sum_{i=1}^{d} w_i \sqrt{i/d})$

Let $A = \{z_s | d(y_s, y_0) \leq \frac{d}{2}\}$. The total capacity crossing the minimum cut in G separating A from $K - A$ is $\Theta(k\sqrt{d})$. And because the cuts in H are at least $\frac{1}{\alpha}$ times the corresponding minimum cut in G, this implies $\sum_{i=1}^{d} w_i \sqrt{i/d} \geq \Omega(\frac{\sqrt{d}}{\alpha})$

Next, we consider the set of sub-cube cuts. For $j \in [d]$, let $A_j = \{z_s | s_1 = 0, s_2 = 0, ..s_j = 0\}$. Then the minimum cut in G separating A_j from $K - A_j$ is $\Theta(|A_j| \min(j, \sqrt{d}))$, because each node in the Hypercube which has the first j coordinates as zero has j edges out of the sub-cube, and when $j > \sqrt{d}$, we would instead choose cutting each terminal $z_s \in A_j$ from the graph directly by cutting the edge (y_s, z_s).

Also, for any terminal in A_j, the fraction of length i edges that cross the cut is approximately $1-(1-\frac{j}{d})^i = \Theta(\min(\frac{ij}{d}, 1))$. So the constraints that each cut in H be at most the corresponding minimum cut in G give the inequalities $\sum_{i=1}^{d} \min(\frac{ij}{d}, 1) w_i \leq O(\min(j, \sqrt{d}))$

We refer to the above constraint as B_j. Multiply each B_j constraint by $\frac{1}{j^{3/2}}$ and adding up the constraints yields a linear combination of the variables w_i on the left-hand side. The coefficient of any w_i is

$$\sum_{j=1}^{d-1} \frac{\min(\frac{ij}{d}, 1)}{j^{3/2}} \geq \sum_{j=1}^{d/i} \frac{\frac{ij}{d}}{j^{3/2}}$$

And using the Integration Rule this is $\Omega(\sqrt{\frac{i}{d}})$.

This implies that the coefficients of the constraint B resulting from adding up $\frac{1}{j^{3/2}}$ times each B_j for each w_i are at least as a constant times the coefficient of w_i in the Majority Cut Inequality. So we get

$$\sum_{j=1}^{d-1} \frac{1}{j^{3/2}} \min(j, \sqrt{d}) \geq \Omega\Big(\sum_{j=1}^{d-1} \frac{1}{j^{3/2}} \sum_{i=1}^{d} \min(\frac{ij}{d}, 1) w_i\Big) \geq \Omega\Big(\sum_{i=1}^{d} w_i \sqrt{\frac{i}{d}}\Big) \geq \Omega\Big(\frac{\sqrt{d}}{\alpha}\Big)$$

And we can evaluate the constant

$$\sum_{j=1}^{d-1} j^{-3/2} \min(j, \sqrt{d}) = \sum_{j=1}^{\sqrt{d}} j^{-1/2} + \sqrt{d} \sum_{j=\sqrt{d}+1}^{d-1} j^{-3/2}$$

Using the Integration Rule, this evaluates to $O(d^{1/4})$. This implies $O(d^{1/4}) \geq \frac{\sqrt{d}}{\alpha}$ and in particular this implies $\alpha \geq \Omega(d^{1/4})$. So the quality of the best cut-sparsifier for H is at least $\Omega(\log^{1/4} k)$. $\square$

5.1.2 A Candidate Cut-Sparsifier

Here we give a cut-sparsifier H which we can prove has quality $o(\sqrt{\log k})$ for the graph G, which is asymptotically better than the best cut-sparsifier that can be generated from contractions.

As we noted, we can assume that the capacity assigned between a pair of terminals in H, $c_H(a, b)$ is only a function of the Hamming distance from a to b. In G, the minimum cut separating any singleton terminal $\{z_s\}$ from $K - \{z_s\}$ is just the cut that deletes the edge (z_s, y_s). So the capacity of this cut is $\sqrt{d}$. We want a good cut-sparsifier to approximately preserve this cut, so the total capacity incident to any terminal in H will also be $\sqrt{d}$ - i.e. $c'(\{z_s\}) = \sqrt{d}$.

We distribute this capacity among the other terminals as follows: We sample $t \sim_\rho s$, and allocate an infinitesimal fraction of the total weight $\sqrt{d}$ to the edge (z_s, z_t). Equivalently, the capacity of the edge connecting z_s and z_t is just $Pr_{u \sim_\rho t}[u = s]\sqrt{d}$. We choose $\rho = 1 - \frac{1}{\sqrt{d}}$. This choice of ρ corresponds to flipping each bit in t with probability $\Theta(\frac{1}{\sqrt{d}})$ when generating u from t. We prove that the graph H has cuts at most the corresponding minimum-cut in G.

This cut-sparsifier H has cuts at most the corresponding minimum-cut in G. In fact, a stronger statement is true: $\vec{H}$ can be routed as a flow in G with congestion $O(1)$. Consider the following explicit routing scheme for $\vec{H}$: Route the $\sqrt{d}$ total flow in $\vec{H}$ out of z_s to the node y_s in G. Now we need to route these flows through the Hypercube in a way that does not incur too much congestion on any edge. Our routing scheme for routing the edge from z_s to z_t in $\vec{H}$ from y_s to y_t will be symmetric with respect to the edges in the Hypercube: choose a random permutation of the bits $\pi : [d] \rightarrow [d]$, and given $u \sim_\rho t$, fix each bit in the order defined by π. So consider $i_1 = \pi(1)$. If $t_{i_1} \neq u_{i_1}$, and the flow is currently at the node x, then flip the i_1^{th} bit of x, and continue for $i_2 = \pi(2), i_3, ...i_d = \pi(d)$.

Each permutation π defines a routing scheme, and we can average over all permutations π and this results in a routing scheme that routes $\vec{H}$ in G.

Claim 19. *This routing scheme is symmetric with respect to the automorphisms J_s and J_π of G defined above.*

Corollary 8. *The congestion on any edge in the Hypercube incurred by this routing scheme is the same.*

Lemma 13. *The above routing scheme will achieve congestion at most $O(1)$ for routing $\vec{H}$ in G.*

Proof: Since the congestion of any edge in the Hypercube under this routing scheme is the same, we can calculate the worst case congestion on any edge by calculating the average congestion. Using a symmetry argument, we can consider any fixed terminal z_s and calculate the expected increase in average congestion when sampling a random permutation $\pi : [d] \rightarrow [d]$ and routing all the edges out of z_s in H using π. This expected value will be k times the average congestion, and hence the worst-case congestion of routing $\vec{H}$ in G according to the above routing scheme.

As we noted above, we can define H equivalently as arising from the random process of sampling $u \sim_\rho t$, and routing an infinitesimal fraction of the $\sqrt{d}$ total capacity out of z_t to z_u, and repeating until all of the $\sqrt{d}$ capacity is allocated. We can then calculate the the expected increase in average congestion (under a random permutation π) caused by routing the edges out of z_s as the expected increase in average congestion divided by the total fraction of the $\sqrt{d}$ capacity allocated when we choose the target u from $u \sim_\rho t$. In particular, if we allocated a Δ fraction of the $\sqrt{d}$ capacity, the expected increase in total congestion is just the total capacity that we route multiplied by the length of the path. Of course, the length of this path is just the number of bits in which u and t differ, which in expectation is $\Theta(\sqrt{d})$ by our choice of ρ.

So in this procedure, we allocate $\Delta\sqrt{d}$ total capacity, and the expected increase in total congestion is the total capacity routed $\Delta\sqrt{d}$ times the expected path length $\Theta(\sqrt{d})$. We repeat this procedure $\frac{1}{\Delta}$ times, and so the expected increase in total congestion caused by routing the edges out of z_t in G is $\Theta(d)$. If we perform this procedure for each terminal, the resulting total congestion is $\Theta(kd)$, and because there are $\frac{kd}{2}$ edges in the Hypercube, the average congestion is $\Theta(1)$ which implies that the worst-case congestion on any edge in the Hypercube is also $O(1)$, as desired. Also, the congestion on any edge (z_s, y_s) is 1 because there is a total of $\sqrt{d}$ capacity out of z_s in H, and this is the only flow routed on this edge, which has capacity $\sqrt{d}$ in G by construction. So the worst-case congestion on any edge in the above routing scheme is $O(1)$. $\square$

Corollary 9. *For any $A \subset K$, $h'(A) \leq O(1)h_K(A)$.*

Proof: Consider any set $A \subset K$. Let U be the minimum cut in G separating A from $K-A$. Then the total flow routed from A to $K-A$ in $\vec{H}$ is just $h'(A)$, and if this flow can be routed in G with congestion $O(1)$, this implies that the total capacity crossing the cut from U to $V-U$ is at least $\Omega(1)h'(A)$. And of course the total capacity crossing the cut from U to $V-U$ is just $h_K(A)$ by the definition of U, which implies the corollary. $\square$

So we know that the cuts in H are never too much larger than the corresponding minimum cut in G, and all that remains to show that the quality of H is $o(\sqrt{\log k})$ is to show that the cuts in H are never too small. We conjecture that the quality of H is actually $\Theta(\log^{1/4} k)$, and this seems natural since the quality of H just restricted to the Majority Cut and the sub-cube cuts is actually $\Theta(\log^{1/4} k)$, and often the Boolean functions corresponding to these cuts serve as extremal examples in the harmonic analysis of Boolean functions. In fact, our lower bound on the quality of any cut-sparsifier for G is based only on analyzing these cuts so in a sense, our lower bound is tight given the choice of cuts in G that we used to derive infeasibility in the system of equalities characterizing α-quality cut-sparsifiers.

5.1.3 Harmonic Analysis

We give a brief introduction to the harmonic analysis of Boolean functions, along with formal statements that we will use in the proof of our main theorem in this section.

Consider the group $F_2^d = \{-1,+1\}^d$ equipped with the group operation $s \circ t = [s_1 * t_1, s_2 * t_2, ... s_d * t_d] \in F_2^d$. Any subset $S \subset [d]$ defines a character $\chi_S(x) = \prod_{i \in S} x_i : F_2^d \rightarrow \{-1,+1\}$. See [52] for an introduction to the harmonic analysis of Boolean functions.

Then any function $f : \{-1,+1\}^d \rightarrow \Re$ can be written as:

$$f(x) = \sum_S \hat{f}_S \chi_S(x)$$

Fact 1. *For any $S, T \subset [d]$ s.t. $S \neq T$, $E_x[\chi_S(x)\chi_T(x)] = 0$*

For any $p > 0$, we will denote the p-norm of f as $||f||_p = \Big(E_x[f(x)^p]\Big)^{1/p}$. Then

Theorem 24 (Parseval)**.**

$$\sum_S \hat{f}_S^2 = E_x[f(x)]^2] = ||f||_2^2$$

Definition 28. *Given $-1 \leq \rho \leq 1$, Let $y \sim_\rho x$ denote choosing y depending on x s.t. for each coordinate i, $E[y_i x_i] = \rho$.*

Definition 29. *Given $-1 \leq \rho \leq 1$, the operator T_ρ maps functions on the Boolean cube to functions on the Boolean cube, and for $f : \{-1, +1\}^d \to \Re$, $T_\rho(f(x)) = E_{y \sim_\rho x}[f(y)]$.*

Fact 2. $T_\rho(\chi_S(x)) = \chi_S(x)\rho^{|S|}$

In fact, because T_ρ is a linear operator on functions, we can use the Fourier representation of a function f to easily write the effect of applying the operator T_ρ to the function f:

Corollary 10. $T_\rho(f(x)) = \sum_S \rho^{|S|}\hat{f}_S\chi_S(x)$

Definition 30. *The* Noise Stability *of a function f is $NS_\rho(f) = E_{x, y \sim_\rho x}[f(x)f(y)]$*

Fact 3. $NS_\rho(f) = \sum_S \rho^{|S|}\hat{f}_S^2$

Theorem 25 (Hypercontractivity)**.** *[16] [14] For any $q \geq p \geq 1$, for any $\rho \leq \sqrt{\frac{p-1}{q-1}}$*

$$||T_\rho f||_q \leq ||f||_p$$

A statement of this theorem is given in [52] and [24] for example.

Definition 31. *A function $g : \{-1, +1\}^d \to \Re$ is a j-junta if there is a set $S \subset [d]$ s.t. $|S| \leq j$ and g depends only on variables in S - i.e. for any $x, y \in F_2^d$ s.t. $\forall_{i \in S} x_i = y_i$ we have $g(x) = g(y)$. We will call a function f an (ϵ, j)-junta if there is a function $g : \{-1, +1\}^d \to \Re$ that is a j-junta and $Pr_x[f(x) \neq g(x)] \leq \epsilon$.*

We will use a quantitative version of Bourgain's Junta Theorem [17] that is given by Khot and Naor in [41]:

Theorem 26. *[Bourgain] [17], [41] Let $f\{-1,+1\}^d \rightarrow \{-1,+1\}$ be a Boolean function. Then fix any $\epsilon, \delta \in (0, 1/10)$. Suppose that*

$$\sum_S (1-\epsilon)^{|S|} \hat{f}_S^2 \geq 1 - \delta$$

then for every $\beta > 0$, f is a

$$\Big(2^{c\sqrt{\log 1/\delta \log\log 1/\epsilon}}\Big(\frac{\delta}{\sqrt{\epsilon}} + 4^{1/\epsilon}\sqrt{\beta}\Big), \frac{1}{\epsilon\beta}\Big)\text{-junta}$$

This theorem is often described as mysterious, or deep, and has lead to some breakthrough results in theoretical computer science [41], [**?**] and is also quite subtle. For example, this theorem crucially relies on the property that f is a Boolean function, and in more general cases only much weaker bounds are known [24].

5.1.4 Characterizing Cuts via Noise-Sensitivity

Next we give a simple formula for the size of a cut in H, given the Fourier representation of the cut. So here we consider cuts $A \subset K$ to be Boolean functions of the form $f_A : \{-1,+1\}^d \rightarrow \{-1,+1\}$ s.t. $f_A(s) = +1$ iff $z_s \in A$.

Lemma 14. $h'(A) = k\frac{\sqrt{d}}{2}\frac{1-NS_\rho[f_A(x)]}{2}$

Proof: We can again use the infinitesimal characterization for H, in which we choose $u \sim_\rho t$ and allocate Δ units of capacity from z_s to z_t and repeat until all $\sqrt{d}$ units of capacity are spent.

If we instead choose z_s uniformly at random, and then choose $u \sim_\rho t$ and allocate Δ units of capacity from z_s to z_t, and repeat this procedure until all $k\frac{\sqrt{d}}{2}$ units of capacity are spent, then at each step the expected contribution to the cut is exactly $\Delta\frac{1-NS_\rho[f_A(x)]}{2}$ because $\frac{1-NS_\rho[f_A(x)]}{2}$ is exactly the probability that if we choose t uniformly at random, and $u \sim_\rho t$ that $f_A(u) \neq f_A(t)$ which means that this edge contributes to the cut. We repeat this procedure $\frac{k\sqrt{d}}{2\Delta}$ times, so this implies the lemma. $\square$

Lemma 15. $h'(A) = \Theta\Big(k \sum_S \hat{f}_S^2 \min(|S|, \sqrt{d})\Big)$

Proof: Using the setting $\rho = 1 - \frac{1}{\sqrt{d}}$, we can compute $h'(A)$ using the above lemma:

$$h'(A) = k\frac{\sqrt{d}}{4}(1 - NS_\rho[f_A(x)])$$

And using Parseval's Theorem, $\sum_S \hat{f}_S^2 = ||f||_2 = 1$, so we can replace 1 with $\sum_S \hat{f}_S^2$ in the above equation and this implies

$$h'(A) = k\frac{\sqrt{d}}{4}\sum_S \hat{f}_S^2(1 - (1 - \frac{1}{\sqrt{d}})^{|S|})$$

Consider the term $(1-(1-\frac{1}{\sqrt{d}})^{|S|})$. For $|S| \leq \sqrt{d}$, this term is $\Theta(\frac{|S|}{\sqrt{d}})$, and if $|S| \geq \sqrt{d}$, this term is $\Theta(1)$. So this implies

$$h'(A) = \Theta\Big(k\sum_S \hat{f}_S^2 \min(|S|, \sqrt{d})\Big)$$

$\square$

The edge-isoperimetric constant of the Hypercube is 1, but on subsets of the cube that are imbalanced, the Hypercube expands more than this.

Definition 32. *For a given set $A \subset \{-1,+1\}^{[d]}$, we define $bal(A) = \frac{1}{k}\min(|A|, k - |A|)$ as the balance of the set A.*

Given any set $A \subset \{-1,+1\}^{[d]}$ of balance $b = bal(A)$, the number of edges crossing the cut $(A, \{-1,+1\}^{[d]} - A)$ in the Hypercube is $\Omega(bk\log\frac{1}{b})$. So the Hypercube expands better on small sets, and we will prove a similar small set expansion result for the cut-sparsifier H. In fact, for any set $A \subset K$ (which we will associated with a subset of $\{-1,+1\}^{[d]}$ and abuse notation), $h'(A) \geq bal(A)k\Omega(\min(\log\frac{1}{bal(A)}, \sqrt{d}))$. We will prove this result using the Hypercontractive Inequality.

Lemma 16. $h'(A) \geq bal(A)k\Omega(\min(\log\frac{1}{bal(A)}, \sqrt{d}))$

Proof: Assume that $|A| \leq |\{-1,+1\}^{[d]} - A|$ without loss of generality. Throughout just this proof, we will use the notation that $f_A : \{-1,+1\}^d \to \{0,1\}$ and $f_A(s) = 1$ iff $s \in A$. Also we will denote $b = bal(A)$.

Let $\gamma << 1$ be chose later. Then we will invoke the Hypercontractive inequality with $q = 2$, $p = 2 - \gamma$, and $\rho = \sqrt{\frac{p-1}{q-1}} = \sqrt{1-\gamma}$. Then

$$||f||_p = E_x[f(x)^p]^{1/p} = b^{1/p} \approx b^{1/2(1+\gamma/2)}$$

Also $||T_\rho(f(x))||_q = ||T_\rho(f(x))||_2 = \sqrt{\sum_S \rho^{2|S|}\hat{f}_S^2}$. So the Hypercontractive Inequality implies

$$\sum_S \rho^{2|S|}\hat{f}_S^2 \leq b^{1+\gamma/2} = be^{-\frac{\gamma}{2}\ln\frac{1}{b}}$$

And $\rho^{2|S|} = (1-\gamma)^{|S|}$. Using Parseval's Theorem, $\sum_S \hat{f}_S^2 = ||f||_2^2 = b$, and so we can re-write the above inequality as

$$\begin{aligned} b - be^{\frac{-\gamma}{2}\ln\frac{1}{b}} \leq b - \sum_S (1-\gamma)^{|S|}\hat{f}_S^2 \leq b^{1-\gamma/2} &= \sum_S \hat{f}_S^2(1-(1-\gamma)^{|S|}) \\ &= \Theta\Big(\sum_S \hat{f}_S^2 \gamma \min(|S|, \frac{1}{\gamma})\Big) \end{aligned}$$

This implies

$$\frac{b}{\gamma}(1 - e^{-\frac{\gamma}{2}\ln\frac{1}{b}}) \leq \Theta\Big(\sum_S \hat{f}_S^2 \min(|S|, \frac{1}{\gamma})\Big)$$

And as long as $\frac{1}{\gamma} \leq \sqrt{d}$,

$$\sum_S \hat{f}_S^2 \min(|S|, \frac{1}{\gamma}) \leq \frac{1}{k}O(h'(A)) = O\Big(\sum_S \hat{f}_S^2 \min(|S|, \sqrt{d})\Big)$$

If $\frac{\gamma}{2}\ln\frac{1}{b} \leq 1$, then $e^{-\frac{\gamma}{2}\ln\frac{1}{b}} = 1 - \Omega(\frac{\gamma}{2}\ln\frac{1}{b})$ which implies

$$\frac{b}{\gamma}(1 - e^{-\frac{\gamma}{2}\ln\frac{1}{b}}) \geq \Omega(b\ln\frac{1}{b})$$

However if $\ln\frac{1}{b} = \Omega(\sqrt{d})$, then we cannot choose γ to be small enough (we must choose $\frac{1}{\gamma} \leq \sqrt{d}$) in order to make $\frac{\gamma}{2}\ln\frac{1}{b}$ small.

So the only remaining case is when $\ln\frac{1}{b} = \Omega(\sqrt{d})$. Then notice that the quantity $(1-$

$e^{-\frac{\gamma}{2}\ln\frac{1}{b}})$ is increasing with decreasing b. So we can lower bound this term by substituting $b = e^{-\Theta(\sqrt{d})}$. If we choose $\gamma = \frac{1}{\sqrt{d}}$ then this implies

$$\frac{1}{\gamma}(1 - e^{-\frac{\gamma}{2}\ln\frac{1}{b}}) = \Omega(\sqrt{d})$$

And this in turn implies that

$$\frac{b}{\gamma}(1 - e^{-\frac{\gamma}{2}\ln\frac{1}{b}}) = \Omega(b\sqrt{d})$$

which yields $h'(A) \geq \Omega(bk\sqrt{d})$. So in either case, $h'(A)$ is lower bounded by either $\Omega(bk\sqrt{d})$ or $\Omega(bk\ln\frac{1}{b})$, as desired.

□

5.1.5 Beating the Contraction Bound

Here we show that the quality of the cut-sparsifier H is $o(\sqrt{\log k})$, thus beating how well the best distribution on 0-extensions can approximate cuts in G by a super-constant factor.

We will first give an outline of how we intend to combine Bourgain's Junta Theorem, and the small set expansion of H in order to yield this result. In a previous section, we gave a Fourier theoretic characterization of the cut function of H. We will consider an arbitrary cut $A \subset K$ and assume for simplicity that $|A| \leq |K - A|$. If the Boolean function f_A that corresponds to this cut has significant mass at the tail of the spectrum, this will imply (by our Fourier theoretic characterization of the cut function) that the capacity of the corresponding cut in H is $\omega(k)$. Every cut in G has capacity at most $O(k\sqrt{d})$ because we can just cut every edge (z_s, y_s) for each terminal $z_s \in A$, and each such edge has capacity $\sqrt{d}$. Then in this case, the ratio of the minimum cut in G to the corresponding cut in H is $o(\sqrt{d})$.

But if the tail of the Fourier spectrum of f_A is not significant, and applying Bourgain's Junta Theorem implies that the function f_A is close to a junta. Any junta will have a small cut in G (we can take axis cuts corresponding to each variable in the junta) and so for any function that is different from a junta on a vanishing fraction of the inputs, we will be able to construct a cut in G (not necessarily minimum) that has capacity $o(k\sqrt{d})$. On all balanced cuts (i.e. $|A| = \Theta(k)$), the capacity of the cut in H will be $\Omega(k)$, so again in this case the ratio of the minimum cut in G to the corresponding cut in H is $o(\sqrt{d})$.

So the only remaining case is when $|A| = o(k)$, and from the small set expansion of H the capacity of the cut in H is $\omega(|A|)$ because the cut is imbalanced. Yet the minimum cut in G is again at most $|A|\sqrt{d}$, so in this case as well the ratio of the minimum cut in G to the corresponding cut in H is $o(\sqrt{d})$.

Theorem 27. *There is an infinite family of graphs for which the quality of the best cut-sparsifier is $\Omega(\frac{\log^2 \log\log k}{\log\log\log\log k})$ better than the quality of the best cut-sparsifier that can be realized as a convex combination of 0-extensions.*

In Bourgain's Junta Theorem 26, we will choose:

$\frac{1}{\epsilon} = \frac{1}{32}\log d$

$\frac{1}{\beta} = d^{1/4}$

$\frac{1}{\delta'} = \log^{2/3} d$

And also let $b = bal(A) = \frac{|A|}{k}$, and remember for simplicity we have assumed that $|A| \leq |K - A|$, so $b \leq \frac{1}{2}$.

$\delta = \delta' b$

Lemma 17. *If $\sum_S (1-\epsilon)^{|S|}\hat{f}_S^2 \leq 1 - \delta$ then this implies $\sum_S \hat{f}_S^2 \min(|S|, \sqrt{d}) \geq \Omega\left(\frac{\delta}{\epsilon}\right) = \Omega(b \log^{1/3} d)$*

Proof: The condition $\sum_S (1-\epsilon)^{|S|}\hat{f}_S^2 \geq 1 - \delta$ implies $\delta \leq 1 - \sum_S (1-\epsilon)^{|S|}\hat{f}_S^2 = O(\sum_S \hat{f}_S^2 \min(|S|\epsilon, 1))$ and rearranging terms this implies

$$\frac{\delta}{\epsilon} \leq O(\sum_S \hat{f}_S^2 \min(|S|, \frac{1}{\epsilon})) = O(\sum_S \hat{f}_S^2 \min(|S|, \sqrt{d}))$$

where the last line follows because $\frac{1}{\epsilon} = O(\log d) \leq O(\sqrt{d})$. $\square$

So combining this lemma and Lemma 15: if the conditions of Bourgain's Junta Theorem are not met, then the capacity of the cut in the sparsifier is $\Omega(kb \log^{1/3} d)$. And of course, the capacity of the minimum cut in G is at most $kb\sqrt{d}$, because for each $z_s \in A$ we could separate A from $K - A$ by cutting the edge (z_s, y_s), each of which has capacity $\sqrt{d}$.

Case 1

If the conditions of Bourgain's Junta Theorem are not met, then the ratio of the minimum cut in G separating A from $K - A$ to the corresponding cut in H is at most $O(\frac{\sqrt{d}}{\log^{1/3} d})$.

What if the conditions of Bourgain's Junta Theorem are met? We can check what Bourgain's Junta Theorem implies for the given choice of parameters. First consider the case in which b is not too small. For our choice of parameters the following three inequalities hold:

$$2^{c\sqrt{\log 1/\delta \log\log 1/\epsilon}} \leq \log^{1/24} d \tag{5.1}$$

$$\frac{\delta}{\sqrt{\epsilon}} \geq 4^{1/\epsilon}\sqrt{\beta} \tag{5.2}$$

$$\sqrt{d} b \log^{-1/8} d \geq d^{1/4} \log d \tag{5.3}$$

Claim 20. *If (5.2) is true,* $\left(\frac{\delta}{\sqrt{\epsilon}} + 4^{1/\epsilon}\sqrt{\beta}\right) = O\left(b \log^{-1/6} d\right)$

Claim 21. *If (5.1) and (5.2) are true,* $2^{c\sqrt{\log 1/\delta \log\log 1/\epsilon}}\left(\frac{\delta}{\sqrt{\epsilon}} + 4^{1/\epsilon}\sqrt{\beta}\right) = O\left(b \log^{-1/8} d\right)$

Applying Bourgain's Junta Theorem, if the conditions are met (for our given choice of parameters), we obtain that f_A is an $\left(O\left(b \log^{-1/8} d\right), O(d^{1/4}\log d)\right)$-junta.

Lemma 18. *If f_A is a (ν, j)-junta, then* $h_K(A) \leq k\nu\sqrt{d} + j\frac{k}{2}$

Proof: Let g be a j-junta s.t. $Pr_x[f_A(x) \neq g(x)] \leq \nu$. Then we can disconnect the set of nodes on the Hypercube where g takes a value $+1$ from the set of nodes where g takes a value -1 by performing an axis cut for each variable that g depends on. Each such axis cut, cuts $\frac{k}{2}$ edges in the Hypercube, so the total cost of cutting these edges is $j\frac{k}{2}$ and then we can alternatively cut the edge (z_s, y_s) for any s s.t. $f_A(s) \neq g(s)$, and this will be a cut separating A from $K - A$ and these extra edges cut are each capacity $\sqrt{d}$ and we cut at most νk of these edges in total. $\square$

So if f_A is an $\left(O\left(b \log^{-1/8} d\right), O(d^{1/4}\log d)\right)$-junta and (5.3) holds, then $h_K(A) \leq O\left(\frac{kb\sqrt{d}}{\log^{1/8} d}\right)$.

Case 2

Suppose the conditions of Bourgain's Junta Theorem are met, and (5.1), (5.2) and (5.3) are true, then the ratio of the minimum cut in G separating A from $K - A$ to the corresponding cut in H is at most $O(\frac{\sqrt{d}}{\log^{1/8} d})$.

Proof: Lemma 16 also implies that the edge expansion of H is $\Omega(1)$, so given a cut $|A|$, $h'(A) \geq \Omega(|A|) = \Omega(kb)$. Yet under the conditions of this case, the capacity of the cut in G is $O\left(\frac{kb\sqrt{d}}{\log^{1/8} k}\right)$ and this implies the statement. $\square$

The only remaining case is when the conditions of Bourgain's Junta Theorem are met at least one of the three conditions is not true. We can apply Lemma 16 directly to get that in this case $h'(A) = \omega(|A|)$ and of course $h_K(A) \leq |A|\sqrt{d}$.

Case 3

Suppose the conditions of Bourgain's Junta Theorem are met, and at least one of the the inequalities is not true, then the ratio of the minimum cut in G separating A from $K - A$ to the corresponding cut in H is at most $O(\frac{\sqrt{d}\log\log\log d}{\log^2\log d})$.

Proof: If (5.1) is false, $\log(1/\delta')+\log(1/b) = \log(1/\delta) > \frac{(\log\log^{1/30} d/c)^2}{\log\log 1/\epsilon} = \Omega\left(\frac{\log^2\log d}{\log\log\log d}\right)$. Since $1/\delta' = O(\log\log d)$, it must be the case that $\log(1/b) = \Omega\left(\frac{\log^2\log d}{\log\log\log d}\right)$.

If (5.2) is false, $b < \frac{4^{1/\epsilon}\sqrt{\beta}\sqrt{\epsilon}}{\delta'} = O(d^{-1/8}\log^{1/6} d)$, and $\log(1/b) = \Omega(\log d)$.

If (5.3) is false, $b < d^{-1/4}\log^{9/8} d$ and $\log(1/b) = \Omega(\log d)$.

The minimum of the 3 bounds is the first one. So, $\log(1/b) = \Omega\left(\frac{\log^2\log d}{\log\log\log d}\right)$ if at least 1 of the 3 conditions is false. Applying Lemma 16, we get that $h'(A) \geq \Omega(|A|\log\frac{1}{b}) = \Omega(|A|\frac{\log^2\log d}{\log\log\log d})$. And yet $h_K(A) \leq |A|\sqrt{d}$, and this implies the statement. Combining the cases, this implies that the quality of H is $O(\frac{\sqrt{d}\log\log\log d}{\log^2\log d})$. $\square$

Conjecture 1. *The quality of H as a cut-sparsifier for G is $O(d^{1/4}) = O(\log^{1/4} k)$*

We believe that establishing this conjecture would require quantitatively stronger version of the "Majority Is Stablest" Theorem [50] in the regime in which the probability of flipping a bit is $o(1)$.

5.2 Separation Between Flow and Cut Sparsification

In Section 3.2.2 we proved that flow-sparsification is no easier than cut-sparsification: any α-quality flow-sparsifier is also an α-quality cut-sparsifier. Here we prove that flow-sparsification can be asymptotically harder than cut-sparsification. We given an example in which the quality of the best cut-sparsifier is $O(1)$, yet the quality of the best flow-sparsifier is $\Omega(\log \log k)$. The results in this section appear in [44].

5.2.1 The Graph G

Here we define the graph $G = (V, E)$ and the set $K \subset V$, and highlight the properties of G that will be used in proving an $\Omega(\log \log k)$ for the quality of the best flow-sparsifier. We also give an $O(1)$-quality cut-sparsifier.

G is generated from another graph $G' = (K, E')$. Set $x = \log^{1/3} n$ and let $k = \frac{n}{x}$. We choose $G' = (K, E')$ to be an x-regular graph that has

$$girth(G') \geq \Omega(\frac{\log k}{\log x}) = \Omega(\frac{\log n}{\log \log n})$$

For each terminal $a \in K$, we denote the set of neighbors of a in G' as $\Gamma(a)$. Additionally, order this set arbitrarily (for each node a), and let $\Gamma^i(a)$ be the i^{th} neighbor of a in G'. This labeling scheme need not be consistent, and terminal a can label a neighbor b as the i^{th} neighbor ($b = \Gamma^i(a)$) but b labels a as the j^{th} neighbor ($a = \Gamma^j(b)$).

We construct $G = (V, E)$ from G': For each terminal $a \in K$, construct a path $P_a = a, a_1, a_2, ...a_x$ in G. And for each edge $(a, b) \in G'$, suppose that $b = \Gamma^i(a), a = \Gamma^j(b)$, add an edge in G connecting a_i and b_j. See Figure 5-1.

Claim 22. *Any matching M in G' can be routed in G - i.e. $cong_G(\vec{M}) \leq 1$*

Lemma 19. *$G = (V, E)$ has an $O(1)$-quality cut-sparsifier*

Proof: Let $H = (K, E_H)$ be a complete, capacitated graph in which each edge has capacity $\frac{2}{k}$. For any $A \subset K$ (and $|A| \leq k/2$), the minimum cut in G separating A from $K - A$ is at most $|A|$ because for each $u \in A$, if we cut the edge (u, u_1) (on the path P_u), then there

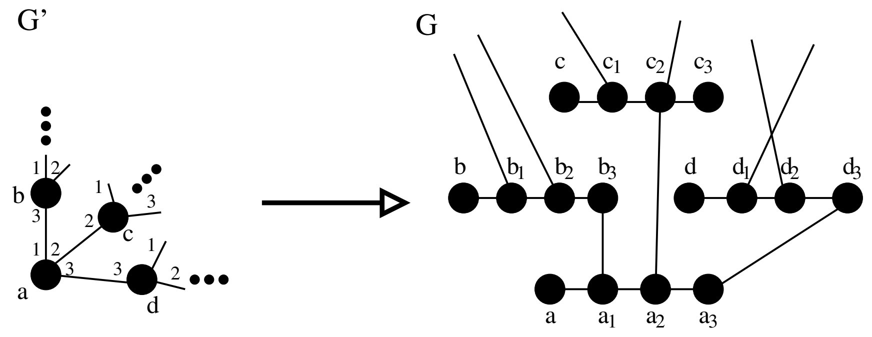

Figure 5-1: Constructing the graph $G = (V, E)$ from an x-regular, high-girth expander $G' = (K, E')$

is no path in G from A to $K - A$. Also, for any $A \subset K$ (again with $|A| \leq k/2$), the total capacity crossing the cut $(A, K - A)$ in H is $|A||K - A|\frac{2}{k} \geq |A|\frac{k}{2} \times \frac{2}{k} = |A|$ and hence H is a cut-sparsifier.

Additionally, we can partition the edges in H into a set of weighted matchings, where the total weight is constant. Hence we can apply Claim 19 and this implies that $\vec{H}$ can be routed in G with constant congestion. Using the results in Section 3.2.2, we conclude that H has $O(1)$-quality as a cut-sparsifier. $\square$

Throughout the remainder of this section, we will consider only flow-sparsifiers and when we use the term "quality" in reference to some candidate vertex sparsifier H, we will mean the quality of H as a flow-sparsifier.

Let H be a flow-sparsifier for G and K.

Definition 33. *Let C_G be the total capacity in G and let $\Delta(G)$ be the maximum degree of G. Also let C_H be the total capacity in H.*

Definition 34. *Let $dist_G(u, v)$ and $dist_{G'}(u, v)$ be the natural shortest path distances on G, G' respectively*

We will need a number of properties of G in order to prove an $\Omega(\log \log k)$ lower bound:

1. $C_G \leq 2n$

2. $\Delta(G) = 3$

3. $\forall_{u,v \in K} dist_G(u,v) \geq dist_{G'}(u,v)$

4. $girth(G) \geq girth(G')$

We will first give some observations that are the foundation of the proof: Suppose there is a flow-sparsifier H that has $O(1)$-quality. This implies that $cong_G(\vec{H}) = O(1)$, and we can enumerate some properties of H. These are the properties that we will use to derive a contradiction (and in fact get a quantitative bound $cong_G(\vec{H}) \geq \Omega(\log \log k)$):

1. The total capacity in H must be $O(k)$

   If H had $\omega(k)$ total capacity, then there would be some terminal $a \in K$ that is incident to $\omega(1)$ total capacity in H. Yet the minimum cut in G separating a from $K - \{a\}$ is constant (G is a constant degree graph), so this would immediately yield a super-constant lower bound on the congestion of embedding H into G.

2. Most capacity in H must be between terminals for which the distance is much smaller than the girth of G

   Suppose, for example, that H is a complete graph with uniform edge capacities of $\frac{1}{k}$. Then the average distance (as measured in G) between two random terminals is roughly $\Omega(girth(G'))$. So if we choose the natural shortest path distance in G as the oblivious dual certificate, then

   $$\sum_{a,b \in K} c_H(a,b) d(a,b) \geq \Omega(girth(G')k)$$

   Yet $\sum_{(u,v) \in E} c(u,v) d(u,v) = \Theta(n)$ and this would yield a super-constant lower bound on the congestion of embedding H into G.

3. Fix the optimal embedding of H into G. Then at most $o(k)$ of the total capacity in H can be routed (in the optimal embedding) along high-girth paths.

   Suppose, for example, that $\Omega(k)$ total capacity in H is routed in G using high-girth paths. The total capacity in G is $\Theta(n)$, yet each unit of capacity in H that is routed along a high-girth path uses up $\Omega(girth(G))$ total capacity in G and $\Omega(girth(G)k)$

total capacity is consumed, but this is $\omega(n)$. So in fact, for most edges (a, b) in H, most of the $c_H(a, b)$ flow in the optimal embedding must traverse a path much shorter than the girth of G, so this flow must traverse the natural shortest path between a and b.

Additionally, the requirement that H can route any matching in G' places constraints on the local neighborhood of a terminal.

A There are $\omega(1)$ neighbors of a in G', and a can route a unit flow to each such neighbor using edges in the graph H

   H cannot hope to support such flows via a direct edge, because this would require a to be incident to $\omega(1)$ total capacity in H. So the paths that support these flows must be indirect.

B In some sense, on average the path used to route a unit flow from a to b in H must be constant length (using an argument similar to Condition 2 above)

   If, on average, the length of any such path were $\omega(1)$, then if when we route a matching in G' in H, there are $\Theta(k)$ total edges in the matching and $\Theta(k)$ total units of flow. But each unit of flow would use up $\omega(1)$ total capacity in H, because the paths on average are $\omega(1)$ length in H.

C So a must be able to send a unit flow to a super-constant number of terminals, using constant length paths. Additionally these paths must not reach any terminal b that is too far from a in G.

So the local neighborhood of a must be able to support all these requirements - a must be able to send a unit of flow to a super-constant number of terminals, using constant length paths that remain within the local neighborhood. Yet we know how most of the edges in the local neighborhood of H embed (in the optimal embedding) into G, because these edges mostly use the natural shortest path in G. This shortest path traverses a sub-interval of P_a. So we can derive a contradiction, because any local neighborhood that can support all these requirements must have large cut-width, and so the natural embedding of this neighborhood would incur super-constant congestion on some sub-interval of P_a.

Roughly, the outline of the argument is the same as in the previous section: we use oblivious dual certificates to establish local constraints on how the neighborhood of any terminal a. We can piece these local constraints together (using the fact that all matchings in G' can be routed in H) to reduce this question to a more structured question about cut-width, and from this we derive a contradiction (Section 5.2.4).

5.2.2 Capacity Surgery

Let $H = (K, E_H)$ be an arbitrary flow-sparsifier. We will argue that if all matchings M in G' can be routed in H with congestion at most 1, then $cong_G(\vec{H}) = \Omega(\log \log k)$. We do not assume anything structurally about H, so there are many different ways in which the demand $\vec{H}$ can result in super-constant congestion. We will need a complex argument to handle this, and ultimately we will reduce computing a lower bound on $cong_G(\vec{H})$ to a more structured embedding question about cut-width.

Assume $c_1 > 32 \times c_2$ and $c_2 > 24$

Given H, we fix the optimal embedding of H into G. This is a min-congestion routing of $\vec{H}$ in G. Each edge $(u, v) \in E_H$ is mapped to a flow in G from u to v of value $c_H(u, v)$. We can decompose such a flow from u to v in G into paths. We refer to such a decomposition as a path decomposition, and we assume that this decomposition uses only simple paths.

Definition 35. *We call an edge $(u, v) \in E_H$ girth-routed if at least $\frac{c_H(u,v)}{2}$ flow (of the $c_H(u, v)$ total flow) from u to v is routed using paths (in G) of length at least $\frac{girth(G')}{4}$.*

Definition 36. *Let $H' = (K, E(H'))$ be the graph defined by deleting all girth-routed edges in H. Let C_{gr} be the total capacity of girth-routed edges in H - i.e. the total capacity deleted from H to obtain H'.*

If H $O(1)$-approximates the congestion of all multicommodity flows in G, then the total capacity in H must be $O(k)$. This is true because each node $u \in K$ has degree 1 in G, and so if the total capacity in H is $\omega(k)$ then there is a node $u \in K$ which is incident to $\omega(1)$ total units of capacity, and even just routing the demands in $\vec{H}$ incident to u would incur $\omega(1)$ congestion in G.

Given this, we know that if H $O(1)$-approximates the congestion of all multicommodity flows in G, then on average edges (u, v) in G' must be able to route a unit flow from u to v using constant length paths. Otherwise we could construct a matching M in which all edges need super-constant length paths to route a unit flow in H, and this flow would not be feasible in H because there would be $\Omega(k)$ pairs each using $\omega(1)$ units of capacity - but H has only $O(k)$ total capacity.

We need to formalize this notion, that edges in G' must on average use short paths (and in fact a stronger condition, that edges in G' must also not route flow in H that reaches nodes that are too far away according to the shortest path metric in G'). So we set $D = \log^{2/3} n$.

Definition 37. *We call a path P connecting u and v (in H or H') good if P is length $< \frac{1}{c_2} \log\log n$ and reaches no terminal a for which either $dist_{G'}(u, a)$ or $dist_{G'}(v, a) \geq D$.*

We call an edge $(u, v) \in E'$ good if u can send at least $\frac{1}{2}$ unit of flow to v using only good paths in H', and otherwise we call edge (u, v) bad.

Note that the definition of a good edge is with respect to flows in H', not H. We need this for a technical reason.

Case 1

Suppose that $C_H \geq \frac{k}{c_1} \log\log n$

Then there is a node $u \in K$ that has at least $\frac{2}{c_1} \log\log n$ total capacity incident in H. And because v is degree one in G, this implies that $cong_G(\vec{H}) \geq \frac{2}{c_1} \log\log n$ so we can conclude that $C_H \leq \frac{k}{c_1} \log\log n$.

Case 2

Suppose that $C_{gr} \geq \frac{k}{\log^{1/3} n}$.

Then, in the optimal embedding of H into G there is at least $\frac{C_{gr}}{2} \geq \frac{k}{2\log^{1/3} n}$ total capacity that uses paths of length at least $\frac{girth(G')}{4} = \Omega(\log n / \log\log n)$. So the total capacity used in routing H into G is $\Omega(n \log^{1/3} n / \log\log n)$. And $C_G \leq 2n$ so this implies:

$$cong_G(\vec{H}) \geq \Omega(\frac{\log^{1/3} n}{\log\log n})$$

So we can conclude that $C_{gr} \leq \frac{k}{\log^{1/3} n}$

5.2.3 Matchings

We will use the following elementary lemma to simplify the case analysis:

Lemma 20. *Let G' be an x-regular graph on k terminals. Let $F \subset E'$, s.t. $|F| \geq ckx$. Then there is a matching $M \subset F$ s.t. $|M| \geq \frac{c}{2}k$*

Proof: Let M be a maximal matching using only edges in F. Then let A be the set of terminals that are matched in M, and let B be the remaining terminals. No edge in F can have both endpoints in B, because otherwise M would not be maximal. So all edges in F are adjacent to a terminal in A. Because G' is x-regular, the number of edges in E' adjacent to some terminal in A is at most $|A|x$, and this must be at least the size of F. So $|A|x \geq ckx$ In particular, $|A| \geq ck$, and every terminal in A is matched in M so there are at least $\frac{c}{2}k$ edges in M. □

Case 3

Suppose that there are at least $\frac{kx}{4}$ bad edges (in G'). We let F be the set of bad edges in G'. $|F| \geq \frac{kx}{4}$. We can use the above lemma to find a matching $M \subset F$ so that $|M| = \frac{k}{8}$. M is a matching, so $\vec{M}$ is routable with congestion at most 1 in G, and so must also be routable with congestion at most 1 in H.

For each edge $(u, v) \in M$, set $y_{u,v} = 1$. Then using Case 2, $C_{gr} \leq \frac{k}{\log^{1/3} n} = o(k)$. Deleting any edge $(a, b) \in E_H$ of capacity $c_H(a, b)$ affects at most $c_H(a, b)$ units of flow in the routing of $\vec{M}$ in H because $\vec{M}$ can be routed in H with congestion at most 1. So we can set $y'_{u,v}$ as the amount of flow remaining using the same routing scheme for $\vec{M}$ in H' as for $\vec{M}$ in H, just deleting paths that traverse some girth-routed edge in H (which is deleted in order to obtain H').

$|M| = \frac{k}{8}$ and so $\sum_{u,v} y_{u,v} = \frac{k}{8}$. Because $C_{gr} \leq o(k)$, $\sum_{u,v} y'_{u,v} = \frac{k}{8} - o(k)$. Let $\vec{f'}(M)$ be the flow in H' that satisfies $y'_{u,v}$ that results from deleting all flow paths that traversed a girth-routed edge in H. Each edge $(u, v) \in M$ is bad, so if we decompose $\vec{f'}(M)$ into paths, then for any $(u, v) \in M$ at most $\frac{1}{2}$ unit of flow is carried by good paths. So the total flow carried by bad paths is at least

$$\sum_{u,v} y'_{u,v} - \frac{1}{2}|M| \geq \frac{k}{8} - o(k) - \frac{1}{2}|M| = \frac{k}{16} - o(k)$$

So there must be $\frac{k}{16}$ total units of flow on bad paths in H', and in particular there must be at least $\frac{k}{32}$ total units of flow on either paths of length at least $\frac{1}{c_2}\log\log n$ or at least $\frac{k}{32}$ total units of flow on paths that reach nodes a that are at least distance D from one of the two endpoints u or v of the flow. We consider these two sub-cases separately:

Case 3a

Suppose that there are $\frac{k}{32}$ total units of flow on paths in H' of length at least $\frac{1}{c_2}\log\log n$. Notice that the total capacity in H', $C_{H'}$ is at most the total capacity in H, C_H (because H' resulted from H by deleting girth-routed edges). And the total capacity in H' is at least $\frac{k}{32\times c_2}\log\log n$. But then

$$C_{H'} \geq \frac{k}{32 \times c_2}\log\log n > \frac{k}{c_1}\log\log n \geq C_H$$

using Case 1, and we have a contradiction

Case 3b

Suppose that there are $\frac{k}{32}$ total units of flow on paths in H' that reach a node a that is distance at least D from one of the two endpoints u or v of the path.

Consider any such path, $P_{u,v}$ in H' that is carrying λ units of flow from $y'_{u,v}$. Suppose that a is the first node such that $dist_{G'}(u,a) \geq D$ or $dist_{G'}(v,a) \geq D$. Assume without loss of generality that $dist_{G'}(u,a) \geq D$. Construct a demand vector $\vec{r} \in \Re^{\binom{k}{2}}$ as follows: For each such path $P_{u,v}$, increment the demand $\vec{r}_{u,a}$ by λ.

So for each flow path carrying λ units of flow, we have added λ demand to some coordinate in $\vec{r}$.

Claim 23. *$\vec{r}$ is routable in H with congestion at most* 1

Proof: We constructed $\vec{r}$ demands based on a flow $\vec{f'}(M)$ of congestion at most 1 in H', and the same routing scheme for $\vec{f'}(M)$ that achieved congestion at most 1 in H' also achieves congestion at most 1 in H. We then constructed demands by deleting segments of the flow paths - i.e. in the above example we preserved only the sub-path of $P_{u,v}$ from u to a and deleted the rest of the path. This flow resulting from deleting sub-paths still has congestion 1 in H and also satisfies the demands $\vec{r}$. $\square$

Notice that for every path carrying λ flow we have added a demand equal to this flow value λ between two nodes $u, a \in K$ which are distance at least D apart in G' - i.e. $dist_{G'}(u, a) \geq D$.

Consider the minimum congestion embedding of the demand vector $\vec{r}$ into G. We can use the natural shortest path metric on G to certify $\vec{r}$ has large congestion in G: for each edge in G, place a unit of distance on this edge.

The total distance $\times$ capacity units used is $C_G \leq 2n$. And we note that $dist_G(u, v) \geq dist_{G'}(u, v)$ for all $u, v \in K$ from the claim Section 6.1. So $\sum_{u,v} \vec{r}_{u,v} dist_G(u, v) \geq \frac{k}{32} D$.

This implies $cong_G(\vec{H}) \geq cong_G(\vec{r}) \geq \frac{\Omega(k)D}{2n} = \Omega(\log^{1/3} n)$. So we can conclude using Case 3a and 3b that there are at most $\frac{kx}{4}$ bad edges in G'.

5.2.4 Cut-Width

So we can remove all bad edges in G' from consideration, and there are at least $\frac{kx}{4}$ good edges remaining. This implies that there is a terminal $a \in G'$ that is incident to at least $\frac{x}{2}$ good edges. Consider such a terminal a.

Let $\Gamma_{good}(a)$ be the set of neighbors u of a for which the edge $(u, a) \in E'$ is good. We can define the induced path P_{good} as the path on just Γ_{good} in the order in which these nodes are labeled according to Γ. For each terminal b adjacent to a in G', we define a component C_j (assuming $b = \Gamma^j(a)$) that consists of all terminals that are distance at most D according to G' from terminal a and are in the depth-first search sub-tree (of G') rooted at terminal b. For each terminal b adjacent to a s.t. the edge (a, b) is bad, if $b = \Gamma^j(a)$ then choose the closest terminal c on the path P_a for which the edge (c, a) is good, and break ties arbitrarily. For each node $c \in \Gamma_{good}$, consider the set of terminals d that chose c, and set $C'_i = \cup_{d \text{ chooses } c} C_j$ if $\Gamma^i(u) = c$ and $\Gamma^j(u) = d$, and by definition any good terminal on the path P_a chooses itself.

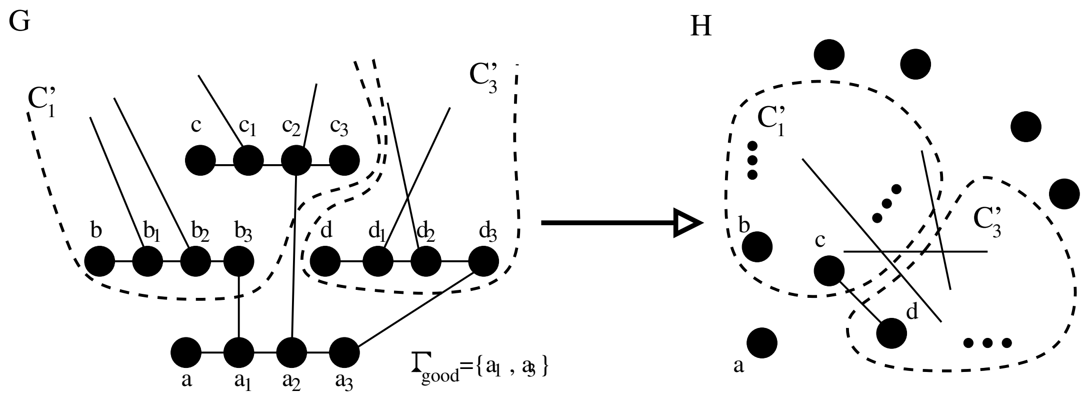

Figure 5-2: Constructing the partition $\{C'_i\}$ of terminals at distance at most D

So C'_i forms a partition of the set of terminals that are distance at most D (since $girth(G') >> D$) from terminal a in G', and for each component C'_i there is a terminal $c \in C'_i$ that is adjacent to a in G' and for which (a, c) is a good edge.

We construct a graph B in which we add a node for a, and a node i for each component C'_i. For each edge in H' from a node C'_i to a node in C'_j, we add an edge in B of the same capacity connecting the node i to the node j (and we allow parallel edges). See Figure 5-2.

Intuitively, in B we allow routing within C'_i for free. But because each node i in B contains a terminal c which is a neighbor of a and for which (a, c) is a good edge, terminal a must be able to send at least $\frac{1}{2}$ units of flow from a to terminal c in H' using paths that do not reach any other node distance more than D from terminal a (according to $dist_{G'}$) and using only paths of length at most $\frac{1}{c_2} \log \log n$. Because in B we allow routing within C'_i for free, and have only removed nodes that are too far from a to be on any good path, in the graph B we must be able to send at least $\frac{1}{2}$ units of flow of flow from a to i using paths of length at most $\frac{1}{c_2} \log \log n$.

So consider the path P_{good} defined on $\{a\} \cup \Gamma_{good}(a)$ in which nodes in Γ_{good} are visited according to the order defined by $\Gamma(a)$. We will prove a lower bound on the minimum cut-width embedding of B into P_{good}. This will imply a cut-width lower bound on embedding B into P_a, and because every edge in H' routes at least $\frac{1}{2}$ of its capacity through paths of length at most $\frac{girth(G')}{4}$ (and consequently every edge in B connecting a node i to a node j must traverse the sub-path $a_i, a_{i+1}, ...a_j$ in P_a), and so this implies a lower bound on the congestion of embedding H' into G. So consider an interval I of P_{good}:

Definition 38. *Given a set of flows out of* a*, we define the average path length as the sum over all flow paths* P *(in a path decomposition of the flow) of the weight of flow on the path times the length of the path:* $v(P) \times length(P)$.

Note that the average path length of a flow is independent of the particular choice of the path decomposition of the flow, provided every path in the path decomposition is simple.

Definition 39. *We define the minimum cost-routing into an interval* I *as the minimum average path length of any flow (in* B*) that sends exactly* $\frac{1}{2}$ *unit of flow in total to the terminals in interval* I.

Definition 40. *An interval* I *has the* (C, E) *property if the total capacity (of* B*) crossing* I *- i.e. has one endpoint to the left and one endpoint to the right of* I *- is at least* C *and if the minimum cost routing to* I *is at least* E.

Suppose that interval I has the (C, E)-property. Split I into I_1, I_2, I_3, three contiguous intervals. If there is total capacity at least $\frac{1}{6}$ in B crossing I_1 but not I, then I_1 has the $(C + \frac{1}{6}, E)$ property (because the minimum cost routing to I_1 is at least the minimum cost routing to I). Similarly, if there is at least $\frac{1}{6}$ capacity in B crossing I_3 but not I, then I_3 has the $(C + \frac{1}{6}, E)$ property.

Otherwise, there is at most $\frac{1}{6}$ total capacity entering I_2 from the left of I_1, and at most $\frac{1}{6}$ total capacity entering I_2 from the right of I_3. So consider the minimum cost routing to I_2. There must be at least $\frac{1}{2} - \frac{1}{6} - \frac{1}{6} = \frac{1}{6}$ total flow that enters I_2 from I_1 or I_3. So given the minimum cost routing to I_2 of cost α, we can stop paths that enter I_2 from I_1 or I_3 short, and reduce the cost of the routing by at least $\frac{1}{6}$. This would give a routing into I of cost $\alpha - \frac{1}{6}$, but the minimum cost routing into I is at least E, so $\alpha \geq E + \frac{1}{6}$. So either I_1 or I_3 has the $(C + \frac{1}{6}, E)$ property or I_2 has the $(C, E + \frac{1}{6})$ property.

So this implies that if we continue for $\log_3 \frac{x}{2}$ levels of this recursion (and there are in fact $\frac{x}{2}$ nodes in the path P_{good}), then we find an interval I that has the (C, E) property and $C + E \geq \frac{\log_3 \frac{x}{2}}{6}$ and so either C or E must be $\geq \frac{\log_3 \frac{x}{2}}{12}$.

If $C \geq \frac{\log_3 \frac{x}{2}}{12}$, then the cut-width of embedding B into the path P_{good} is at least C, which implies that at least C total capacity in H' has one endpoint in C'_i and one endpoint in C'_j,

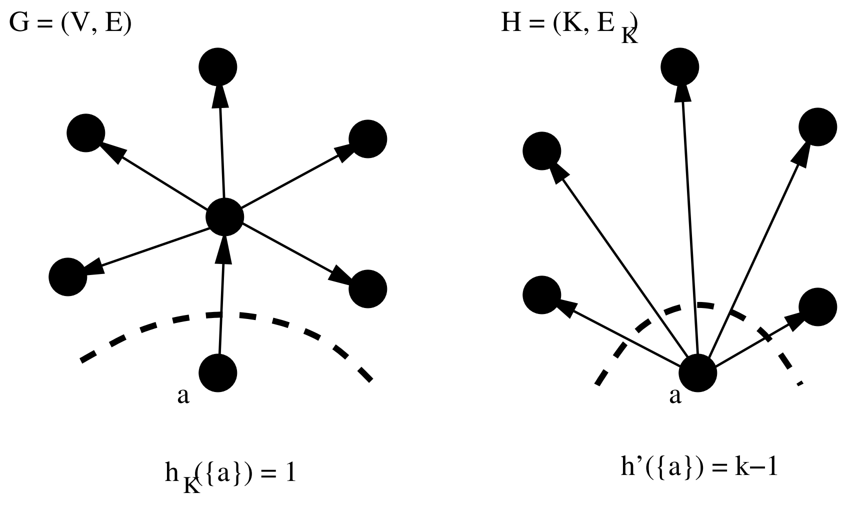

Figure 5-3: A counter-example to flow-sparsification in directed graphs.

and the interval I in P_a is contained between the image of C_i' in P_a and the image of C_j' in P_a. Because this capacity is in H', and at least $\frac{1}{2}$ of this capacity is routing using the natural path in G and traverses the interval I, and in this case the congestion of embedding H' into G (using the fixed routing of H' into G - note that H' was defined based on the optimal embedding of H into G) is at least $\frac{C}{2} = \Omega(\log \log n)$.

If $E \geq \frac{\log_3 \frac{x}{2}}{12}$, then there is a node $u \in I$ for which the minimum cost routing to c is at least $\frac{\log_3 \frac{x}{2}}{12} > \frac{\log \log n}{c_2}$, but then the edge (a, u) would not be good. I is an interval in P_{good} and this yields a contradiction.

Theorem 28. *There is an infinite family of graphs so that the best quality flow-sparsifier has quality $\Omega(\log \log k)$ and yet the best quality cut-sparsifier has quality $O(1)$.*

5.3 Counter-Example to Directed Flow-Sparsification

Here prove that, in the case of directed graphs, a good quality flow-sparsifier does not always exist. Consider the example in Figure 5-3.

The terminal a can send a unit flow to each other terminal. Additionally, no terminal besides a can send any flow to any other terminal. Hence for any terminal $b \in K - \{a\}$, the total capacity of directed edges out of b in finite quality flow-sparsifier must be zero. So the only directed edges that are assigned capacity must be from a to another terminal.

Yet for each $b \in K - \{a\}$, if a can send a unit flow to b the capacity of the edge from a to b in H must be at least one. Hence the total capacity out of a in any flow-sparsifier (again of finite quality) must be at least $k - 1$.

So a can send a total of $k - 1$ units of flow to other terminals in the flow-sparsifier H. But in G, a can send at most one total unit of flow to other terminals. So the quality of H as a flow-sparsifier is at least $k - 1$. In the case of directed graphs, we cannot hope for a flow-sparsifier whose quality is poly-logarithmic in k. However, if we allow a small number of non-terminals in H (say a linear number in k) we could still hope that good quality flow-sparsifiers exist in the directed case.

Chapter 6

Additional Applications

Here give additional applications of vertex sparsification to oblivious routing and to graph layout problems. The results in this section appear in [49].

6.1 Oblivious Routing

An oblivious routing scheme makes routing decisions based only on the starting and ending nodes of a routing request (and independently of the current load in the network and what other routing requests have been made). So routing decisions are based only on local knowledge, and consequently an oblivious routing scheme can be easily implemented in a distributed manner. We consider the routing goal of minimizing the congestion in a network - i.e. minimizing the maximum ratio of load to capacity on any edge.

Then the competitive ratio of an oblivious routing scheme is measured by how well the oblivious routing scheme performs (with respect to congestion) compared to the performance of the optimal routing scheme with fore-knowledge of what demands need to be routed. Valiant and Brebner [59] were the first to prove any performance guarantees for oblivious routing on any network topology, and gave an $O(\log n)$-competitive oblivious routing algorithm for the hypercube (on n nodes). In a breakthrough paper, Räcke [53] proved that for arbitrary graphs, there are oblivious routing algorithms that are $poly(\log n)$-competitive. Recently, Räcke [54] proved that there are in fact $O(\log n)$-competitive oblivious routing algorithms for general graphs. This performance guarantee even matches the competitive ratio of (optimal) adaptive routing algorithms that are given updated knowledge of the loads in the network before being asked to route each successive request!

But consider a related problem: Suppose a service provider is only responsible for routing requests between some small set of clients. In this setting, an oblivious routing protocol can make the assumption that all routing requests will have endpoints in some small set of terminals K. Suppose that $|K| = k$, and that $k^2 << O(\log n)$. These routing requests are certainly allowed to use paths that contain nodes in the entire web graph. We will call this problem the Steiner oblivious routing problem. If we set $K = V$, we recover the original oblivious routing problem in [59].

In this restricted scenario, how well can oblivious routing perform? We could ignore this promise that all routing requests will be between terminals in K, and we can use the oblivious routing scheme in [54] and get an oblivious routing scheme that is $O(\log n)$-competitive, and still based only on local knowledge. But if $k^2 << O(\log n)$, then this is a trivial guarantee: For each pair (a, b) of terminals, we could alternatively find a minimum congestion routing of a unit flow from a to b. Then a naive union bound over all $\binom{k}{2}$ pairs yields a performance guarantee that outperforms the $O(\log n)$-competitive oblivious routing guarantee!

Here, we give an oblivious routing scheme (which only needs to pre-define a routing for all pairs of terminals) that achieves an $O(\log k)$-competitive ratio. Hence the guarantee is logarithmic in the number of clients as opposed to logarithmic in the size of the web graph.

Theorem 29. *There is an algorithm to compute an $O(\log k)$-competitive Steiner oblivious routing scheme that runs in time polynomial in n and k.*

We can use the results in Section 4.2.1 in conjunction with the results in Section 3.2.4 and this immediately implies that we can efficiently construct a distribution $\vec{\lambda}$ on pairs $G_{f,T} \times R$ of a 0-decomposition $G_{f,T}$ and a set of $\binom{k}{2}$ simple paths in G - one connecting each pair of terminals. Furthermore, this distribution will satisfy the condition that the total load on any edge e is at most $O(\log k)c(e)$ – where the total load is the expected sum of the capacity $c_H(a, b)$ of each edge (a, b) in H for which the corresponding path $R_{a,b}$ in R

connecting a and b uses the edge e.

This distribution in turn defines a Steiner oblivious routing scheme: Each $G_{f,T}$ is a tree on K, and hence given a demand vector $\vec{r}$, the minimum congestion routing of $\vec{r}$ in $G_{f,T}$ is the natural routing in which for each pair of terminals $a, b \in K$, we send $\vec{r}_{a,b}$ units of flow directly on the unique path in $G_{f,T}$ from a to b. Note that since $G_{f,T}$ is a flow-sparsifier, any demand vector $\vec{r}$ that can be routed in G with congestion at most one, $\vec{r}$ has congestion at most one when routed in $G_{f,T}$.

For each pair $a, b \in K$ we will determine how to route $\lambda_{G_{f,T} \times R} \vec{r}_{a,b}$ total units of flow based on $G_{f,T}$ and R: Let $p = a, u_1, u_2, ..u_r, b$ be the unique simple path connecting a and b in $G_{f,T}$. And let $R_{a,u_1}, R_{u_1,u_2}, ...R_{u_r,b}$ denote the set of paths in R connecting a to u_1 and u_1 to u_2, ... u_r to b respectively. Then we will send a total of $\lambda_{G_{f,T} \times R} \vec{r}_{a,b}$ units of flow along the path $p = R_{a,u_1} \circ R_{u_1,u_2} \circ ....R_{u_r,b}$ where $\circ$ denote the concatenation operator. Notice that this routing scheme decides how to route between two terminals based only on the identity of the start and end location and hence this routing scheme is oblivious.

The total amount of flow that we route for each pair of terminals $a, b \in K$ is $\vec{r}_{a,b}$. To analyze the competitive ratio, suppose that the $\vec{r}$ is routable in G with congestion at most one. Hence $\vec{r}$ is routable in each $G_{f,T}$ with congestion at most one and consequently the total load incurred by this routing scheme on any edge e will be at most

$$\sum_{f,T,K} \lambda_{G_{f,T} \times R} load_{G_{f,T} \times R}(e) \leq O(\log k) c(e)$$

This routing scheme incurs congestion at most $O(\log k)$ when routing $\vec{r}$ and so this routing scheme is $O(\log k)$-competitive.

For comparison, we note that the routing scheme given in Section 3.2.1 are not oblivious (but can still be computed in time polynomial in the number of terminals, and independent of the size of G), but can achieve an $O(1)$-competitive ratio in graphs that exclude a fixed minor. The oblivious routing scheme, however, cannot achieve a competitive ratio better than $\Omega(\log k)$ even in planar graphs [12], [47].

6.2 Layout Problems

Here we demonstrate another pattern for applying our results: we re-write graph layout problems using an un-crossing argument. This allows us to prove that the value of solutions to graph layout problems are approximately preserved by a good quality flow-sparsifier, although here the optimal value to a layout problem is not determined by a single minimum cut but rather based on a nested family of minimum cuts. Specifically, we apply our results on vertex sparsification to Steiner generalizations of minimum cut linear arrangement and minimum linear arrangement.

The minimum cut linear arrangement problem is defined as: Given an undirected, capacitated graph $G = (V, E)$ we want to find an ordering of the vertices $v_1, v_2, ...v_n$ which minimizes the value of

$$C = \max_{1 \leq i \leq n} h(\{v_1, v_2, ...v_i\})$$

We can define a natural generalization of this problem in which we are given a set $K \subset V$ of size k, and we want to find an ordering of the terminals in K, $u_1, u_2, ...u_k$ and a partition $A_1, A_2, ...A_k$ of the remaining nodes $V - K$ (and let $B_i = A_i \cup \{u_i\}$) which minimizes the value of

$$C_K = \max_{1 \leq i \leq k} h(\cup_{1 \leq j \leq i} B_i)$$

We will refer to this problem as Steiner minimum cut linear arrangement and we will refer to the sets B_i as a Steiner layout. We can also give an identical generalization to minimum linear arrangement too. Here, we can re-write this optimization problem using an un-crossing argument so that the value only depends on the terminal cut function h_K.

Lemma 21. *Given an ordering $u_1, u_2, ...u_k$ of the terminals we can find a partition*

$$A_1, A_2, ...A_k$$

of V such that $A_i \cap K = \{u_i\}$ and

$$h(\cup_{j=1}^{i} A_j) = h_K(\{u_1, u_2, ...u_i\})$$

Proof: The cut function $h : 2^V \to \mathbb{R}^+$ is a submodular function. So for all $S, T \subset V$:

$$h(S) + h(T) \geq h(S \cap T) + h(S \cup T)$$

For each i, find a set $B_i \subset V$ such that $B_i \cap K = \{u_1, u_2, ...u_i\}$ and $h(B_i) = h_K(\{u_1, u_2, ...u_i\})$. Consider the sets B_1 and B_2. We can find sets C_1, C_2 such that $C_1 \subset C_2$ and $h(B_1) = h(C_1), h(B_2) = h(C_2)$ and $C_1 \cap K = B_1 \cap K$, $C_2 \cap K = B_2 \cap K$. This is true via submodularity: Choose $C_1 = B_1 \cap B_2$ and $C_2 = B_1 \cup B_2$.

So $C_1 \cap K = (B_1 \cap K) \cap (B_2 \cap K) = (B_1 \cap K)$ and also $C_2 \cap K = (B_1 \cap K) \cup (B_2 \cap K) = B_2 \cap K$ because $B_1 \cap K \subset B_2 \cap K$.

Also $h(B_1) + h(B_2) \geq h(C_1) + h(C_2)$ via submodularity. However $h(B_1)$ is the minimal value of an edge cut separating $B_1 \cap K$ from $K - (B_1 \cap K)$ and C_1 also separates $B_1 \cap K$ from $K - (B_1 \cap K)$, and a similar statement holds for C_2. So the above inequality implies

$$h(B_1) = h(C_1), h(B_2) = h(C_2)$$

We can continue the above argument and get sets $C_1, C_2, ...C_k$ such that

$$h(C_i) = h_K(\{u_1, u_2, ...u_i\}) \text{ and } C_1 \subset C_2 ... \subset C_k$$

Lastly, we set $A_i = C_i - \cup_{j=1}^{i-1} C_i$ and this implies the lemma.

□

Since an α-quality cut-sparsifier preserves *all* values of h_K within an α-factor, we can once again algorithmically reduce G to a tree at the cost of an $O(\log k)$-factor using Theorem 20. Note that the results of [60] and [23] give polynomial time algorithms for minimum cut-linear arrangement and minimum linear arrangement when the input graph is a tree.

Also, since $G_{f,T}$ is a cut-sparsifier for G and K, we can apply Lemma 21 and this implies given a layout for $G_{f,T}$ of cost C we can find a Steiner layout of cost at most C (for either the minimum cut linear arrangement or the minimum linear arrangement problems):

Corollary 11. *There is a polynomial time $O(\log k)$-approximation algorithm for Steiner minimum cut linear arrangement.*

Corollary 12. *There is a polynomial time $O(\log k)$-approximation algorithm for Steiner minimum linear arrangement.*

Chapter 7

Open Questions

The main open question in this work is whether allowing additional non-terminals can result in better (or even constant) quality flow-sparsifiers. We do not know of *any* bound $f(k)$ for which every graph G and set of k terminals $K \subset V$ has a constant quality flow-sparsifier that contains at most $f(k)$ non-terminals.

Open Question 2. *Does allowing non-terminals allow a flow-sparsifier to be significantly more expressive?*

Answering this question, either in the positive or negative, may require a significant departure from the techniques in this thesis. One technical difficulty is that if we allow non-terminals, the set of good flow-sparsifiers (or even the set of good cut-sparsifiers) is no-longer convex.

Another interesting question is to determine the integrality gap of the semi-metric relaxation for the 0-extension problem. The integrality gap is only known to be bounded between $O(\frac{\log k}{\log \log k})$ and $\Omega(\sqrt{\log k})$. Improving the upper bound on this integrality gap would immediately result in an asymptotic improvement in the quality of both cut and flow-sparsifiers. In fact, even improving this bound in the case in which the input metric is ℓ_1 would improve the best known bounds for cut-sparsifiers.

Additionally, we do not know of any hardness results for constructing cut or flow-sparsifiers.

Open Question 3. *Is it hard to compute the best quality flow-sparsifier?*

Open Question 4. *Is it hard to compute the best quality cut-sparsifier?*

An exact algorithm for the congestion minimization problem would immediately yield an algorithm for computing the best flow-sparsifier, yet we suspect that computing the best quality cut-sparsifier is computationally difficult.

Bibliography

[1] M. Adler, N. Harvey, K. Jain, R. Kleinberg, and A. R. Lehman. On the capacity of information networks. In *SODA*, pages 251–260, 2006.

[2] A. Aggarwal and J. Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM*, pages 116–127, 1988.

[3] R. Ahlswede, N. Cai, S. Y. Li, and R. Yeung. Network information flow. *IEEE Transactions on Information Theory*, pages 1204–1216, 2000.

[4] R. Andersen and U. Feige. Interchanging distance and capacity in probabilistic mappings. *Arxiv*, 2009.

[5] A. Archer, J. Fakcharoenphol, C. Harrelson, R. Krauthgamer, K. Talwar, and E. Tardos. Approximate classification via earthmover metrics. In *SODA*, pages 1079–1087, 2004.

[6] S. Arora, J. Lee, and A. Naor, ‘ Euclidean distortion and the sparsest cut. *Journal of the AMS*, pp. 1–21, 2008. Preliminary version in *STOC*, pages 553-562, 2005.

[7] S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM*, 2009. Preliminary version in *STOC*, pages 222-231, 2004.

[8] Y. Aumann and Y. Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM Journal on Computing*, pages 291–301, 1998.

[9] Y. Azar, E. Cohen, A. Fiat, K. Haim, and H. Räcke. Optimal oblivious routing in polynomial time. In *STOC*, pages 383–388, 2003.

[10] K. Ball. Markov chains, Riesz transforms and Lipschitz maps. *Geometric and Functional Analysis*, pages 137–172, 1992.

[11] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *FOCS*, pages 184–193, 1996.

[12] Y. Bartal and S. Leonardi. On-line routing in all optimal networks. *Theoretical Computer Science*, pages 19–39, 1999. Preliminary version in *ICALP*, pages 516–526, 1997.

[13] J. Batson, D. Spielman, and N. Srivastava. Twice-ramanujan sparsifiers. In *STOC*, pages 255–262, 2009.

[14] W. Beckner. Inequalities in Fourier analysis. *Annals of Mathematics*, pages 159–182, 1975.

[15] A. Benczúr and D. Karger. Approximating s-t minimum cuts in õ(n^2) time. In *STOC*, pages 47–55, 1996.

[16] A. Bonami. Etude des coefficients de Foureir des fonctions de $l^p(g)$. *Annales de 'institut Fourier*, pages 335–402, 1970.

[17] J. Bourgain. On the distribution of the Fourier spectrum of boolean functions. *Israel Journal of Mathematics*, pages 269–276, 2002.

[18] G. Calinescu, H. Karloff, and Y. Rabani. Approximation algorithms for the 0-extension problem. *SIAM Journal on Computing*, pages 358–372, 2004. Preliminary version in *SODA* 2001, pages 8–16.

[19] Y. Chan, W. Fung, L. Lau, and C. Yung. Degree bounded network design with metric costs. In *FOCS*, pages 125–134, 2008.

[20] M. Charikar, T. Leighton, S. Li, A. Moitra. Vertex sparsifiers and abstract rounding algorithms. In *FOCS* 2010, to appear.

[21] C. Chekuri, S. Khanna, J. Naor, and L. Zosin. A linear programming formulation and approximation algorithm for the metric labeling problem *SIAM Journal on Discrete Math*, pages 608–625, 2004. Preliminary version in *SODA*, pages 109–118, 2001.

[22] P. Chew. There is a planar graph almost as good as the complete graph. In *SoCG*, pages 169–177, 1986.

[23] F. Chung. On optimal linear arrangements of trees. *Computers and Math with Applications*, pages 43–60, 1984.

[24] I. Dinur, E. Friedgut, G. Kindler, and R. O'Donnell. On the fourier tails of bounded functions over the discrete cube. *Israel Journal of Mathematics*, pages 389–412, 2007. Preliminary version in *STOC*, pages 437–446, 2006.

[25] M. Englert, A. Gupta, R. Krauthgamer, H. Räcke, I. Talgam-Cohen, and K. Talwar. Vertex sparsifiers: new results from old techniques. In *APPROX* 2010, to appear.

[26] J. Fakcharoenphol, C. Harrelson, S. Rao, and K. Talwar. An improved approximation algorithm for the 0-Extension Problem. In *SODA* 2003, pages 257–265.

[27] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, pages 485–497, 2004. Preliminary version in *STOC* 2003, pages 448–455.

[28] J. Fakcharoenphol and K. Talwar. An improved decomposition theorem for graphs excluding a fixed minor. In *APPROX + RANDOM*, pages 36–46, 2003.

[29] R. Floyd. Permuting information in idealized two-level storage. *Complexity of Computer Calculations*, pages 105–109, 1972.

[30] N. Garg and J. Könemann. Faster and simpler approximation algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing*, pp. 680–652, 2007. Preliminary version in *FOCS*, pages 300-309, 1998.

[31] N. Garg, V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing*, 25:235–251, 1996. Preliminary version in *STOC* 1993, pages 698–707.

[32] D. Golovin, V. Nagarajan, and M. Singh. Approximating the k-multicut problem. In *SODA*, pages 621–630, 2006.

[33] M. Grötschel, L. Lovász, and A. Schrijver. Geometric algorithms and combinatorial optimization. *Springer Verlag*, 1993.

[34] A. Gupta, V. Nagarajan, and R. Ravi. An improved approximation algorithm for requirement cut. *Operations Research Letters*, pages 322–325, 2010.

[35] C. Harrelson, K. Hildrum, and S. Rao. A polynomial-time tree decomposition to minimize congestion. In *SPAA*, pages. 34–43, 2003.

[36] N. Harvey, R. Kleinberg, and A. R. Lehman. On the capacity of information networks. *IEEE Transactions on Information Theory*, pages 2345–2364, 2006.

[37] K. Jain, V. Vazirani, R. Yeung, and G. Yuval. On the capacity of multiple unicast sessions in undirected graphs. *IEEE International Symposium on Information Theory*, 2005.

[38] W. Johnson, J. Lindenstrauss, and G. Schechtman. Extensions of Lipschitz maps into Banach spaces. *Israel Journal of Mathematics*, pages 129–138, 1986.

[39] A. Karzanov. Minimum 0-extensions of graph metrics. *European Journal of Combinatorics*, pages 71–101, 1998.

[40] R. Khandekar, S. Rao, and U. Vazirani. Graph partitioning using single commodity flows. *Journal of the ACM*, 2009. Preliminary version in *STOC*, pages 385-390, 2006.

[41] S. Khot and A. Naor. Nonembeddability theorems via fourier analysis. In *FOCS* 2005, pages 101–112.

[42] P. Klein, S. Plotkin, and S. Rao. Excluded minors, network decomposition, and multicommodity flow. In *STOC*, pages 682–690, 1993.

[43] L. C. Lau. An approximate max-steiner-tree packing min-steiner-cut theorem. *Combinatorica*, pages 71–90, 2007. Preliminary version in *FOCS*, pages 61–70, 2004.

[44] T. Leighton and A. Moitra. Extensions and limits to vertex sparsification. In *STOC* 2010, pages 47–56.

[45] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, pages 787–832, 1999. Preliminary version in *FOCS* 1988, pages 422-431.

[46] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, pages 215–245, 1995. Preliminary version in *FOCS* 1994, pages 577–591.

[47] B. Maggs, F. Meyer auf der Heide, B. Vöcking, and M. Westermann. Exploiting locality for networks of limited bandwidth. In *FOCS*, pages 284–293, 1997.

[48] K. Makarychev and Y. Makarychev. Metric extension operators, vertex sparsifiers and Lipschitz extendability. In *FOCS* 2010, to appear.

[49] A. Moitra. Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size. In *FOCS*, pages 3–12, 2009.

[50] E. Mossel, R. O'Donnell, K. Oleszkiewicz. Noise stability of functions with low influences: invariance and optimality. *Annals of Mathematics*, pages 389–412, 2007. Preliminary version in *FOCS*, pages 21–30, 2005.

[51] V. Nagarajan and R. Ravi. Approximation algorithms for requirement cut on graphs. In *APPROX + RANDOM*, pages 209–220, 2005.

[52] R. O'Donnell. Some topics in the analysis of boolean functions. In *STOC*, pages 569–578, 2008.

[53] H. Räcke. Minimizing congestion in general networks. In *FOCS* 2002, pages 43–52.

[54] H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *STOC* 2008, pages 255–264.

[55] D. Randall. Mixing. In *FOCS*, pages 4–15, 2003.

[56] D. Shmoys. *Approximation algorithms for Cut problems and their application. to divide-and-conquer* In *Approximation Algorithms for NP-hard Problems*. PWS, 1997.

[57] D. Spielman and N. Srivastava. Graph sparsification by effective resistances. In *STOC*, pages 563–568, 2008.

[58] D. Spielman and S. Teng. Nearly-linear time algorithms for graph partitioning, sparsification and solving linear systems. In *STOC*, pages 81–90, 2004.

[59] L. Valiant and G. Brebner. Universal schemes for parallel communication. In *STOC*, pages 263–277, 1981.

[60] M. Yannakakis. A polynomial time algorithm for the min-cut linear arrangement of trees. *Journal of the ACM*, pages 950–988, 1985. Preliminary version in *FOCS*, pages 274–281, 1983.

[61] N. Young. Sequential and parallel algorithms for mixed packing and covering. In *FOCS*, pages 538–546, 2001.