Thesis for the degree
Doctor of Philosophy

Submitted to the Scientific Council of the
Weizmann Institute of Science
Rehovot, Israel

עבודת גמר (תזה) לתואר
דוקטור לפילוסופיה

מוגשת למועצה המדעית של
מכון ויצמן למדע
רחובות, ישראל

By
Lior Kamma

מאת
ליאור קמה

אלגוריתמים לדילול צמתים בגרפים
Algorithms for Graphical Vertex Sparsifiers

Advisor:
Prof. Robert Krauthgamer

מנחה:
פרופ' רוברט קראוטגמר

March 2017

אדר תשע"ז

Acknowledgments

Upon completing one of the most educational, influential and perhaps even crucial periods of my life, it is with great pleasure that I thank the people who have helped me through the journey.

My first and foremost gratitude goes to my adviser, Robert Krauthgamer, for sharing with me his insights, ingenuity and knowledge, and never withholding his wisdom, intuition and patience. Robi, I want to thank you for educating me in the ways of scientific writing, and teaching me that a good idea is not the end of the path, but merely the beginning. For never letting me get away with less than perfect phrasing, and especially, for never letting me rephrase without first explaining the motivation, and making sure I am convinced that this is the way to go. I cannot thank you enough.

I would like to thank my fellow students in the institute, Yossi Arjevani, Chen Attias, Arnold Filtser, Ariel Jaffe, Bundit Laekhanukit, Amit Moscovich Eiger, Yosef Pogrow, Inbal Rika, Nimrod Talmon, Ohad Trabelsi and Otniel Van-Hendel. Thank you for making the wonderful environment at Weizmann even more pleasant and warm. A special thanks is devoted to the good friends I have acquired during my time in the Weizmann Institute. Rajesh Chitnis, Roee David and Elazar Goldenberg, thank you for endless discussions and talks (scientific and otherwise). You have been my pillar of strength throughout my time here, and I am most grateful to have you in my life.

A warm and loving gratitude goes to my parents, Aliza and Moti, and my wife's parents, David and Hannah, for their love, support and encouragement (even prior to my graduate studies). My deepest gratitude goes to my mother, whose invaluable help during the last few years is one of the main reasons I was able to follow through. Mum, I could not have done this without you.

Last, but certainly not least, I would like to thank my wife Gili and my kids, Yael, Yoav and Michal for their endless love and support. Thank you for making all the necessary compromises, and for putting up with the stress, anxiety and exhaustion incurred by my studies (almost) without complaining. You make it all worthwhile. I thank you and I love you very much.

Declaration

I declare that the thesis summarizes my own research. Parts of the research were performed in collaboration with other researchers, and have already been published in refereed journals and conferences, as described below.

The results presented in Part A, as well as in Section 6 of Part B are joint work with Robert Krauthgamer and Huy L. Nguyen, and were published in *SIAM Journal of Computing* [KKN15].

The results presented in Section 7 of Part B are joint work with Robert Krauthgamer, and were accepted for publication in *Algorithmica* [KK16].

The results presented in Part C are joint work with Rajesh Chitnis and Robert Krauthgamer, and were published in the *International Workshop in Graph-Theoretic Concepts in Computer Science* [CKK16].

The results presented in Part D are joint work with Alexandr Andoni and Robert Krauthgamer, and have recently been submitted for publication [AKK17].

Abstract

Given a graph G, *graph compression* problems ask to construct succinct data structures that maintain certain properties of G. Within this context, this thesis studies problems of *vertex sparsification* in graphs, where the input is a graph $G = (V, E)$ and a set $T \subseteq V$ of terminals, perhaps with some extra attributes, such as edge weights or capacities, and the goal is to construct a "small" graph that maintains properties of T.

Our first result involves *distance sparsifiers*. Specifically, we study the Steiner Point Removal (SPR) problem, where, given an edge-weighted graph $G = (V, E, w)$ and a subset of terminals $T \subseteq V$, the goal is to construct a graph $G' = (T, E', w')$ that is isomorphic to a minor of G, and such that inter-terminal distances are distorted by at most $\alpha \geq 1$. We show that SPR can always be solved with distortion $\alpha \leq O(\log^5 |T|)$.

In the second part of the thesis we study *randomized low-diameter metric decompositions*. We present a new property of randomized decompositions called *degree of separation*, which gives a tail bound on the probability that "many" distinct subsets intersect a shortest path in a metric space. We also design a new decomposition of metrics that are induced by so called *p-path-separable graphs*, which roughly refers to edge-weighted graphs that admit vertex-separators consisting of at most p shortest paths in the graph.

The third part of this thesis considers *cut sparsifiers*, and specifically generalizes a classical result of Gomory and Hu (1961), who showed that in every edge-weighted graph $G = (V, E, w)$, the minimum st-cut values, when ranging over all $s, t \in V$, take at most $|V| - 1$ distinct values. That is, these $\binom{|V|}{2}$ instances exhibit *redundancy factor* $\Omega(|V|)$. Motivated by this result, we obtain *tight* bounds for the redundancy factor of several generalizations of the minimum st-cut problem. A natural application of these bounds is to construct small data structures that store all relevant cut values. We initiate this direction by giving some upper and lower bounds.

In the final part of this thesis we introduce a *batch* version of sparse recovery, where the goal is to construct a sequence of estimates $A'_1, \ldots, A'_m \in \mathbb{R}^n$ of unknown signals $A_1, \ldots, A_m \in \mathbb{R}^n$, using linear measurements, under an assumption of *average sparsity*. We resolve the question of minimizing the number of measurements up to polylogarithmic factors, by presenting a randomized adaptive scheme that with high probability performs $\tilde{O}(km)$ measurements, which is asymptotically tight up to logarithmic factors. Additionally, we show that adaptivity is necessary for every non-trivial scheme that solves the batch sparse recovery problem.

Contents

1 Introduction

In today's day and age, our technology-driven civilization generates and consumes massive amounts of data. For many reasonable purposes, most of this data is irrelevant, or even useless, and therefore the utility of compressing the data while keeping the important or relevant *information*, either exactly or approximately is clear, both theoretically and practically. Once the compressed data (be it a graph, a network traffic log, an image or audio signal etc.) is computed as a pre-processing step, further processing can be performed on the compressed instance instead of on the original one, using less resources like running time, memory and communication bandwidth, or achieving better accuracy (for example, when the solution is approximate and the approximation ratio depends on the size of the object in question). The main theme of this thesis is algorithms for lossy compression of data and retrieval of information from partial, compressed or corrupted data.

Within this context we mainly focus on *graph compression*. Loosely speaking, graph compression is the process of constructing, given a graph G, a (small) data structure, that maintains certain features (quantities) of G, such as distances, cut or flow values. Extensively studied examples for this genre include graph spanners, distance oracles, cut and flow sparsifiers, and spectral sparsifiers, see e.g. [PS89, TZ05, BK96, BSS09]. A prime example for this genre is *vertex-sparsification*, where G has a designated subset of vertices T, referred to as *terminals*, and the goal is construct a new graph G' (rather than an arbitrary data structure), such that

- $T \subseteq V(G')$ and $|V(G')|$ is "small"; and
- certain properties of T, such as cut or flow values and connectivity requirements, or structural properties of G (e.g. planarity) are maintained in G' (either exactly or approximately).

When properties of T are maintained approximately, the approximation ratio is referred to as the *quality* of the sparsifier. The algorithmic utility of such constructions is clear – once the new graph is computed (as a preprocessing step), further processing can be performed on G' instead of on G, using less resources like runtime and memory, or achieving better accuracy; for example, when the solution is approximate, and the approximation factor depends on the number of vertices, executing the algorithm on a smaller graph may attain a better approximation ratio.

This work focuses mainly on two vertex sparsification problems: Distance Sparsifiers and Cut Sparsifiers. These examples are extensively studied in the literature and have many applications, both theoretical and practical. Prior studies introduced new combinatorial techniques, and showed interesting connections to several other problems e.g. the 0-extension problem, embedding into trees and several metric decomposition problems, as well as connections between distance and cut sparsifiers.

This thesis is divided into four parts. In what follows, we present a brief overview of each part.

1.1 Distance Sparsifiers and Steiner Point Removal

The first part of the thesis deals with distance sparsifiers, which are vertex sparsifiers that maintain inter-terminal distances, either exactly or approximately. Specifically, we focus on sparsifiers whose vertex set is exactly the designated set of terminals.

Formally, given an edge-weighted undirected graph, $G = (V, E, w)$, let $d_{G,w}$ denote the shortest-path metric induced by w on V. Given a designated set of terminals $T \subseteq V$, an *α-quality distance sparsifier* for G is an edge-weighted graph $G' = (V', E', w')$ such that

- G' is isomorphic to a minor of G (thus maintaining certain properties of G, e.g. planarity).
- $T \subseteq V'$ and V' is "small" (in terms of $|T|$).
- Distances between the terminals are distorted by at most factor $\alpha \geq 1$, that is,

$$\forall u, v \in T, \qquad d_{G,w}(u, v) \leq d_{G',w'}(u, v) \leq \alpha \cdot d_{G,w}(u, v) \ .$$

Vertices in $V' \setminus T$ are usually referred to as *Steiner points*. Our results deal with the *Steiner Point Removal (SPR)* problem, where the second requirement above is the strongest possible, that is $V' = T$. This formulation of the SPR problem was proposed by Chan, Xia, Konjevod, and Richa [CXKR06, Section 5], who posed the problem of bounding the distortion α. In the first part of this thesis we answer their open question, and give a *polylog*$(|T|)$-quality sparsifier. The formal statement follows.

Theorem 1.1. *Let $G = (V, E, w)$ be an edge-weighted graph with terminal set $T \subseteq V$. Then there exists a graph $G' = (T, E', w')$, that is isomorphic to a minor of G and attains quality $O(\log^5 |T|)$, i.e.,*

$$\forall u, v \in T, \qquad 1 \leq \frac{d_{G',w'}(u, v)}{d_{G,w}(u, v)} \leq O(\log^5 |T|).$$

Moreover, G' is computable in randomized polynomial time.

A previous result by Englert *et al.* [EGK+10] introduced a variation of the Steiner Point Removal problem. They gave an algorithm which produces an $O(\log |T|)$ quality distance preserving graph, which is a *convex combination* of minors of the original graph. The fundamental innovation of Theorem 1.1 over [EGK+10] is the existence of a *single* minor of G that maintains distances with polylogarithmic quality.

We prove Theorem 1.1 in Part A, and elaborate on previous work as well as potential applications. This result has been published in the *SIAM Journal of Computing* [KKN15].

1.2 Metric Decompositions

In recent decades, the problem of decomposing a metric space into low-diameter subspaces has become a key step in the solution for many problems, including metric embeddings (e.g. [Ass83, Bar96, Rao99, GKL03, FRT04, KLMN05]), distance oracles and routing schemes

design (e.g. [AP90, DSB97, Tal04, CGMZ05, MN07]), graph sparsification (e.g. [EGK+14], as well as the results described in Section 1.1) and optimization problems, such as multi-commodity cuts [KPR93, GVY96, LR99], 0-extension [CKR04] and the traveling salesman problem [Tal04].

Following the more recent literature, our results in the second part of the thesis focus on *randomized* (or *stochastic*) decompositions, which generally refers to a probability distribution over partitions of a metric space into sets (called clusters) of low diameter, such that nearby points are likely to be "clustered" together. Specifically, we consider *padded decompositions*, in which, loosely speaking, for every point x we "expect" the cluster containing x to contain in addition a (small) ball around x. A formal definition follows.

Let (X, d) be a metric space, and denote the ball of radius $\varrho > 0$ around $x \in X$ by $B(x, \varrho) := \{y \in X : d(x, y) \leq \varrho\}$. Let Π be a partition of X. Every $S \in \Pi$ is called a *cluster*, and for every $x \in X$, let $\Pi(x)$ denote the unique cluster $S \in \Pi$ such that $x \in S$.

Definition 1.2. *A metric space (X, d) is called* β-decomposable *if for every $\Delta > 0$, there is a probability distribution μ over partitions of X, satisfying the following requirements:*

(a). *Diameter bound: for all $\Pi \in \mathrm{supp}(\mu)$ and all $S \in \Pi$,* $\mathrm{diam}(S) \leq \Delta$.

(b). *Padding probability: for every $x \in X$,* $\Pr_{\Pi \sim \mu}[B(x, \Delta/\beta) \subseteq \Pi(x)] \geq 1/2$.

μ is called a β-decomposition of X.

We note that while definition 1.2 is commonly used in literature, and is given here for sake of simplicity, our work, described in detail in Part B involves a slightly refined definition of padded decompositions introduced by Abraham *et al.* [AGG+14].

1.2.1 Metric Decompositions with Concentration

Our proof in Part A for the SPR problem in fact features a new property of metric decompositions, called *degree of separation*. We believe that this tool is important, and that further applications of this property will be found. In previous work (e.g. [EGK+14]) that proved existence of a convex combination of minors of G that maintains distances approximately, it was enough to show that the metric space induced by the graph was β-decomposable, and rely on linearity of expectation. However, to show existence of a *single* minor that maintains distances with low quality, β-decomposability was not sufficient, and we proposed the following notion of degree of separation, which provides high probability bounds.

Let $P = (x_0, x_1, \ldots, x_\ell)$ be a *shortest path*, i.e. a sequence of points in X such that $\sum_{i \in [\ell]} d(x_{i-1}, x_i) = d(x_0, x_\ell)$. We denote its length by $d(P) := d(x_0, x_\ell)$, and say that P *meets* some $S \subseteq X$ if $S \cap P \neq \emptyset$. Given a partition Π of X, we define the *degree of separation* of P with respect to Π as

$$Z_P(\Pi) := \sum_{S \in \Pi} \mathbb{1}_{\{P \text{ meets } S\}}.$$

Roughly speaking, the degree of separation property augments Definition 1.2 with a tail bound on $Z_P(\Pi)$.

Theorem 1.3. *For every n-point metric space (X, d) and every $\Delta > 0$ there is a padded decomposition μ, that satisfies requirements* (a)-(b) *of Definition 1.2 for $\beta = O(\log n)$, and furthermore*

(c). *Degree of separation: For every shortest path P of length $d(P) \leq \frac{\Delta}{\beta}$,*

$$\forall t \geq 1, \qquad \Pr_{\Pi \sim \mu}[Z_P > t] \leq 2e^{-\Omega(t)}. \tag{1.1}$$

In Section 6 of Part B we present a slightly stronger and more elaborate result than Theorem 1.3, that involves terminals and discusses general shortest paths rather than paths of small length. This result was published in *SIAM Journal of Computing* [KKN15].

1.2.2 Metric Decomposition of Path-Separable Graphs

A second result regarding metric decompositions deals with metrics that arise as shortest-path metrics in (certain) graphs. Specifically, given $G = (V, E, w)$ a connected graph with non-negative edge weights, recall that $d_{G,w}$ denotes the shortest-path metric induced on V by G. We say that a graph G is β-decomposable if the metric space $(V, d_{G,w})$ is β-decomposable. Loosely speaking, a p-path-separable graph is an edge-weighted graph $G = (V, E, w)$ that admits a vertex separator consisting of at most p shortest paths. Note that such separators inherently depend on the weight assignment w, and not only the "topology" of G, since they are composed of shortest paths in G. Our result is that every p-path-separable graph G admits an $O(\ln(p \ln |V|))$-decomposition. This result refines the $\Theta(\ln |V|)$ bound known for general graphs. Moreover, this result implies new bounds for special families of graphs. While formally describing the main result requires the technically involved definition of p-path-separable graphs, and is therefore deferred to Part B, a principal implication gives bounds for graph families with low treewidth.

Proposition 1.4. *Let $G = (V, E)$ be of treewidth t. Then for every assignment w of edge weights to E, $(V, d_{G,w})$ is $O(\ln(t \ln |V|))$-decomposable.*

The known upper bound for graphs of treewidth t is $\beta = O(t)$ due to [AGG$^+$14]. Our decomposition provides a tradeoff between t and $|V|$ and matches or improves all other bounds when $t \geq \ln \ln |V|$. It is conjectured that $\beta = O(\log t)$, which would be tight due to the lower bound of Bartal [Bar96], and our result provides partial evidence in favor of this conjecture. Specifically, it provides a tight upper bound if $t \geq \ln |V|$. The technical details are presented in Section 7 of Part B. This result was accepted for publication in *Algorithmica* [KK16].

1.3 Cut Sparsifiers for Restricted Families of Cuts

The third part of the thesis deals with cut sparsifiers. It initiates the study of small data structures that store the cut values of certain families of cuts in a given graph. Specifically,

we show that for the families in question we can, in a sense, enumerate all cut values more efficiently than the naïve approach due to inherent redundancy.

Consider first the *minimum st-cut* problem, where, given an edge-weighted graph $G = (V, E, w)$ and two vertices $s, t \in V$, the goal is to find a set of edges of minimum total weight that separates s, t (meaning that removing these edges from G ensures there is no s-t path). This problem is one of the most fundamental combinatorial optimization problems, and was studied extensively, see e.g. the famous minimum-cut/maximum-flow duality [FF56]. It has numerous theoretical applications, such as bipartite matching and edge-disjoint paths, in addition to being extremely useful in many practical settings, including network connectivity, network reliability, and image segmentation, see e.g. [AMO93] for details. Several generalizations of the problem, such as multiway cut, multicut, and k-cut, have been well-studied in operations research and theoretical computer science.

In every undirected graph $G = (V, E, w)$, there are in total $\binom{|V|}{2}$ instances of the minimum st-cut problem, given by all pairs $s, t \in V$. Potentially, each of these instances could have a different value for the minimum cut. However, the seminal work of Gomory and Hu [GH61] discovered that *undirected* graphs admit a significantly stronger bound (see also [AMO93, Lemma 8.15] or [CCPS98, Section 3.5.2]).

Theorem 1.5 ([GH61])**.** *Let $G = (V, E, w)$ be an edge-weighted undirected graph. Then the number of distinct values over all possible $\binom{|V|}{2}$ instances of the minimum st-cut problem is at most $|V| - 1$.*

The beautiful argument of Gomory and Hu shows the existence of a tree $\mathcal{T} = (V, E', w')$, usually called a *flow-equivalent tree*, such that for every $s, t \in V$ the minimum st-cut value in $\mathcal{T}$ is exactly the same as in G. (They further show how to construct a so-called *cut-equivalent tree*, which has the stronger property that every vertex-partitioning that attains a minimum st-cut in $\mathcal{T}$, also attains a minimum st-cut in G; see Section 8.3 for more details on this and related work.) Every G which is a tree (e.g., a path) with distinct edge weights has exactly $|V| - 1$ distinct values, and hence the Gomory-Hu bound is existentially tight.

Another way to state Theorem 1.5 is that there is always a huge redundancy between the $\binom{|V|}{2}$ minimum st-cut instances in a graph. More precisely, the "redundancy factor", measured as the ratio between the number of instances and the number of distinct optimal values attained by them, is always $\Omega(|V|)$. In Part C we study the redundancy factor in several generalizations of minimum st-cut. Specifically, we study the Group-Cut problem. Given an undirected edge-weighted graph $G = (V, E, w)$, Given two disjoint sets $A, B \subseteq V$ find a minimum (A, B)-cut, i.e., a set of edges of minimum weight that separates every vertex in A from every vertex in B. Our main result regarding Group-Cut gives a tight bound on the redundancy factor of the family of all instances where A and B are of given sizes α and β, respectively. This result generalizes Theorem 1.5, which proves a similar bound for the special case $\alpha = \beta = 1$.

Theorem 1.6. *For every two constants $\alpha, \beta = O(1)$, there are at most $O(n^{\alpha+\beta-1})$ distinct cut values over all possible $\Omega(|V|^{\alpha+\beta})$ instances of* Group-Cut *satisfying $|A| = \alpha, |B| = \beta$.*

Theorem 1.6 implies that the family of (α, β)-group-cuts has redundancy factor $\Omega(|V|)$. Our work further shows that this bound is existentially tight (attained by some graph G) for all α, β and n.

Theorem 1.6 is proved in Part C. We additionally present similar results regarding two more generalizations of *st*-cuts, and show that for the analogous problems in directed graphs the redundancy factor is at most a constant. The results presented in Part C were published in the *International Workshop in Graph-Theoretic Concepts in Computer Science* [CKK16].

1.4 Batch Sparse Recovery

The fourth and final part of this thesis goes beyond the boundaries of vertex sparsification in graphs, and deals with the broader theme of data compression, and more specifically with a field called *sparse recovery* (or *compressed sensing*).

In sparse recovery the goal is to reconstruct a signal vector $x \in \mathbb{R}^n$ using only *linear measurements*, meaning that x can be accessed only via queries of a linear form $x \mapsto a^t x = \sum_i a_i x_i$. To keep a tab on the number of linear measurements, one usually assumes that the unknown signal $x \in \mathbb{R}^n$ is k-sparse (defined as having at most k non-zero entries, i.e., $\|x\|_0 \leq k$), or that x is close to a k-sparse vector x^*, and then the goal is to construct an estimate to x. The astounding development of a concrete mathematical foundation for the problem by Candès, Tao and Romberg [CRT06] and by Donoho [Don06], over a decade ago, has granted the problem huge attention, see, e.g., [CW08, GI10, EK12, FR13] for exposition and references.

Probably the most well studied version of the problem, called *stable sparse recovery*, is formulated as follows. A *scheme* for dimension n and sparsity bound $k \in [n]$, consists of (a) $t = t(n, k)$ non-adaptive linear measurements, arranged as the rows of a *sensing matrix* $S \in \mathbb{R}^{t \times n}$; and (b) a *recovery algorithm* that uses the measurements vector Sx to output $x' \in \mathbb{R}^n$. Together, these should satisfy, for every signal $x \in \mathbb{R}^n$,

$$\|x - x'\|_p \leq C \min_{k\text{-sparse } x^*} \|x - x^*\|_p \ , \tag{1.2}$$

where $C \geq 1$ and p are some (predetermined) constants. The main goal is to minimize the number of measurements $t = t(n, k)$.

Average Sparsity and Batch Recovery. Although it is well established that many signal types are *typically* sparse, a reasonable sparsity bound k need not hold for all signals, and in some natural scenarios, a good upper bound might simply not be known in advance. Consider, for example, m servers operating in a large network, and denote the frequency vector of the requests made to server $j \in [m]$ by a column vector $A_j \in \mathbb{R}^n$. To perform network analysis, such as anomalies detection and traffic engineering, a designated coordinator needs to examine information from all the m servers, represented as a collective traffic matrix $A = (A_1, \ldots, A_m) \in \mathbb{R}^{n \times m}$. This vast amount of information exceeds communication constraints, and thus the coordinator usually collects only the most relevant data, such as the "heavy" entries from each server [CQZ+14, Yu14]. Since typical traffic vectors have few

heavy entries, it is more plausible to assume the columns have *average sparsity* k (or even $\Theta(k)$), than the significantly stricter assumption that every A_j is k-sparse.

To formalize this scenario as the problem of batch sparse recovery, we will need the following notation. We gather a sequence of column vectors $A_1, \ldots, A_m \in \mathbb{R}^n$ into an $n \times m$ matrix $A := (A_1, \ldots, A_m)$. For $p \in [1, \infty)$, we let $\|A\|_p$ denote the ℓ_p-norm when A is viewed as a "flat" vector of dimension nm, i.e., $\|A\|_p^p := \sum_{ij} |A_{ij}|^p = \sum_{j \in [m]} \|A_j\|_p^p$, for example, $p = 2$ gives the Frobenius norm. Similarly, let $\|A\|_0 := \sum_{j \in [m]} \|A_j\|_0$ denote the sparsity of A, i.e., the number of nonzero entries in A.

Definition 1.7. *In* batch recovery*, the input is a matrix $A \in \mathbb{R}^{n \times m}$ as well as a parameter $k \in [n]$. The goal is to perform linear measurements to columns of A (one column in each measurement), and recover a matrix A' satisfying $\frac{\|A - A'\|_1}{\|A\|_1} \leq C\varepsilon$ for some constant $C \geq 1$, where $\varepsilon = \min_{(km)\text{-sparse } A^*} \frac{\|A - A^*\|_1}{\|A\|_1}$.*

In Part D we show the existence of an adaptive algorithm that recovers a sequence of m vectors with average sparsity k using at most $\tilde{O}(km)$ linear measurements. Formally, we show the following.

Theorem 1.8. *There is a randomized adaptive scheme for batch recovery that, for every input A and k, outputs a matrix A' such that with high probability $\frac{\|A - A'\|_1}{\|A\|_1} = O(\varepsilon)$, where ε is the optimum as in Definition 1.7. The algorithm performs $O(km \log n \log m)$ linear measurements in $O(\log m)$ adaptive rounds, and its output A' is $O(km \log m)$-sparse.*

We prove Theorem 1.8 in Part D and provide some background to previous results and several future research directions in compressed sensing. We additionally show that every non-trivial algorithm for batch sparse recovery must employ adaptivity. Specifically, we show that every non-adaptive randomized algorithm for batch recovery must make $\Omega(mn)$ linear measurements in the worst case. The results presented in Part D have recently been submitted for publication [AKK17].

Part A
Distance Sparsifiers

2 Introduction: The Steiner Point Removal Problem

Let $G = (V, E, w)$ be an edge-weighted graph and let $T = \{t_1, \ldots, t_k\} \subseteq V$ be a designated set of k terminals. Here and throughout, $d_{G,w}(\cdot, \cdot)$ denotes the shortest-path metric between vertices of G according to the weights w. Recall that the *Steiner Point Removal* problem, first formulated by Chan, Xia, Konjevod, and Richa [CXKR06, Section 5], asks to construct on the terminals a new graph $G' = (T, E', w')$ such that (i) distances between the terminals are *distorted* at most by factor $\alpha \geq 1$, formally

$$\forall u, v \in T, \qquad d_{G,w}(u, v) \leq d_{G',w'}(u, v) \leq \alpha \cdot d_{G,w}(u, v);$$

and (ii) the graph G' is (isomorphic to) a minor of G.

Requirement (ii) above expresses structural similarity between G and G'; for instance, if G is planar then so is G'. The SPR formulation above actually came about as a generalization to a result of Gupta [Gup01], which asserts that if G is a tree, then there exists a tree G', which preserves terminal distances with distortion $\alpha = 8$. Later Chan *et al.* [CXKR06] observed that this same G' is actually a minor of the original tree G, and proved the factor of 8 to be tight. The upper bound for trees was later extended by Basu and Gupta [BG08], who achieve distortion $\alpha = O(1)$ for the larger class of outerplanar graphs.

How to construct minors. We now describe a general methodology that is natural for the SPR problem. The first step constructs a minor G' with vertex set T, but without any edge weights, and is prescribed by Definition 2.2. The second step determines edge weights w' that are minimal subject to $d_{G',w'}$ dominating $d_{G,w}$ on the terminals T, as given in Definition 2.3. These steps are illustrated in Figure 2. Our definitions are actually more general (anticipating the technical sections), and consider G' whose vertex set is sandwiched between T and V.

Definition 2.1. *A* partial partition *of a set V is a collection $V_1, \ldots, V_k$ of pairwise disjoint subsets of V, referred to as* clusters*.*

Definition 2.2 (Terminal-Centered Minor)**.** *Let $G = (V, E)$ be a graph with k terminals $T = \{t_1, \ldots, t_k\}$, and let $V_1, \ldots, V_k$ be a partial partition of V, such that each induced subgraph $G[V_j]$ is connected and contains t_j. The graph $G' = (V', E')$ obtained by contracting each $G[V_j]$ into a single vertex that is identified with t_j, is called the* terminal-centered minor *of G induced by $V_1, \ldots, V_k$.*

By identifying the "contracted super-node" V_j with t_j, we may think of the vertex-set V' as containing T and (possibly) some vertices from $V \setminus T$, which implies $V' \subset V$. A terminal-centered minor G' of G can also be described by a mapping $f : V \to T \cup \{\perp\}$, such that

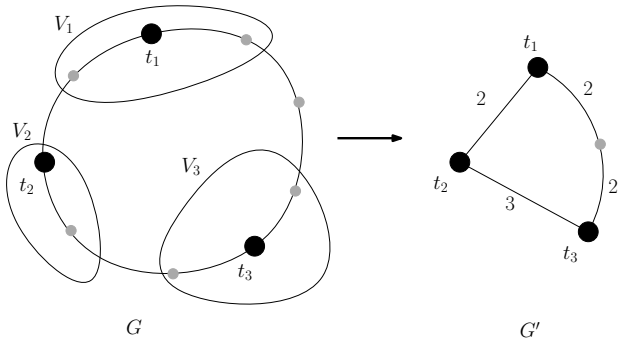


Figure 1: The graph G, a 9-cycle with unit edge weights, is depicted on the left with 3 terminals and disjoint subsets V_1, V_2, V_3. Its terminal-centered minor G' and the standard-restriction edge weights are shown on the right.

$f|_T \equiv \mathrm{id}$ and $f^{-1}(\{t_j\})$ is connected in G for all $j \in [k]$. Indeed, simply let $V_j = f^{-1}(\{t_j\})$ for all $j \in [k]$, and thus $V \setminus (\cup_j V_j) = f^{-1}(\{\perp\})$.

Definition 2.3 (Standard Restriction)**.** *Let $G = (V, E, w)$ be an edge-weighted graph with terminal set T, and let $G' = (V', E')$ be a terminal-centered minor of G. (Recall we view $V' \subset V$.) The* standard restriction *of w to G' is the edge weight w' given by the respective distances in G, formally*

$$\forall (x, y) \in E', \qquad w'_{xy} := d_{G,w}(x, y).$$

This edge weight w' is optimal in the sense that $d_{G',w'}$ dominates $d_{G,w}$ (where it is defined, i.e., on V'), and the weight of each edge $(x, y) \in E'$ is minimal under this domination condition.

2.1 Main Result

Our main result below gives an efficient algorithm that achieves polylog(k) distortion for the SPR problem. Its proof spans Sections 3 and 4, though the former contains the heart of the matter. For the convenience of the readers we restate the main theorem.

Theorem 1.1 *Let $G = (V, E, w)$ be an edge-weighted graph with k terminals $T \subseteq V$. Then there exists a terminal-centered minor $G' = (T, E', w')$ of G that attains distance distortion $O(\log^5 k)$, i.e.,*

$$\forall u, v \in T, \qquad 1 \leq \frac{d_{G',w'}(u, v)}{d_{G,w}(u, v)} \leq O(\log^5 k).$$

Moreover, w' is the standard restriction of w, and G' is computable in randomized polynomial time.

This theorem answers a question of Chan *et al.* [CXKR06]. The only distortion lower bound known for general graphs is a factor of 8 (which actually holds for trees) [CXKR06],

and thus it remains a challenging open question whether $O(1)$ distortion can be achieved in general graphs.

Our proof of Theorem 1.1 begins similarly to the proof of Englert *et al.* [EGK+10], by iterating over the "distance scales" 2^i, going from the smallest distance $d_{G,w}(u,v)$ among all terminals $u,v \in T$, towards the largest such distance. Each iteration i first employs a "stochastic decomposition", which is basically a randomized procedure that finds clusters of V whose diameter is at most 2^i. Then, some clusters are contracted to a nearby terminal, which must be "adjacent" to the cluster; this way, the current graph is a minor of the previous iteration's graph, and thus also of the initial G. After iteration i is executed, we roughly expect "neighborhoods" of radius proportional to 2^i around the terminals to be contracted. As i increases, these neighborhoods get larger until eventually all the vertices are contracted into terminals, at which point the edge weights are set according to the standard restriction. To eventually get a minor, it is imperative that every contracted region is connected. To guarantee this, we perform the iteration i decomposition in the graph resulting from previous iterations' contractions (rather than the initial G), which introduces further dependencies between the iterations.

The main challenge is to control the distortion, and this is where we crucially deviate from [EGK+10] (and differ from all previous work). In their randomized construction of a minor G', for every two terminals $u,v \in T$ it is shown that G' contains a uv-path of *expected length* at most $O(\log k)d_G(u,v)$. Consequently, they design a *distribution* D over minors G', such that the stretch $d_{G'}(u,v)/d_G(u,v)$ between any $u,v \in T$ has expectation at most $O(\log k)$. Note, however, that it is possible that no $G' \in \mathrm{supp}(D)$ achieves a low stretch simultaneously for all $u,v \in T$. In contrast, in our randomized construction of G', the stretch between $u,v \in T$ is polylogarithmic *with high probability*, say at least $1 - 1/k^3$. Applying a simple union bound over the $\binom{k}{2}$ terminal pairs, we can then obtain a single graph G' achieving a polylogarithmic distortion. Technically, these bounds follow by fixing in G a shortest-path P between two terminals $u,v \in T$, and then tracking the execution of the randomized algorithm to analyze how the path P evolves into a uv-path P' in G'. In [EGK+10], the length of P' is analyzed in expectation, which by linearity of expectation, follows from analyzing the case where P consists of a single edge; In contrast, we provide for P a high-probability bound, which inevitably must consider (anti)correlations along the path.

The next section features a new tool that we developed in our quest for high-probability bounds, and which may be of independent interest. For the sake of clarity, we provide below a vanilla version that excludes technical complications such as terminals, strong diameter, and consistency between scales. The proof of Theorem 1.1 actually does require these complications, and thus cannot use the generic form described below.

2.2 Related Work

Applications. Vertex-sparsification, and the "graph compression" approach in general, is obviously beneficial when G' can be computed from G very efficiently, say in linear time,

and then G' may be computed on the fly rather than in advance. But compression may be valuable also in scenarios that require the storage of many graphs, like archiving and backups, or rely on low-throughput communication, like distributed or remote processing. For instance, the succinct nature of G' may be indispensable for computations performed frequently, say on a smartphone, with preprocessing done in advance on a powerful machine.

We do not have new theoretical applications that leverage our SPR result, although we anticipate these will be found later. Either way, we believe this line of work will prove technically productive, and may influence, e.g., work on metric embeddings and on approximate min-cut/max-flow theorems.

Probabilistic SPR. Here, the objective is not to find a single graph $G' = (T, E', w')$, but rather a distribution D over graphs $G' = (T, E', w')$, such that every graph $G' \in \mathrm{supp}(D)$ is isomorphic to a minor of G and its distances $d_{G',w'}$ dominate $d_{G,w}$ (on $T \times T$), and such that the distortion inequalities hold in expectation, that is,

$$\forall u, v \in T, \qquad \mathop{\mathbb{E}}_{G' \sim D}[d_{G',w'}(u,v)] \leq \alpha \cdot d_{G,w}(u,v).$$

This problem, first posed by Chan *et al.* in [CXKR06], was answered by Englert *et al.* in [EGK+10] with $\alpha = O(\log |T|)$.

Distance Preserving Minors. This problem is a relaxation of SPR in which the minor G' may contain a few non-terminals, while preserving terminal distances exactly. Formally, the objective is to find a small graph $G' = (V', E', w')$ such that (i) G' is isomorphic to a minor of G; (ii) $T \subseteq V' \subseteq V$; and (iii) for every $u, v \in T$, $d_{G',w'}(u,v) = d_{G,w}(u,v)$. This problem was originally defined by Krauthgamer, Nguyễn and Zondiner [KNZ14], who showed an upper bound $|V'| \leq O(|T|^4)$ for general graphs, and a lower bound of $\Omega(|T|^2)$ that holds even for planar graphs.

3 Terminal-Centered Minors: Main Construction

This section proves Theorem 1.1 when $\mathcal{D} := \frac{\max_{u,v \in T} d_G(u,v)}{\min_{u,v \in T} d_G(u,v)}$ satisfies the following assumption (the extension to the general case is proved in Section 4).

Assumption 3.1. $\mathcal{D} \leq 2^{k^3}$.

By scaling all edge weights, we may further assume that $\min_{u,v \in T} d_G(u,v) = 1$.

Notation 1. *Let $V_1, \ldots, V_k \subseteq V$. For $S \subseteq [k]$, denote $V_S := \bigcup_{j \in S} V_j$. In addition, denote $V_\perp := V \setminus V_{[k]}$ and $V_{\perp + j} := V_\perp \cup V_j$ for any $j \in [k]$.*

We now present a randomized algorithm that, given a graph $G = (V, E, w)$ and terminals $T \subset V$, constructs a terminal-centered minor G' as stated in Theorem 1.1. The algorithm maintains a partial partition $\{V_1, V_2, \ldots, V_k\}$ of V, starting with $V_j = \{t_j\}$ for all $j \in [k]$.

The sets grow monotonically during the execution of the algorithm. We may also think of the algorithm as if it maintains a mapping $f : V \to T \cup \{\perp\}$, starting with $f(t_j) = t_j$ for all $j \in [k]$ and gradually assigning a value in T to additional vertices, which correspond to the set $V_{[k]}$. Thus, we will also refer to the vertices in $V_{[k]}$ as *assigned*, and to vertices in $V_\perp$ as *unassigned*. The heart of the algorithm is two nested loops (lines 4-9). During every iteration of the outer loop, the inner loop performs k iterations, one for every terminal t_j. Every inner-loop iteration picks a random radius (from an exponential distribution) and "grows" V_j to that radius (but without overlapping any other set) thus removing nodes from $V_\perp$ and assigning them to V_j. Every outer-loop iteration increases the expectation of the radius distribution. Eventually, all nodes are assigned, i.e. $\{V_1, V_2, \ldots, V_k\}$ is a partition of V. Note that the algorithm does not actually contract the clusters at the end of each iteration of the outer loop. However, subsequent iterations grow each V_j only in the subgraph of G induced by the respective $V_{\perp+j}$, which is effectively the same as contracting clusters to their respective terminals at the end of each outer-loop iteration.

Every cluster in the partial partition maintained by the algorithm needs to induce a connected subgraph of G. In particular, the algorithm has to grow the clusters so that they do not overlap. We therefore require the following definition.

Definition 3.2. *For $U \subseteq V$, let $G[U]$ denote the subgraph of G induced by U, with induced edge lengths (i.e. $w|_{E(G[U])}$). For a subgraph H of G with induced edge lengths, a vertex $v \in V(H)$ and $r > 0$, denote $B_H(v, r) := \{u \in V(H) : \ d_H(u, v) \leq r\}$, where d_H is the shortest path metric in H induced by w.*

Input: $G = (V, E, w)$, $T = \{t_1, \ldots, t_k\} \subseteq V$
Output: A partition $\{V_1, V_2, \ldots, V_k\}$ of V.
1: set $b \leftarrow 1 + 1/(45 \log k)$
2: for every $j \in [k]$ set $V_j \leftarrow \{t_j\}$, $r_j = 0$.
3: set $i \leftarrow 0$. *// i is the iteration number of the outer loop.*
4: **while** $V_{[k]} \neq V$ **do**
5: $\quad i \leftarrow i + 1$.
6: $\quad$ **for all** $j \in [k]$ **do**
7: $\quad\quad$ choose independently at random $R_j^i \sim \exp(b^i)$.
8: $\quad\quad$ $r_j \leftarrow r_j + R_j^i$.
9: $\quad\quad$ $V_j \leftarrow V_j \cup B_{G[V_{\perp+j}]}(t_j, r_j)$. *// This is the same as $V_j \leftarrow B_{G[V_{\perp+j}]}(t_j, r_j)$.*
10: **return** $\{V_1, V_2, \ldots, V_k\}$.

Algorithm 3.1: Partitioning V

Claim 3.3. *The following properties hold throughout the execution of the algorithm.*

1. *For all $j \in [k]$, V_j is connected in G, and $t_j \in V_j$.*
2. *For every $j_1, j_2 \in [k]$, if $j_1 \neq j_2$, then $V_{j_1} \cap V_{j_2} = \emptyset$.*

3. *For every outer loop iteration i and every $j \in [k]$, if V_j' denotes the set V_j at the beginning of the i-th iteration (of the outer loop), and V_j'' denotes the set V_j at the end of that iteration, then $V_j' \subseteq V_j''$.*

In what follows, we analyze the stretch in distance between a fixed pair of terminals. We show that with probability at least $1 - O(k^{-5})$, the distance between these terminals in G' is at most $O(\log^5 k)$ times their distance in G. By a union bound over all $\binom{k}{2}$ pairs of terminals, we deduce Theorem 1.1. Let $s, t \in T$, and let P^* be a shortest st-path in G. Due to the triangle inequality, we may focus on pairs which satisfy $V(P^*) \cap T = \{s, t\}$, where $V(P^*)$ is the node set of P^*. We denote $\ell := w(P^*) = d_{G,w}(s, t)$.

3.1 High-Level Analysis

Following an execution of the algorithm, we maintain a (dynamic) path P between s and t. In a sense, in every step of the algorithm, P simulates an st-path in the terminal-centered minor induced by $V_1, V_2, \ldots, V_k$. At the beginning of the execution, set P to be simply P^*. During the course of the execution update P to satisfy two invariants. At every step of the algorithm, the weight of P is an upper bound on the distance between s and t in the terminal centered minor induced by $V_1, \ldots, V_k$ (in that step). In addition, if I is a subpath of P, whose inner vertices are all unassigned, then I is a subpath of P^*. Throughout the analysis, we think of P as directed from s to t, thus inducing a linear ordering of the vertices in P.

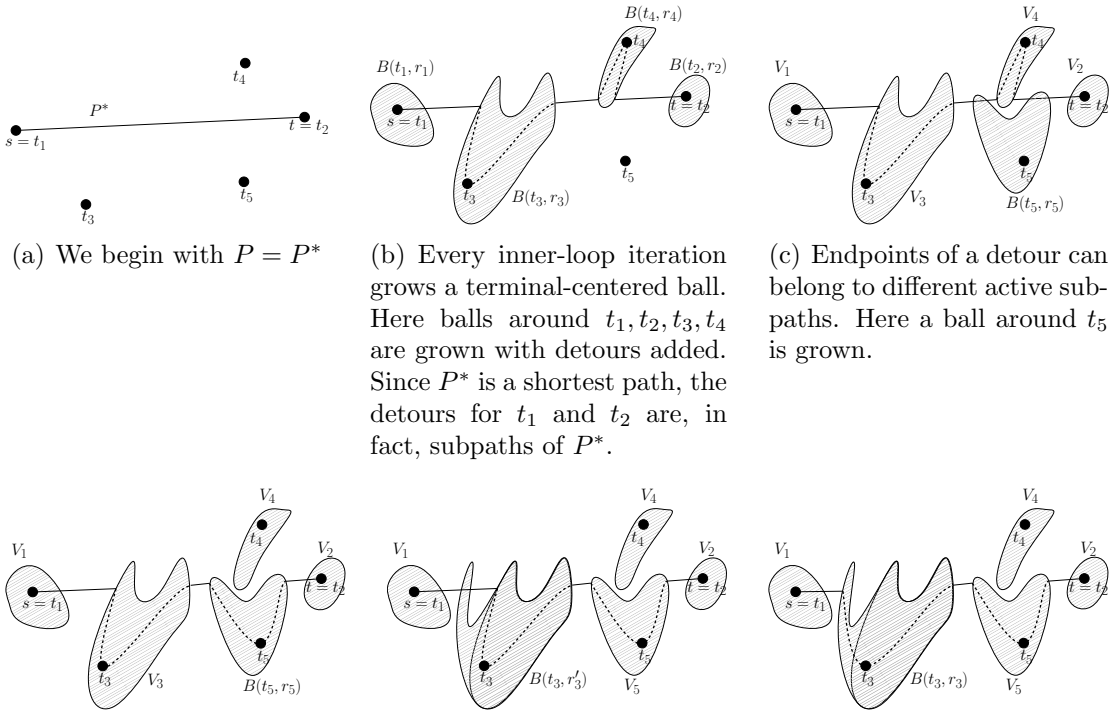
Definition 3.4. *A subpath of P will be called* active *if it is a maximal subpath whose inner vertices are unassigned.*

Note that a single edge whose endpoints are both assigned will not be considered active.

We now describe how P is updated during the execution of the algorithm. Consider line 9 of the algorithm for the i-th iteration of the outer loop, and some $j \in [k]$. We say that the ball $B = B_{G[V_{\perp+j}]}(t_j, r_j)$ *punctures* an active subpath A of P, if there is an inner node of A that belongs to the ball. If B does not puncture any active subpath of P, we do not change P. Otherwise, denote by u, v the first and last unassigned nodes (possibly not in the same active subpath) in $V(P) \cap B$ respectively. Then we do the following.

We replace the entire subpath of P between u and v with a concatenation of a shortest ut_j-path and a shortest t_jv-path that lie in B; this is possible, since $G[B]$ is connected, and $u, t_j, v \in B$. This addition to P will be called a *detour* from u to v through t_j. The process is illustrated in figures 2(a)-2(b). Beginning with P^*, the figure describes the update after the first four balls. Note that the detour might not be a simple path. It is also worth noting that here u and v may belong to different active subpaths of P. For example, in figure 2(c), the new ball punctures two active subpaths, and therefore in figure 2(d), the detour goes from a node in one active subpath to a node in another active subpath. Note that in this case, we remove from P portions which are not active.

It is worth noting that this update process implies that at any given time, there is at most one detour that goes through t_j. If, for some iteration $i' < i$ of the outer loop, and



(a) We begin with $P = P^*$

(b) Every inner-loop iteration grows a terminal-centered ball. Here balls around t_1, t_2, t_3, t_4 are grown with detours added. Since P^* is a shortest path, the detours for t_1 and t_2 are, in fact, subpaths of P^*.

(c) Endpoints of a detour can belong to different active subpaths. Here a ball around t_5 is grown.

(d) Update detour for V_5.

(e) The ball around t_3 is further grown.

(f) Update detour for V_3.

Figure 2: Updating P

---

for some $u', v' \in V(P)$, we added a detour from u' to v' through t_j in iteration i', we keep only one detour through t_j, from the first node between u, u' and to the last between v, v'. For example, in figure 2(e), the ball centered in t_3 punctures an active subpath. Only one detour is kept in figure 2(f).

The total weight of all detours during the execution will be called the *additional weight* to P (ignoring portions of P that are deleted from P). Denote the set of active subpaths of P at the beginning of the i-th iteration of the outer loop by $\mathcal{A}_i$.

Let $V_1^{\text{fin}}, \ldots, V_k^{\text{fin}}$ be the partition returned by the algorithm, let G' the terminal-centered minor induced by that partition, and let w' be the standard restriction of w to G'. Denote by P^{fin} the path obtained at the end of the execution.

Claim 3.5. *At every step of the algorithm the following holds:*

1. *The weight of P is an upper bound on the distance between s and t in the terminal centered minor induced by $V_1, \ldots, V_k$. Moreover, once $\mathcal{A}_i = \emptyset$ (namely, P has no active subpaths), the weight of P is an upper bound on the distance between s and t in the terminal centered minor induced by $V_1^{fin}, \ldots, V_k^{fin}$ (actually, from this point on, $P = P^{fin}$).*

2. *If A is a subpath of P, whose inner points are all in $V_\perp$, then A is a subpath of P^*.*

3. If A_1, A_2 are two different active subpaths of P, they are internally disjoint.

4. $|\mathcal{A}_i| \leq k$ for all i.

Proof. Follows easily by induction on i, j. □

Corollary 3.6. $d_{G',w'}(s,t) \leq w(P^{fin})$.

Let $A \in \mathcal{A}_i$. During the execution of the inner loop, A is either removed from P entirely, or some subpaths of A remain active (perhaps A remains active entirely). Therefore, for every $A' \in \mathcal{A}_{i+1}$, either A' is a non-trivial subpath of A (by non-trivial we mean $|V(A')| \geq 3$), or A' and A are internally disjoint. Therefore there is a laminar structure on $\bigcup_i \mathcal{A}_i$.

We describe this structure using a tree $\mathcal{T}$, whose node set is $\{\langle i, A\rangle |\ A \in \mathcal{A}_i\}$. The root of $\mathcal{T}$ is $\langle 1, P^*\rangle$, and for every i and every $A \in \mathcal{A}_i$, the children of $\langle i, A\rangle$, if any, are all pairs $\langle i+1, A'\rangle$, where $A' \in \mathcal{A}_{i+1}$ is a subpath of A. Whenever we update P we log the weight of the detour by charging it to one of the nodes of $\mathcal{T}$ as follows. Consider a detour from u to v in the i-th iteration of the outer loop for some i. Before adding this detour, u and v are unassigned nodes in P. Because u is unassigned, u is an inner vertex of some active subpath. In either case, there is exactly one active subpath containing u. The weight of the detour is charged to the unique active subpath $A \in \mathcal{A}_i$ such that $u \in A$. For every i and $A \in \mathcal{A}_i$, let $w_{i,A}$ be the total weight charged to $\langle i, A\rangle$. If the node is never charged, the weight of the node is set to 0. Therefore,

$$w(P^{\text{fin}}) \leq w(P^*) + \sum_{\langle i,A\rangle \in \mathcal{T}} w_{i,A}. \tag{3.1}$$

Eqn. (3.1) together with Corollary 3.6 imply that if we show that with high probability, the total weight charged to the tree is at most $O(\log^5 k) \cdot \ell$, we can deduce Theorem 1.1. For the rest of this section, we therefore prove the following lemma.

Lemma 3.7. *With probability at least $1 - O(k^{-5})$, the total weight charged to the tree is at most $O(\log^5 k)\ell$.*

Consider an iteration $i \geq 1$ and an active subpath $A \in \mathcal{A}_i$. Informally, since the distortion is measured relatively to $\ell = w(P^*)$, if the expected radius b^i is small compared to $w(A)$, then with high probability a detour will not add "much" to the distortion, and thus we are more concerned with the opposite case where $w(A)$ is small relative to the current expected radius.

Formally, let $p = 1/100$. An active subpath $A \in \mathcal{A}_i$ will be called *short* if $w(A) \leq pb^i$. Otherwise, A will be called *long*. Notice that $P^* \in \mathcal{A}_1$ is long, and for $i \geq \log_b(\ell/p)$, every $A \in \mathcal{A}_i$ is short.

Definition 3.8. *Let $i > 1$ and let $A \in \mathcal{A}_i$ be a short subpath. Denote by $\mathcal{T}_{i,A}$ the subtree of $\mathcal{T}$ rooted in $\langle i, A\rangle$. Denote the parent of $\langle i, A\rangle$ in $\mathcal{T}$ by $\langle i-1, A'\rangle$ for $A' \in \mathcal{A}_{i-1}$. If A' is long, $\mathcal{T}_{i,A}$ will be called a* short subtree *of $\mathcal{T}$.*

Once an active subpath becomes short (during the course of the iterations), we want all its vertices to be assigned quickly, and by a few detours. For this reason, the height and weight of short subtrees will play an important role in the analysis of the height and weight of $\mathcal{T}$.

To bound the total weight of the tree $\mathcal{T}$, we analyze separately the weights charged to long active subpaths at each level, and the weights of short subtrees rooted at each level. More formally, for every $i \leq \log_b(\ell/p)$, denote by l_i the total weight charged to nodes of the form $\langle i, A\rangle$, where $A \in \mathcal{A}_i$ is a long active subpath. Denote by s_i the total weight charged to short subtrees rooted at the level i of $\mathcal{T}$. For $i \geq \log_b(\ell/p)$, every $A \in \mathcal{A}_i$ is short and thus $\langle i, A\rangle$ belongs to some short subtree rooted at level at most $\log_b(\ell/p)$. Therefore,

$$\sum_{\langle i,A\rangle \in \mathcal{T}} w_{i,A} = \sum_{i=1}^{\log_b(\ell/p)} (l_i + s_i) \ . \tag{3.2}$$

We will first analyze the behavior of short subpaths of active paths, and then use it to bound s_i. To bound the weight of long active paths, we will divide them into short segments, and then sum everything up to bound l_i.

3.1.1 The Effect of a Single Ball on a Short Segment

Let $i_0 \geq 1$ and let I be a subpath of P such that all the inner nodes of I are unassigned in the beginning of the i_0-th iteration of the outer loop, and $w(I) \leq pb^{i_0}$. Note that I is not necessarily maximal with that property, and therefore is not necessarily an active subpath. However, I is a subpath of some (unique) active subpath $A \in \mathcal{A}_{i_0}$. We first consider the effect of a single ball over I, in some iteration $i \geq i_0$.

Fix some $i \geq i_0$, and some $j \in [k]$. Let X denote the number of active subpaths A' such that $V(A') \cap V(I) \neq \emptyset$ at the beginning of the j-th iteration of the inner loop (during the i-th iteration of the outer loop). Note that every such subpath A' is necessarily a subpath of A, due to the laminar structure of active subpaths. Since every such active subpath will add at least one detour before it is completely assigned, we want to show that X is rapidly decreasing. Let X' denote the number of active subpaths A' such that $V(A') \cap V(I) \neq \emptyset$ at the end of the j-th iteration. Denote by B the ball considered in this iteration, namely $B := B_{G[V_{\perp+j}]}(t_j, r_j)$.

Proposition 3.9. *With certainty, $X' \leq X + 1$.*

Proof. Let $A_1, A_2, \ldots, A_X$ be all active subpaths of A which intersect I and are active in the beginning of the j-th iteration ordered by their location on P. For $\alpha \in [X]$ denote by u_α, v_α the first and last unassigned nodes in A_α, respectively. If B does not puncture any of these subpaths, then $X' \leq X < X + 1$ (Note that subpaths of A can still be removed if B punctures active subpaths of P not contained in A). So assume B punctures A_α. Assume first that A_α is the only subpath of P which is active and is punctured by B. Then there are three options: If both $u_\alpha, v_\alpha \in B$, then A_α is replaced and removed entirely from P when adding the detour, and $X' \leq X - 1 < X + 1$. If $u_\alpha \in B$ and $v_\alpha \notin B$, let v' be the last node

in $V(A_\alpha) \cap B$ then the $u_\alpha v'$ segment of A_α is replaced, and the segment $v' v_\alpha$ remains active. Therefore $X' \leq X < X + 1$. The argument is similar, if $u_\alpha \notin B$ and $v_\alpha \in B$. Otherwise, some of the inner portion of A_α is replaced by a non-active path, and both end segments of A_α remain active, therefore $X' = X + 1$. Next, assume the ball punctures several active subpaths of A, and maybe more subpaths of P. Denote by I_α, I_β the first and last subpaths of A punctured by B. Denote by u the first node in $V(I_\alpha) \cap B$, and v the last node in $V(I_\beta) \cap B$. When updating P, the entire subpath of P between u and v is removed. Thus $X' \leq X - (\beta - \alpha + 1) \leq X < X + 1$. □

We now want to show that if some unassigned vertex $v \in V(I)$ gets assigned due to a ball B, i.e., B punctures some active subpath intersecting I, then X is likely to decrease. Recalling that unassigned nodes in P must be in P^*, this goal is stated formally as

$$\Pr[X' \geq X \mid B \cap V(I) \cap V(P^*) \neq \emptyset] \leq p \, .$$

However, this statement is not sufficient for our needs, as it does not imply that with high probability a short active subpath is assigned quickly. Indeed, let I be a short active subpath and suppose no ball punctures any subpath of I for many iterations following i_0; then the detour that will eventually be added to replace a subpath of I might be too long relative to I (as expected radii increase exponentially). Therefore, when arguing that with reasonable probability X decreases, we shall condition on a more refined event, which generalizes the notion of a ball puncturing an active subpath. Loosely speaking, we consider events in which the ball B includes a vertex $v \notin P^*$ (i.e., v is already assigned), and assume there is an unassigned $u \in V(I)$, such that uv is an edge in G (since P^* is a shortest path, this edge uv must be part of P^*), which means there is an active subpath intersecting I adjacent to v (in particular, v is one of its endpoints). By the memoryless property of the exponential distribution, conditioned on $v \in B$, with reasonable probability B covers that subpath. The formal definition follows.

Definition 3.10. *Let I be a subpath of P. We say that a ball B* reaches *I if there is $v \in V(I) \cap B$ such that either $v \in V(P^*)$ is unassigned, or v has an unassigned neighbor which is in $V(I)$.*

Consider again the case where I is active. Then both its endpoints are assigned. Note that the endpoints of I cannot both be assigned to the same terminal (otherwise I would have been removed entirely). By the definition of the balls in the algorithm, $B \subseteq G[V_{\perp+j}]$ and therefore B may reach I and not puncture it if and only if I has exactly one endpoint in V_j. Note that all active subpaths are reached at least twice in every iteration of the outer loop (by the clusters which contain their endpoints). Therefore in every iteration of the outer loop at least two balls reach I with certainty, even though it could be the case that no ball punctures I.

Proposition 3.11. $\Pr[X' \geq X \mid B \text{ reaches } I] \leq p$.

Proof. Assume that B reaches I. Then there exists a node $v \in V(I) \cap B$ such that either v is unassigned, or v has an unassigned neighbor $u \in V(I)$. Let $d = d_{G[V_{\perp+j}]}(t_j, v)$. Assume first that $v \in V(P^*)$ is unassigned. Let A' be the active subpath such that $v \in V(A')$. Following the analysis of the previous proof, if $X' \geq X$, then B punctures exactly one active subpath (namely A') that intersects I and does not cover the part of A' contained in I, or B punctures exactly two such active subpaths and covers neither of them. In either case, B punctures A' and does not cover the part of A' contained in I. Since $w(I) \leq pb^{i_0} \leq pb^i$, the length of A' is at most pb^i. We conclude that $r_j + R^i_j \geq d$, and $r_j + R^i_j < d + pb^i$. If v has an unassigned neighbor $u \in V(I)$, we get the same conclusion, since this again means $r_j + R^i_j \geq r_j \geq d$. By the memoryless property,

$$\Pr[X' \geq X \mid B \text{ reaches } I] \leq \Pr[R^i_j < d - r_j + pb^i \mid R^i_j \geq d - r_j] \leq 1 - e^{-p} \leq p \,.$$

□

3.1.2 The Effect of a Sequence of Balls on a Short Segment

Consider now the first N balls that reach I, starting from the beginning of iteration i_0 of the outer loop, and perhaps during several iterations of that loop. For every $a \in [N]$, let Y_a be the indicator random variable for the event that the a-th ball reaching I decreased the number of active subpaths intersecting I. In these notations, Proposition 3.11 stated that

$$\forall a \in [N], \qquad \Pr[Y_{a+1} = 1 \mid Y_1, \ldots, Y_a] \geq 1 - p \,.$$

Let $Y = \sum_{a \in [N]} Y_a$ and let $Z \sim Bin(N, 1-p)$. Simple induction on N implies the following claim.

Claim 3.12. $\forall k,\ \Pr[Y > k] \geq \Pr[Z > k]$.

Lemma 3.13. *With probability at least $1 - 1/k^{10}$, after $90 \log k$ balls have reached I, there are no active subpaths intersecting I.*

Proof. Assume $N = 70 \log k$. Since whenever $Y_a = 0$, the number of active subpaths increases by at most 1, and whenever $Y_a = 1$, the number of active subpaths decreases by at least 1, if $Y > N/2$, then there are no active subpaths intersecting I. Therefore by the Chernoff bound,

$$\begin{aligned}\Pr[&\text{there are no active subpaths intersecting } I \text{ after } N \text{ balls reach } I] \\ &\geq \Pr[Y > N/2] \geq \Pr[Z > N/2] \geq 1 - 1/k^{10} \,.\end{aligned}$$

□

3.2 The Behavior of Short Subtrees

As stated before, the most crucial part of the proof is to bound the weight and height of short subtrees of $\mathcal{T}$. Let $i_0 > 1$, and let $A \in \mathcal{A}_{i_0}$ be a short subpath such that $\mathcal{T}_{i_0,A}$ is a short subtree of $\mathcal{T}$. Clearly, A' is short for every node $\langle i', A' \rangle$ of $\mathcal{T}_{i_0,A}$. In order to bound the height of $\mathcal{T}_{i_0,A}$ we combine the fact that not too many balls may reach A, with the fact that at least two balls reach A during each iteration of the outer loop.

Claim 3.14. *With probability at least $1 - 1/k^{10}$, the height of $\mathcal{T}_{i_0,A}$ is at most $45 \log k$.*

Proof. In the notations of Lemma 3.13, consider some $i \geq i_0$. If A has an active subpath at the end of the i-th iteration of the outer loop, then at least two times during the i-th iteration an active subpath of A is reached by a ball. After $45 \log k$ iterations of the outer loop, if A has an active subpath, then $N \geq 90 \log k$. By similar arguments to Lemma 3.13,

$$\Pr[\text{The height of } \mathcal{T}_{i_0,A} \text{ is at most } N/2] \geq \Pr[Y > N/2] \geq \Pr[Z > N/2] \geq 1 - 1/k^{10} .$$

□

We denote by $\mathbf{E_1}$ the event that for every i, and every $A \in \mathcal{A}_i$, if $\mathcal{T}_{i,A}$ is a short subtree then after at most $90 \log k$ balls reach an active subpath of A, A has no more active subpaths and in addition, the height of $\mathcal{T}_{i,A}$ is at most $45 \log k$.

Lemma 3.15. $\Pr[\mathbf{E_1}] \geq 1 - 1/k^5$.

Proof. Fix some i, and $A \in \mathcal{A}_i$. Assume that $\mathcal{T}_{i,A}$ is a short subtree. By definition, $\langle i, A \rangle$ has no short ancestor. For $i' = \log_b(\ell/p) \leq \log_b(\mathcal{D}/p)$, P^* itself is short, since $b^{i'} = \ell/p$, and thus all tree nodes in level i' (and lower) are short. Therefore, $i \leq i'$. Since there are at most k nodes in every level of the tree, the number of short subtrees of $\mathcal{T}$ is at most $k \cdot \log_b(\mathcal{D}/p) = k \cdot (\log_b \mathcal{D} + \log_b 100) \leq k^4 \log k + O(k \log k) \leq O(k^4 \log k)$. By the previous lemma, and a union bound over all short subtrees, the result follows. □

Since every node in level $\log_b(\ell/p)$ of the tree belongs to some short subtree, we get the following corollary.

Corollary 3.16. *With probability at least $1 - 2/k^5$, the height of $\mathcal{T}$ is at most $\log_b(\ell/p) + O(\log k) \leq 10 \log_b \mathcal{D}$.*

We denote by $\mathbf{E_2}$ the event that for all $i \leq 10 \log_b \mathcal{D}$ and $j \in [k]$, the radius of the j-th ball of the i-th iteration of the outer loop is at most $O(b^i \log k)$. We wish to prove that $\mathbf{E_2}$ holds with high probability. We will need the following lemma, which gives a concentration bound on the sum of independent exponential random variables.

Lemma 3.17. *Let $X_1, \ldots, X_n$ be independent random variables such that each $X_j \sim \exp(\lambda_j)$ for $\lambda_j > 0$, and denote $\lambda = \max_j \lambda_j$. Then $X = \sum_j X_j$ has expectation $\mu = \mathbb{E}[X] = \sum_j \lambda_j$ and satisfies*

$$\forall \delta > 1, \qquad \Pr[X > (1+\delta)\mu] \leq e^{(1-\delta)\frac{\mu}{2\lambda}} .$$

Proof. We proceed by applying Markov's inequality to the moment generating function (similarly to proving Chernoff bounds). Let $j \in [n]$ and consider $0 \leq t \leq \frac{1}{2\lambda_j}$. Then the moment generating function of X_j is known and can be written as $\mathbb{E}\left[e^{tX_j}\right] = \frac{1}{1-t\lambda_j} \leq 1+2t\lambda_j \leq e^{2t\lambda_j}$. Now set $t = \frac{1}{2\lambda}$ and use Markov's inequality to get

$$\Pr[X > (1+\delta)\mu] = \Pr[e^{tX} > e^{t(1+\delta)\mu}] \leq \frac{\mathbb{E}\left[e^{tX}\right]}{e^{t(1+\delta)\mu}} = \frac{\prod_{j\in[n]}\mathbb{E}\left[e^{tX_j}\right]}{e^{t(1+\delta)\mu}} \leq \frac{e^{2t\mu}}{e^{t(1+\delta)\mu}} = e^{(1-\delta)\frac{\mu}{2\lambda}} .$$

□

Lemma 3.18. $\Pr[\mathbf{E_2}] \geq 1 - 1/k^5$.

Proof. Fix $i \leq 10\log_b \mathcal{D}$ and $j \in [k]$. In the notations of Algorithm 3.1, let r_j be the radius of the jth ball in the ith iteration of the outer loop. Then $r_j = \sum_{i'\leq i} R_j^{i'}$ is the sum of independent exponential random variables. $\mathbb{E}[r_j] = \sum_{i'\leq i} b^{i'} = b\cdot\frac{b^i-1}{b-1} \geq 20b^i\log k$. Applying Lemma 3.17 we get that

$$\Pr[r_j > 40b^i\log k] = \Pr[r_j > 2\mathbb{E}[r_j]\ ] \leq e^{\frac{-\mathbb{E}[r_j]}{b^i}} \leq k^{-10}$$

By assumption 3.1, $10\log_b\mathcal{D} = O(\log\mathcal{D}\log k) = O(k^3\log k)$. Thus by a union bound over all values of i and j in question,

$$\Pr[\forall i,j.\ r_j \leq 40b^i\log k \text{ in the } i\text{th iteration of the outer loop}] \geq 1 - \frac{k\cdot 10\log_b\mathcal{D}}{k^{10}} \geq 1 - \frac{1}{k^5} .$$

□

Summing everything up, we can now bound with high probability the weights of all short subtrees of $\mathcal{T}$.

Claim 3.19. *Conditioned on the events* $\mathbf{E_1}$ *and* $\mathbf{E_2}$*, for every* $i_0 > 1$ *and* $A \in \mathcal{A}_{i_0}$*, if* $\mathcal{T}_{i_0,A}$ *is a short subtree of* $\mathcal{T}$*, then the total weight charged to nodes of* $\mathcal{T}_{i_0,A}$ *is at most* $O(b^{i_0}\log^2 k)$ *with certainty.*

Proof. Conditioned on $\mathbf{E_1}$, at most $90\log k$ detours are charged to nodes of every short subtree and for $i = i_0 + 45\log k$, there are no more active subpaths of A. Conditioned on $\mathbf{E_2}$, the most expensive detour is of weight at most $O(b^{i_0+45\log k}\log k)$, we get that the total weight charged to nodes of the subtree is $90\log k\cdot O(b^{i_0}\cdot b^{45\log k}\cdot\log k) \leq O(b^{i_0}\log^2 k)$, since $b^{45\log k} = O(1)$. □

3.3 Bounding The Weight Of $\mathcal{T}$

We are now ready to bound the total weight charged to the tree. Recall that for every $i \leq \log_b(\ell/p)$ we denoted by l_i the total weight charged to nodes of the form $\langle i, A\rangle$, where $A \in \mathcal{A}_i$ is a long active subpath, and by s_i the total weight charged to short subtrees rooted

in the i-th level of $\mathcal{T}$. Since $\ell \geq 1$, $P^* \in \mathcal{A}_1$ is long. Therefore, $s_1 = 0$. We can therefore rearrange Eqn. (3.2) to get the following.

$$\sum_{\langle i,A\rangle \in \mathcal{T}} w_{i,A} = \sum_{i=1}^{\log_b(\ell/p)} (l_i + s_{i+1}) \; . \tag{3.3}$$

Let $i \leq \log_b(\ell/p)$. Let $A \in \mathcal{A}_i$ be a long active subpath. That is, $w(A) \geq pb^i$. Thinking of A as a continuous path, divide A into $w(A)/(pb^i)$ segments of length pb^i. Some segments may contain no nodes. Let I be a segment of A, and assume I contains nodes (otherwise, no cost is charged to A on account of detours from I). Following Lemma 3.13, we get the following.

Lemma 3.20. *With probability at least $1 - 1/k^{10}$, no more than $90 \log k$ balls reach I.*

Denote by $\mathbf{E_3}$ the event that for every $i \geq \log_b(\ell/k^2)$, and for every long active subpath $A \in \mathcal{A}_i$, in the division of A to segments of length pb^i, every such subsegment is reached by at most $90 \log k$ balls.

Lemma 3.21. $\Pr[\mathbf{E_3}] \geq 1 - 1/k^5$.

Proof. Since for every $i \geq \log_b(\ell/p)$, every $A \in \mathcal{A}_i$ is short, the number of relevant iterations (of the outer loop) is at most $\log_b(\ell/p) - \log_b(\ell/k^2) = \log_b(k^2/p) \leq O(\log^2 k)$. For every $i \geq \log_b(\ell/k^2)$ and every long path $A \in \mathcal{A}_i$, the number of segments of A is at most $w(A)/(pb^i) \leq w(A)/(p\ell/k^2) \leq k^2/p$. Therefore the number of relevant segments for all $i \geq \log_b(\ell/k^2)$ and for all long $A \in \mathcal{A}_i$ is at most $O(k^2 \log^2 k)$ Applying a union bound over all relevant segments the result follows. □

Since $\Pr[\mathbf{E_1}] \geq 1 - 1/k^5$ and $\Pr[\mathbf{E_2}] \geq 1 - 1/k^5$, we get the following corollary.

Corollary 3.22. $\Pr[\mathbf{E_1} \wedge \mathbf{E_2} \wedge \mathbf{E_3}] \geq 1 - O(k^{-5})$

It follows that it is enough for us to prove that conditioned on $\mathbf{E_1}$, $\mathbf{E_2}$ and $\mathbf{E_3}$, with probability 1 the total weight charged to the tree is at most $O(\log^5 k)\ell$.

Lemma 3.23. *Conditioned on $\mathbf{E_2}$ and $\mathbf{E_3}$, $l_i \leq O(b^i k \log k)$. In addition, if $i \geq \log_b(\ell/k^2)$, then $l_i \leq O(\log^2 k) \cdot \ell$ with probability 1.*

Proof. To see the first bound, observe that by the update process of P, at most k detours are added to P during the i-th iteration. Conditioned on $\mathbf{E_2}$, each one of them is of weight at most $O(b^i \log k)$. To see the second bound, let $A \in \mathcal{A}_i$ be a long active subpath. The additional weight resulting from detours from vertices of A is at most the number of segments of A of length pb^i, times the additional weight to each segment. Therefore, the additional weight is at most

$$w_{i,A} \leq w(A)/pb^i \cdot O(\log k) \cdot O(b^i \log k) = O(\log^2 k) \cdot w(A) \; .$$

Since all paths in $\mathcal{A}_i$ are internally disjoint subpaths of P^*, we get:

$$l_i = \sum_{\text{long } A \in \mathcal{A}_i} w_{i,A} \leq \sum_{\text{long } A \in \mathcal{A}_i} O(\log^2 k) \cdot w(A) \leq O(\log^2 k) \cdot \ell \ .$$

□

Lemma 3.24. *Conditioned on events* $\mathbf{E_1}$, $\mathbf{E_2}$ *and* $\mathbf{E_3}$, $s_{i+1} \leq O(b^{i+1} k \log^2 k)$. *In addition, if* $i \geq \log_b(\ell/k^2)$, *then* $s_{i+1} \leq O(\log^3 k) \cdot \ell$.

Proof. Conditioned on $\mathbf{E_1}$ and $\mathbf{E_2}$, we proved in Claim 3.19 that the total weight charged to a short subtree rooted in level $i+1$ is at most $O(b^{i+1} \log^2 k)$ with certainty. Since there are at most k such subtrees, the first bound follows. To get the second bound, note that by the definition of a short subtree, for every short subtree $\mathcal{T}'$ rooted at level $i+1$, the parent of the root of $\mathcal{T}'$ consists of a long active subpath A of level i. Conditioned on $\mathbf{E_3}$, every segment of A is intersected by at most $90 \log k$ balls. Therefore, $\langle i, A \rangle$ can have at most $(w(A)/pb^i) \cdot 90 \log k$ children, and in particular, children consisting of short active subpaths. The cost of a short subtree rooted in the $i+1$ level of $\mathcal{T}$ is at most $O(b^i \log^2 k)$. Thus the total cost of all short subtrees rooted in children of A is bounded by

$$(w(A)/pb^i) \cdot 90 \log k \cdot O(b^i \log^2 k) \leq O(\log^3 k) \cdot w(A) \ .$$

Summing over all (internally disjoint) long subpaths of level i, the result follows. □

We now turn to prove Lemma 3.7.

of Lemma 3.7. Since $\Pr[\mathbf{E_1} \wedge \mathbf{E_2} \wedge \mathbf{E_3}] \geq 1 - O(k^{-5})$, it is enough to show that conditioned on **E1**, **E2** and **E3**, the total weight charged to the tree is at most $O(\log^5 k)\ell$ with certainty. Recall that $\sum_{\langle i,A \rangle \in \mathcal{T}} w_{i,A} = \sum_{i=1}^{\log_b(\ell/p)} (l_i + s_{i+1})$. Following Lemmas 3.23 and 3.24 we get that

$$\begin{aligned}\sum_{i=1}^{\log_b(\ell/k^2)} (l_i + s_{i+1}) &\leq \sum_{i=1}^{\log_b(\ell/k^2)} O(b^i k \log k + b^{i+1} k \log^2 k) \\ &\leq O(k \log^2 k) \sum_{i=1}^{\log_b(\ell/k^2)} b^i = O(k \log^2 k) \frac{b}{b-1} \cdot \frac{\ell}{k^2} = o(1) \cdot \ell \ .\end{aligned}$$

In addition,

$$\begin{aligned}\sum_{i=\log_b(\ell/k^2)+1}^{\log_b(\ell/p)} (l_i + s_{i+1}) &\leq \sum_{i=\log_b(\ell/k^2)+1}^{\log_b(\ell/p)} O(\log^2 k)\ell + O(\log^3 k)\ell \\ &\leq O(\log^3 k)\ell \cdot (\log_b(\ell/p) - \log_b(\ell/k^2)) \\ &\leq O(\log^3 k)\ell \cdot O(\log^2 k) = O(\log^5 k) \cdot \ell \ .\end{aligned}$$

□

4 Terminal-Centered Minors: Extension to General Case

In this section we complete the proof of Theorem 1.1 by reducing it to the special case where Assumption 3.1 holds (which we proved in Section 3). We first outline the reduction, which is implemented using a recursive algorithm, as follows. The algorithm initially rescales edge weights of the graph so that minimal terminal distance is 1. If $\mathcal{D} < 2^{k^3}$ then we apply Algorithm 3.1 and we are done. Otherwise, we construct a set of at most $k-1$ low-diameter balls which are mutually far apart, and whose union contains all terminals. Then, for each of the balls, we apply Algorithm 3.1 on the graph induced by that ball. Each ball is then contracted into a "super-terminal". We apply the algorithm recursively on the resulting graph $\tilde{G}$ with the set of super-terminals as the terminal set. Going back from the recursion, we "stitch" together the output of Algorithm 3.1 on the balls in the original graph with the output of the recursive call on $\tilde{G}$, to construct a partition of V as required. The detailed algorithm and proof of correctness are described in Section 4.1. Before that, we need a few definitions.

Assume that the edge weights are already so that the minimum inter-terminal distance is 1. Denote by D the set of all distances between terminals, rounded down to the nearest powers of 2. Note that $|D| < k^2$. Consider the case $\mathcal{D} > 2^{k^3}$. There must exist $0 \leq m_0 \leq k^3 - k$ such that $D \cap \{2^{m_0}, 2^{m_0+1}, \ldots, 2^{m_0+k}\} = \emptyset$. Define $R := \{(x,y) \in T^2 : \ d_G(x,y) < 2^{m_0}\}$.

Claim 4.1. *R is an equivalence relation.*

Proof. Reflexivity and symmetry of R follow directly from the definition of a metric. To see that R is transitive, let $x, y, z \in T$, and assume $(x,y), (y,z) \in R$. Therefore $d_G(x,y) < 2^{m_0}$ and $d_G(y,z) < 2^{m_0}$. By the triangle inequality, $d_G(x,z) < 2^{m_0+1}$. Since $D \cap \{2^{m_0}, \ldots, 2^{m_0+k}\} = \emptyset$, $d_G(x,z) < 2^{m_0}$, and therefore $(x,z) \in R$. □

For every equivalence class $U \in T/_R$, we pick an arbitrary $u \in U$, and define $\hat{U} = B_G(u, 2^{m_0})$.

Claim 4.2. *$\mathcal{U} := \{\hat{U}\}_{U \in T/_R}$ is a partial partition of V. Moreover, for every $U \in T/_R$, $U \subseteq \hat{U}$, $G[\hat{U}]$ is connected and of diameter at most $2^{m_0+1} < 2^{k^3}$.*

Proof. Let $U \in T/_R$. Let $u \in U$ be such that $\hat{U} = B_G(u, 2^{m_0})$. For every $x \in U$, by the definition of R, $d(x,u) < 2^{m_0}$, and thus $x \in \hat{U}$. Therefore $U \subseteq \hat{U}$. By the definition of a ball, $G[\hat{U}]$ is connected and of diameter at most $2^{m_0+1} < 2^{k^3}$. To see that $\mathcal{U}$ is a partial partition of V, take $U' \in T/_R$ such that $U \neq U'$, and let $u' \in U'$ be such that $\hat{U'} = B_G(u', 2^{m_0})$. Since $(u,u') \notin R$, $d_G(u,u') \geq 2^{m_0}$, and since $D \cap \{2^{m_0}, \ldots, 2^{m_0+k}\} = \emptyset$, $d_G(u,u') \geq 2^{m_0+k+1}$, thus $\hat{U} \cap \hat{U'} = \emptyset$. □

Input: $G = (V, E, w)$, $T = \{t_1, \ldots, t_k\} \subseteq V$
Output: A partition $\{V_1, V_2, \ldots, V_k\}$ of V.

1: rescale the edge weights so that the minimal terminal distance is 1.
2: **if** $\mathcal{D} := \max_{u,v \in T} d_G(u, v) \leq 2^{k^3}$ **then**
3: run Algorithm 3.1, and return its output.
4: **else**
5: define R and $\mathcal{U}$ as above.
6: **for all** $\hat{U} \in \mathcal{U}$ **do**
7: run Algorithm 3.1 independently on $G[\hat{U}]$.
8: contract $\hat{U}$ to a single "super-terminal", maintaining edge weights of all remaining edges.
9: denote the resulting graph $\tilde{G}$.
10: run Algorithm 4.1 recursively on $\tilde{G}$ with the set of super-terminals.
11: **for all** super-terminals $u \in \tilde{G}$ **do**
12: let $u_1, \ldots, u_r$ be the terminals contracted to u in line 8 in an arbitrary order.
13: **for all** vertices v assigned to u in the recursive call **do**
14: assign v to its nearest terminal among $u_1, \ldots, u_r$.
Break ties by the ordering of $u_1, \ldots, u_r$. // *Making sure we construct a minor.*
15: **return** the resulting partition of V.

Algorithm 4.1: Partitioning V - The General Case

4.1 Detailed Algorithm

Our algorithm for the general case of Theorem 1.1 is given as Algorithm 4.1 (which makes calls to Algorithm 3.1). It is clear that this algorithm returns a partition of V. In addition, since every level of recursion decreases the number of terminals in the graph, the depth of the recursion is at most k. During each level of the recursion, Algorithm 3.1 is invoked at most k times. Therefore, Algorithm 3.1 is invoked at most k^2 times, each time on a set of at most k terminals. Note that the result of Lemma 3.7 still applies if k is only an upper bound on the number of terminals, and not the exact number of terminals. Therefore, we get that there exists $C_0 > 0$, such that all $O(k^2)$ times that the algorithm is invoked, it achieves a weight stretch factor of at most $C_0 \log^5 k$, with probability at least $1 - O(k^{-3})$. Applying a union bound, we get that with high probability, the stretch bound is obtained in all invocations of the algorithm. It remains to show that this suffices to achieve the desired stretch factor in G.

Lemma 4.3. *With probability at least $1 - 1/k$, on a graph with $\tau \leq k$ terminals, Algorithm 4.1 obtains a stretch factor of at most $C_0 \log^5 k + \log^5 k \cdot 2^{-k} \sum_{k' \leq \tau} 2(k')^2 \leq 2C_0 \log^5 k$.*

Proof. It is enough to show that conditioned on the event that every invocation of Algorithm 3.1 achieves a stretch factor of at most $C_0 \log^5 k$, the generalized algorithm achieves the desired stretch factor. We prove this by induction on k. For the case $k = 2$, rescaling the weights assures that $\mathcal{D} = 1 \leq 2^{k^3}$, and therefore Algorithm 3.1 is applied on G. The

result follows from the proof of Section 3. Assuming correctness for every $\tilde{k} < k$, we prove correctness for k. Let $s, t \in T$. If $(s,t) \in R$, then s and t are in the same set in $\mathcal{U}$, and by the conditioning, the stretch factor of the distance between s and t is at most $C_0 \log^5 k$. This does not change in steps $4-7$ of the algorithm. Otherwise, Let $\tilde{s}, \tilde{t}$ be the terminals (or super-terminals) associated with s and t in $\tilde{G}$ respectively. Denote $d = d_{G,w}(s,t)$ and $\tilde{d} = d_{\tilde{G},\tilde{w}}(\tilde{s},\tilde{t})$. Denote by $G' = (V', E', w')$ the terminal-centered minor induced by the partition returned by the recursive call, and by $G'' = (V'', E'', w'')$ the terminal-centered minor induced by the partition returned in the final step of the algorithm. Denote $d' = d_{G',w'}(\tilde{s},\tilde{t})$. and $d'' = d_{G'',w''}(s,t)$. Let u be a super terminal on a shortest path P' between $\tilde{s}$ and $\tilde{t}$ in G'. Let P'' be the path obtained from P' in G'' in the following manner. In the place of every super-terminal u in P', originating in some node set $\hat{U}$, we add a path between the corresponding terminals in $\hat{U}$ (based on the terminal-centered minor constructed for $G[\hat{U}]$ in step 3). The edges of P' are also replaced with corresponding edges in G''.

Recall that in G', the weight of every edge is the distance between its endpoints in $\tilde{G}$ (by the definition of a terminal-centered minor). In G'' the weight of every edge is the distance between its endpoints in G. Therefore the weight P' contained at most $k-1$ edges. In G'' the weight of each such edge increases by at most $k2^{m_0}$. In addition, every expansion of a super-terminal adds at most $k2^{m_0}$ to the path. Therefore, $w''(P'') \leq w'(P') + 2k^2 2^{m_0} \leq d' + d \cdot 2k^2 2^{-k}$. By the induction hypothesis

$$d' \leq \left(C_0 \log^5 k + \log^5 k \cdot 2^{-k} \sum_{k' \leq k-1} (k')^2 \right) \tilde{d}.$$

Since $\tilde{d} \leq d$, we get that

$$\begin{aligned} d'' &\leq \left(C_0 \log^5 k + \log^5 k \cdot 2^{-k} \sum_{k' \leq k-1} 2(k')^2 \right) d + d \cdot 2k^2 2^{-k} \\ &\leq \left(C_0 \log^5 k + \log^5 k \cdot 2^{-k} \sum_{k' \leq k} 2(k')^2 \right) d. \end{aligned}$$

This completes the proof of Lemma 4.3 (and in fact also of Theorem 1.1). □

Part B
Metric Decompositions

5 Introduction: Randomized Metric Decompositions

A *randomized decomposition* of the metric (X, d) is a distribution μ over partitions of X, although we usually impose additional requirements in order to obtain more specialized decompositions. One of the most basic versions of randomized decompositions is often referred to as *padded decompositions* (see Definition 1.2). Our results in this section employ a refined definition of so-called padded decompositions, introduced by Abraham *et al.* [AGG+14].

Recall that if Π is a partition of X, then every $S \in \Pi$ is called a *cluster*, and for every $x \in X$, $\Pi(x)$ denotes the unique cluster $S \in \Pi$ such that $x \in S$.

Definition 5.1. *A metric space (X, d) is called* β-decomposable *for $\beta > 0$ if for every $\Delta > 0$ there is a probability distribution μ over partitions of X, satisfying the following properties.*

(a). Diameter Bound: For every $\Pi \in \mathrm{supp}(\mu)$ and $S \in \Pi$, $diam(S) \leq \Delta$.

(b). Padding: For every $x \in X$ and $0 \leq \gamma \leq 1/100$,

$$\Pr_{\Pi \sim \mu}[B(x, \gamma\Delta) \subseteq \Pi(x)\ ] \geq 2^{-\beta\gamma}\ .$$

We note that Definition 1.2, which is more common in the literature (see e.g. [KLMN05, LN04]) is a special case of Definition 5.1 obtained by setting in (b) $\gamma = 1/\beta$. Our results provide constructions that satisfy Definition 5.1, and thus immediately apply also to the more common definition.

5.1 Preliminaries

The Truncated Exponential Distribution. Define the *truncated exponential* with parameters $\lambda > 0$ and $0 \leq \alpha < \beta < \infty$, denoted $\mathrm{Texp}_{[\alpha,\beta]}(\lambda)$, to be distribution given by the probability density function

$$g_{\lambda,[\alpha,\beta]}(x) = \frac{1}{\lambda(e^{-\alpha/\lambda} - e^{-\beta/\lambda})} e^{-x/\lambda} \qquad \forall x \in [\alpha, \beta]\ .$$

Note that this is the pdf of an exponential random variable with mean λ that is conditioned to be in the range $[\alpha, \beta]$.

r-Nets. Given a metric space (V, d), an *r-net* of (V, d) is a set $Y \subseteq V$ satisfying

1. Packing: For all distinct $u, v \in Y$ we have $d(u, v) > r$.
2. Covering: For every $v \in V$, there is some $u \in Y$ such that $d(v, u) \leq r$.

6 Metric Decompositions with Concentration

Bartal [Bar96] proved that every n-point metric is $O(\log n)$-decomposable, and that this bound is tight. We remark that by now there is a rich literature on metric decompositions, and different variants of this notion may involve terminals, or (in a graphical context) connectivity requirements inside each cluster, see e.g. [LS93, Bar96, CKR01, FRT04, Bar04, LN05, GNR10, EGK$^+$10, MN07, AGMW10, KR11].

Degree of separation. Consider a metric space (X, d), and let $P = (x_0, x_1, \ldots, x_\ell)$ be a *shortest path* in X. We denote its length by $d(P) := d(x_0, x_\ell)$, and say that P *meets* a cluster $S \subseteq X$ if $S \cap P \neq \emptyset$. Given a partition Π of X, we define the *degree of separation* of P with respect to Π $Z_P(\Pi)$ as the number of different clusters in the partition Π that meet P. Formally,

$$Z_P(\Pi) := \sum_{S \in \Pi} \mathbb{1}_{\{P \text{ meets } S\}}. \tag{6.1}$$

Throughout, we omit the partition Π when it is clear from the context. When we consider a random partition $\Pi \sim \mu$, the corresponding $Z_P = Z_P(\Pi)$ is actually a random variable. If this distribution μ satisfies requirement (b) of Definition 5.1, then

$$\begin{aligned}
\mathop{\mathbb{E}}_{\Pi\sim\mu}[Z_P] &\leq 1 + \sum_{i\in[\ell]} \Pr_{\Pi\sim\mu}[\Pi(x_{i-1}) \neq \Pi(x_i)] \\
&\leq 1 + \sum_{i\in[\ell]} \Pr_{\Pi\sim\mu}[B(x_{i-1}, d(x_{i-1}, x_i)) \not\subseteq \Pi(x_{i-1})] \\
&\leq 1 + \sum_{i\in[\ell]} (1 - 2^{-\beta d(x_{i-1}, x_i)/\Delta}) \\
&\leq 1 + \sum_{i\in[\ell]} \frac{\beta d(x_{i-1}, x_i)}{\Delta} = 1 + \frac{\beta d(P)}{\Delta}.
\end{aligned} \tag{6.2}$$

But what about the concentration of Z_P? More precisely, can every finite metric be decomposed, such that every shortest path P admits a tail bound on its degree of separation Z_P?

A tail bound. We answer this last question in the affirmative by proving a stronger version of Theorem 1.3 that does involve terminals in Section 6.1. The tail bound over $\Pr[Z_P \geq t]$ in Theorem 1.3 (eqn. (1.1) on page 11) can be compared to a naive estimate that holds for every β-decomposition μ: using (6.2) we have $\mathbb{E}[Z_P] \leq 2$ for every path P with $d(P) \leq \frac{\Delta}{\beta}$, and then by Markov's inequality $\Pr[Z_P \geq t] \leq 2/t$.

We remark that for general metric spaces, it is known that $\beta = O(\log n)$ is tight [Bar96]. However, for requirements (a)-(b) of Definition 5.1 several decompositions are known to have better values of β for special families of metric spaces (e.g. metrics induced by planar graphs [KPR93]). We leave it open whether for these families the bounds of Theorem 1.3 can be improved, say to $\beta = O(1)$.

6.1 A Tail Bound for the Degree of Separation

In this section we prove a slightly stronger result than that of Theorem 1.3, stated as Theorem 6.2 below. Let (X, d) be a metric space, and let $\{t_1, \ldots, t_k\} \subseteq X$ be a designated set of terminals. Recall that a partial partition Π of X is a collection of pairwise disjoint subsets of X. For a shortest path P in X, define $Z_P = Z_P(\Pi)$ using Eqn. (6.1), which is similar to before, except that now Π is a partial partition. We first extend Definition 5.1.

Definition 6.1. *We say that X is* β-terminal-decomposable with concentration *if for every $\Delta > 0$ there is a probability distribution μ over partial partitions of X, satisfying the following properties.*

- Diameter Bound: *For all $\Pi \in \mathrm{supp}(\mu)$ and all $S \in \Pi$, $diam(S) \leq \Delta$.*
- Padding Probability: *For every $x \in X$ and $0 \leq \gamma \leq 1/100$,*

$$\Pr_{\Pi \sim \mu}[\exists S \in \Pi \text{ such that } 0 < |S \cap B(x, \gamma\Delta)| < |B(x, \gamma\Delta)| \ ] \leq 1 - 2^{-\beta\gamma} \ .$$

- Terminal Cover: *For all $\Pi \in \mathrm{supp}(\mu)$, we have $T \subseteq \bigcup_{S \in \Pi} S$.*
- Degree of Separation: *For every shortest path P and every $t \geq 1$,*

$$\Pr_{\Pi \sim \mu}\left[Z_P > t \max\{\tfrac{\beta d(P)}{\Delta}, 1\}\right] \leq O\left(\min\left\{k\beta, \left\lceil \tfrac{\beta d(P)}{\Delta} \right\rceil\right\}\right) \ e^{-\Omega(t)}.$$

Theorem 6.2. *Every finite metric space with k terminals is $(4 \log k)$-terminal-decomposable with concentration.*

For simplicity of notation, we prove the result with cluster diameter at most 2Δ instead of Δ. Fix a desired $\Delta > 0$, and set for the rest of the proof $\lambda := \frac{\Delta}{\log k}$ and $g := g_{\lambda,[0,\Delta]}$ (the pdf of the truncated exponential distributions $\mathrm{Texp}_{[0,\Delta]}(\lambda)$). For $x \in X$ and $r > 0$, we use the standard notation of a closed ball $B(x, r) := \{y \in X : \ d(x, y) \leq r\}$. We define the distribution μ via the following procedure that samples a partial partition Π of X.

1: **for** $j = 1, 2, \ldots, k$ **do**
2: choose $R_j \sim \mathrm{Texp}_{[0,\Delta]}(\lambda)$ independently at random, and let $B_j = B(t_j, R_j)$.
3: set $S_j = B_j \setminus \bigcup_{m=1}^{j-1} B_m$.
4: **return** $\Pi = \{S_1, \ldots, S_k\} \setminus \{\emptyset\}$.

The diameter bound and terminal partition properties hold by construction. The proof of the padding probability is identical to the one in [Bar96, Section 3]. The following two lemmas prove the degree of separation property, which will conclude the proof of Theorem 6.2. Fix a shortest path P in X, and let us assume that $t/2$ is a positive integer; a general $t \geq 1$ can be reduced to this case up to a loss in the unspecified constant.

Lemma 6.3. *If $d(P) < \lambda$, then $\Pr[Z_P > t] \leq 2e^{-\Omega(t)}$.*

Proof. Split the k terminals into $J_{\text{far}} := \{j \in [k] : \ d(t_j, P) > \Delta - 2\lambda\}$ and $J_{\text{near}} := [k] \setminus J_{\text{far}}$. Define random variables $Z_{\text{far}} := \#\{j \in J_{\text{far}} : \ B_j \cap P \neq \emptyset\}$ and $Z_{\text{near}} := \#\{j \in J_{\text{near}} : S_j \cap P \neq \emptyset\}$. Then $Z_P \leq Z_{\text{far}} + Z_{\text{near}}$ and

$$\Pr[Z_P > t] \leq \Pr[Z_{\text{far}} + Z_{\text{near}} > t] \leq \Pr[Z_{\text{far}} > t/2] + \Pr[Z_{\text{near}} > t/2].$$

For every $j \in J_{\text{far}}$,

$$\Pr[B_j \cap P \neq \emptyset] \leq \Pr[R_j \geq \Delta - 2\lambda] = \int_{\Delta-2\lambda}^{\Delta} g(x)dx = \frac{k}{k-1}(e^{-\frac{\Delta-2\lambda}{\lambda}} - e^{-\frac{\Delta}{\lambda}}) \leq \frac{8}{k},$$

and therefore $\mathbb{E}[Z_{\text{far}}] \leq 8$. Since Z_{far} is the sum of independent indicators, by the Chernoff bound, $\Pr[Z_{\text{far}} > t/2] \leq 2^{-t/2}$ for all $t \geq 32e$. For smaller t, observe that $\Pr[Z_{\text{far}} = 0] \geq (1 - 8/k)^k \geq \Omega(1)$, and thus for every $t \geq 1$ we have $\Pr[Z_{\text{far}} > t/2] \leq e^{-\Omega(t)}$.

Next, consider the balls among $\{B_j : \ j \in J_{\text{near}}\}$ that have non-empty intersection with P. Let m denote the number of such balls, and let $j_1 < \ldots < j_m$ denote their indices. In other words, we condition henceforth on an event $\mathcal{E} \in \{0,1\}^{J_{near}}$ that determines whether $R_j \geq d(t_j, P)$ occurs or not for each $j \in J_{\text{near}}$. The indices of coordinates of $\mathcal{E}$ that are equal to 1 are exactly $j_1, \ldots j_m$. For $a \in [m]$, let Y_a be the indicator variable for the event that the ball B_{j_a} does *not* contain P. Note that since $\{R_j\}_{j\in[k]}$ are independent, then so are $\{Y_a\}_{a\in[m]}$. Then

$$\begin{aligned}\Pr\left[Y_a = 1 \mid \mathcal{E}\right] &= \Pr\left[P \not\subseteq B_{j_a} \mid P \cap B_{j_a} \neq \emptyset\right] \\ &\leq \Pr\left[R_j < d(t_{j_a}, P) + \lambda \mid R_j \geq d(t_{j_a}, P)\right] \leq \frac{1 - e^{-1}}{1 - e^{-2}} \leq \frac{3}{4}.\end{aligned}$$

Having conditioned on $\mathcal{E}$, the event $\{Z_{\text{near}} > t/2\}$ implies that $m > t/2$ and moreover, $Y_a = 1$ for all $a \in [t/2]$, and since $\{Y_a\}_{a\in[m]}$ are independent, $\Pr[Z_{\text{near}} > t/2 \mid \mathcal{E}] \leq (3/4)^{t/2} \leq e^{-Ct}$ for an appropriate constant $C > 0$. The last inequality holds for all such events $\mathcal{E}$ (with the same constant $C > 0$), and thus also without any such conditioning.

Altogether, we conclude that $\Pr[Z_P > t] \leq 2e^{-\Omega(t)}$. □

Lemma 6.4. *If $d(P) \geq \lambda$, then* $\Pr[Z_P > td(P)/\lambda] \leq O\left(\min\left\{k \log k, \left\lceil \frac{d(P)}{\lambda} \right\rceil\right\}\right) e^{-\Omega(t)}$.

Proof. Treating P as a continuous path, subdivide it into $r := \lceil d(P)/\lambda \rceil$ segments, say segments of equal length that are (except for the last one) half open and half closed. The induced subpaths $P_1, \ldots, P_r$ of P are disjoint (as subsets of X) and have length at most λ each, though some of subpaths may contain only one or even zero points of X. Writing $Z_P = \sum_{i\in[r]} Z_{P_i}$, we can apply a union bound and then Lemma 6.3 on each P_i, to obtain

$$\Pr[Z_P > td(P)/\lambda] \leq \Pr\left[\exists i \in [r] \text{ such that } Z_{P_i} > t/2\right] \leq O\left(\left\lceil \frac{d(P)}{\lambda} \right\rceil\right) \cdot e^{-\Omega(t)}.$$

Furthermore, for every $j \in [k]$, let $\mathcal{A}_j := \{i \in [r] : P_i \cap B(t_j, \Delta) \neq \emptyset\}$, and since P is a shortest path, $|\mathcal{A}_j| \leq 4\Delta/\lambda = 4 \log k$. Observe that $Z_{P_i} = 0$ (with certainty) for all $i \notin \cup_j \mathcal{A}_j$, hence

$$\Pr[Z_P > td(P)/\lambda] \leq \Pr\left[\exists i \in \cup_{j\in[k]} \mathcal{A}_j \text{ such that } Z_{P_i} > t/2\right] \leq 4k \log k e^{-\Omega(t)} .$$

□

By substituting $\beta = 4 \log k$ and $\lambda = \Delta / \log k$, it is easy to verify that Lemmas 6.3 and 6.4 complete the proof of Theorem 6.2.

7 Metric Decomposition of Path-Separable Graphs

Bartal's proof [Bar96] that every n-point metric space is $O(\log n)$-decomposable, and that this bound is tight for general metric spaces, has motivated an extensive research on restricted families of metric spaces. Notable progress has been made for families defined by topological restrictions, such as shortest-path metrics in graphs excluding a fixed minor [KPR93, FT03, AGG$^+$14] or bounded-genus graphs [LS10, AGG$^+$14] and geometric restrictions, such as a bounded doubling dimension [GKL03] or hyperbolic structure [KL06]. Our work considered metrics induced by graphs of bounded "path separability", which is a blend of topological and geometric restrictions, as defined below. The metric spaces we study in this section arise as shortest-path metrics in (certain) graphs. Specifically, given an undirected graph edge-weighted $G = (V, E, w)$, recall that $d_{G,w}$ denotes the shortest-path metric induced on V by w. If w is clear from the context we will simply denote d_G. Denote by $B_G(u, \varrho)$ the ball of radius $\varrho > 0$ around $u \in V$ in the metric space (V, d_G). We say that a graph G is β-decomposable if the metric space (V, d_G) is β-decomposable.

Shortest-Path Separators. Given a graph $G = (V, E, w)$ and a set $S \subseteq V$, an *S-flap* is a connected component of $G[V \setminus S]$. We say that S is a (balanced) *vertex separator* if every S-flap U has size $|U| \leq |V|/2$. Vertex separators are widely used in divide-and-conquer algorithms. Thorup [Tho04] observed that every planar graph has a vertex separator composed of three shortest paths [1], and used this property to design distance and reachability oracles for planar graphs. Abraham and Gavoille [AG06] extended this notion and defined *path separability*. Intuitively, a graph is path separable if it has a vertex separator composed of a few shortest-paths.

Notation 2. *For sets $X_1, \ldots, X_m$ and $S \subseteq [m]$, we denote $X_S := \bigcup_{j\in S} X_j$.*

Definition 7.1. *[AG06] A graph $G = (V, E, w)$ is called* p-path separable *for $p \in \mathbb{N}$ if there exists $S \subset V$ such that the following holds.*

1. *There exist $P_1, \ldots, P_m \subseteq V$ such that $S = P_{[m]}$ and every P_j is the union of p_j shortest-paths in $G_j := G \setminus P_{[j-1]}$.*

[1] If we only require that every S-flap U has size $|U| \leq 2|V|/3$, then two shortest paths suffice.

2. $\sum_{j\in[m]} p_j \leq p$.

3. *S is a vertex separator, and every S-flap is p-path-separable.*

Abraham and Gavoille showed that for every graph H there is a number $p = p(H)$ such that every graph (V, E) that excludes H as a minor is p-path-separable under every edge weights w. Diot and Gavoille [DG10] proved that every graph of treewidth t is $\lceil (t-1)/2 \rceil$-path-separable with every edge weights w.

7.1 Main Results

Theorem 7.2. *Every p-path-separable graph (V, E, w) is $O(\ln(p \ln |V|))$-decomposable.*

We further note that if, in addition, for every subgraph G' of G, we can find a p-path-separator in polynomial time then we can efficiently sample from the distribution guaranteed in Theorem 7.2.

Combining our theorem with the result of Diot and Gavoille [DG10] we get an upper bound for bounded treewidth graphs.

Corollary 7.3. *Every graph (V, E) of treewidth t with every edge weights w is $O(\ln(t \ln |V|))$-decomposable.*

Previously, no bound was known for p-path-separable graphs other than $O(\log |V|)$ due to Bartal [Bar96]. For graphs of treewidth t the known upper bound is $\beta = O(t)$ due to [AGG$^+$14]. Our decomposition provides a tradeoff between t and $|V|$ and matches or improves all other bounds when $t \geq \ln \ln |V|$. It is conjectured that $\beta = O(\log t)$, which would be tight due to the lower bound of Bartal [Bar96], and our result provide partial evidence in favor of this conjecture.

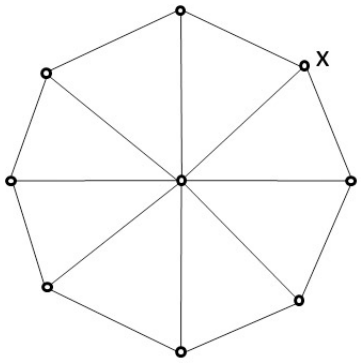
Many known results "interface" the metric only through decompositions, and thus plugging in our decomposition bounds immediately yields new results for the aforementioned families of metric spaces. For example, using a result from [KLMN05] we conclude that every n-vertex graph of treewidth t (with every edge weights w) can be embedded in a Hilbert space with distortion $O(\sqrt{\ln(t \ln n) \cdot \ln n})$, which improves over the known bound $O(\sqrt{t \ln n})$ whenever $t \geq \ln \ln n$.

7.2 Techniques

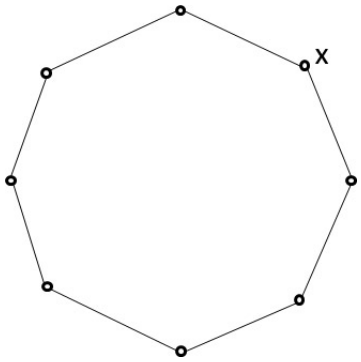
Carving Random Balls. A common approach for constructing a decomposition of a metric (V, d) is to choose a sequence of *centers* $c_1, \ldots, c_k \in V$ and corresponding radii $R_1, \ldots, R_k \leq \Delta/2$, where the choice of centers and/or radii may involve randomization, and then define

$$S_j = \{v \in V : j = \min\{i \in [k] : d(v, c_i) \leq R_i\}\} .$$

Clearly, each S_j has diameter at most Δ. This approach goes back to [Bar96], and has seen many useful variations, for example, randomly ordering the centers [CKR04] or reducing the



(a) Original unit weighted graph G. When all outer-cycle vertices are centers and all $R_i \in [1,2]$, vertex x is threatened by 8 centers.



(b) If balls are carved in this subgraph, x is threatened by only 5 centers.

Figure 3: When carving balls in a subgraph of G, the number of threateners to a vertex is smaller.

number of centers [CKR04, GKL03]. As it turns out, it is enough to bound the number of centers *locally*. More formally, for every $v \in V$ we control the number of centers c_j that threaten v in the sense that $\Pr[d(v,c_j) < R_j + \gamma\Delta] > 0$.

Carving Balls in Subgraphs. Inspired by ideas from [AGG+14], we introduced another modification to the approach described theretofore. In addition to the above, for every center c_j we choose a corresponding subgraph G_j of G, such that $c_j \in V(G_j)$, and define

$$S_j = \{v \in V : j = \min\{i \in [k] : v \in V(G_i) \ \textit{and} \ d_{G_i}(v,c_i) \leq R_i\}\} .$$

Choosing the subgraphs and centers carefully allows us to reduce the number of centers that threaten a vertex v in two ways. The first and more obvious manner is by making sure that $v \in V(G_i)$ for only a few indices i. The second aspect is a bit more subtle. Since distances are considered in subgraphs of G, they might be larger than the corresponding distances in the original graph G, as demonstrated in Figure 3, thus reducing the number of threateners of a vertex v.

Note that we need to ensure that $\{S_j\}_{j\in[k]}$ is indeed a partition of V, i.e. that the balls $\{B_{G_j}(c_j,R_j)\}_{j\in[k]}$ cover all of V.

7.3 Decomposing Path-Separable Graphs

In this section we prove Theorem 7.2. We present a procedure which, given a p-path-separable graph G and a parameter $\Delta > 0$, produces a random partition Π of V. The algorithm works in two phases. The first phase, presented in detail as Algorithm 7.1, constructs a sequence

of centers. This is performed deterministically and by recursion. The algorithm finds a path-separator S of G and chooses a $\Delta/4$-net on each path of the separator to serve as centers. For every center vertex, the algorithm chooses a corresponding subgraph of G. The algorithm is then invoked recursively on every S-flap. The second phase, presented in detail in Algorithm 7.2, samples random radii and carves balls around the centers to obtain a partition of V. The radii are all sampled independently at random from a truncated exponential distribution.

Denote by $\tilde{N}$ the union of all $\Delta/4$-nets throughout the execution of Algorithm 7.1. Note that $\tilde{N}$ is independent of the random radii (in fact, it is constructed deterministically). For sake of clarity, for a center $t \in \tilde{N}$ let G_t, R_t, B_t be the subgraph, radius and ball corresponding to t respectively. To prove that Algorithm 7.2 produces a partition of V, consider $v \in V$. During the execution of Algorithm 7.1 there exists a subgraph G' of G such that Algorithm 7.1 is (recursively) invoked on G' and v is in the path-separator S of G' chosen by the algorithm (in fact, G' is unique). Let P be the path in S such that $v \in P$, let $c \in P$ be the closest net point to v, and let G'' be the respective subgraph, then $P \subseteq G''$. By the definition of a net, $d_P(v,c) \leq \Delta/4$, and therefore $v \in B_P(c, \Delta/4) \subseteq B_{G''}(c, R_c)$ (recall $R_c \geq \Delta/4$ is the radius chosen for c). Therefore $\bigcup_{S \in \Pi} S = V$, and Algorithm 7.2 indeed outputs a partition of V.

To prove the diameter requirement, let $S \in \Pi$, and let $x, y \in S$. Then there exists $t \in \tilde{N}$ such that $S \subseteq B_t \subseteq B_G(t, 2\Delta/5)$ and therefore $d_G(x,y) < \Delta$.

Next, we prove the padding property of the decomposition. Let $x \in V$, and let $0 \leq \gamma \leq 1/80$. Denote $B = B_G(x, \gamma\Delta)$. We say that B is *settled by* $t \in \tilde{N}$ if B_t is the first ball (in order of execution) to have non-empty intersection with B. Therefore, $B \not\subseteq \Pi(x)$ iff B is settled by t and $B_t \cap B \neq B$ for some $t \in \tilde{N}$. Let $\tilde{N}_x := \{t \in \tilde{N} : \Pr[B_t \cap B \neq \emptyset] > 0\}$ be the set of centers that threaten x. In order to bound the size of $\tilde{N}_x$ we consider the execution of Algorithm 7.1. Consider first a single recursion level. Denote the current graph by G', and let $P'_1, \ldots, P'_m$ and $G'_1, \ldots, G'_m$ be as in Definition 7.1. Let $j \in [m]$ and let P be some path in P'_j. Consider the $\Delta/4$-net N picked by the algorithm. Since P is a shortest-path in G'_j,

$$\left|\{t \in N : \Pr[B_t \cap B \neq \emptyset] > 0\}\right| \leq \left|\{t \in N : B_{G'_j}(t, 2\Delta/5) \cap B \neq \emptyset\}\right| .$$

Let $s, t \in N$ be such that $B_{G'_j}(s, 2\Delta/5) \cap B \neq \emptyset$ and $B_{G'_j}(t, 2\Delta/5) \cap B \neq \emptyset$. Since P is a shortest path in G'_j, we get that $d_P(s,t) \leq 2\Delta/5 + 2\gamma\Delta + 2\Delta/5 < \Delta$. Therefore

$$\left|\{t \in N : B_{G'_j}(t, 2\Delta/5) \cap B \neq \emptyset\}\right| \leq \frac{\Delta}{\Delta/4} = 4 .$$

Since in every recursive call of Algorithm 7.1, the number of vertices in the input graph is reduced by at least a factor of $1/2$, the depth of the recursion is at most $\log n$. Every recursion level contains exactly one subgraph that contains x. Since the number of paths in each such subgraph is at most p, we conclude that $|\tilde{N}_x| \leq 4p \log n$. For simplicity, let us further assume this inequality holds with equality, and denote $k := 4p \log n$.

Let $t_1, \ldots, t_k$ be the elements of $\tilde{N}_x$ in the order in which the algorithm considers them. Denote by $\mathcal{E}$ the event that $B \not\subseteq \Pi(x)$, and for every $i \in [k]$, denote by $\mathcal{E}_i$ the event that B was not settled before t_i was considered.

Input: A p-path-separable graph G and a parameter Δ.
Output: A sequence $(c_1, G_1), (c_2, G_2), \ldots$.
1: let $P_1, \ldots, P_m$, $G_1, \ldots, G_m$ be as in Definition 7.1.
2: **for** $j = 1$ to m **do**
3: **for all** paths $P \in P_j$ **do**
4: let $(c_1, c_2, \ldots)$ be a $\Delta/4$-net of $(P, d_G|_P)$ in an arbitrary order.
5: let N_j be the sequence $(c_1, G_j), (c_2, G_j), \ldots$.
6: let N be the concatenation of $N_1, N_2, \ldots, N_m$ in that order.
7: **for all** connected components G' of $G \setminus P_{[m]}$ **do**
8: invoke Choose-Centers(G', Δ) and append the output sequence to N.
9: **return** N.

Algorithm 7.1: Choose-Centers(G, Δ)

Input: A p-path-separable graph G and a parameter Δ.
Output: A partition Π of V.
1: let $(c_1, G_1), (c_2, G_2), \ldots$ be the sequence returned by Choose-Centers(G, Δ).
2: let $\lambda \leftarrow \frac{\Delta}{10 \ln(9p \log n)}$.
3: let $\Pi \leftarrow \emptyset$.
4: **for all** $j \geq 1$ **do**
5: choose $R_j \sim \mathrm{Texp}_{[\Delta/4, 2\Delta/5]}(\lambda)$ independently at random.
6: let $S_j \leftarrow B_{G_j}(c_j, R_j) \setminus \left(\bigcup_{S \in \Pi} S\right)$.
7: let $\Pi \leftarrow \Pi \cup \{S_j\}$.
8: **return** $\Pi \setminus \{\emptyset\}$.

Algorithm 7.2: Decomposing Path-Separable Graphs

Lemma 7.4. $\Pr[\mathcal{E} \mid \mathcal{E}_j] \leq \left(1 + \frac{k-j+1}{k^{3/2}-1}\right)(1 - e^{-20\gamma \ln k})$ *for all* $j \in [k]$.

Proof. Consider some $j \in [k]$. Conditioned on $\mathcal{E}_j$, there are three possible outcomes to the jth round. Either B was not settled also in the jth round, and then (for $j \leq k-1$) the final status of B is left to the following rounds, or B was settled in the jth round, and in that case, either $B \not\subseteq \Pi(x)$, (and thus $\mathcal{E}$ occurred), or $B \subseteq \Pi(x)$. The "bad" event $\mathcal{E}$ can therefore only occur (either in the jth round or some time in the future) if either $j \leq k-1$ and $R_{t_j} < d_G(x, t_j) - \gamma\Delta$ or if $\left|R_{t_j} - d_G(x, t_j)\right| \leq \gamma\Delta$.

Denote $a = \max\{\Delta/4, d(x, t_j) - \gamma\Delta\}$, $b = \min\{d(x, t_j) + \gamma\Delta, 2\Delta/5\}$, then

$$\begin{aligned}\Pr\left[\ \left|R_{t_j} - d_G(x, t_j)\right| \leq \gamma\Delta\ \right] &= \int_a^b \frac{1}{\lambda(k^{-10/4} - k^{-20/5})} e^{-x/\lambda} dx \\ &= \frac{e^{-\frac{\Delta/4}{\lambda}}}{k^{-10/4} - k^{-20/5}}(1 - e^{-\frac{b-a}{\lambda}}) \cdot e^{-\frac{a-\Delta/4}{\lambda}} \ .\end{aligned}$$

Since $b - a \leq 2\gamma\Delta$, then $1 - e^{-\frac{b-a}{\lambda}} \leq 1 - e^{-20\gamma \ln k}$. Substituting $e^{-\frac{\Delta/4}{\lambda}} = k^{-2.5}$ we get that

$$\Pr\left[\ \left|R_{t_j} - d_G(x, t_j)\right| \leq \gamma\Delta\ \right] \leq \frac{k^{3/2}}{k^{3/2} - 1}(1 - e^{-20\gamma \ln k}) \cdot e^{-\frac{a-\Delta/4}{\lambda}} \ . \tag{7.1}$$

Similarly we get that

$$\Pr[R_{t_j} < d_G(x,t_j) - \gamma\Delta] \leq \frac{k^{3/2}}{k^{3/2}-1}(1 - e^{-\frac{a-\Delta/4}{\lambda}}) \; . \tag{7.2}$$

The proof proceeds by induction over $j = k, k-1, \ldots, 1$. As previously noted,

$$\Pr[\mathcal{E} \mid \mathcal{E}_k] \leq \Pr\left[\; \left|R_{t_k} - d_G(x,t_k)\right| \leq \gamma\Delta \;\right] .$$

Plugging (7.1) we get that

$$\Pr[\mathcal{E} \mid \mathcal{E}_k] \leq \frac{k^{3/2}}{k^{3/2}-1}(1 - e^{-20\gamma \ln k}) \cdot e^{-\frac{a-\Delta/4}{\lambda}} \leq \left(1 + \frac{1}{k^{3/2}-1}\right)(1 - e^{-20\gamma \ln k}) \; ,$$

where the last inequality follows from the fact that $a \geq \Delta/4$. Next, let $j \in [k-1]$. Then as previously explained,

$$\Pr[\mathcal{E} \mid \mathcal{E}_j] = \Pr\left[\left|R_{t_j} - d_G(x,t_j)\right| \leq \gamma\Delta\right] + \Pr[R_{t_j} < d_G(x,t_j) - \gamma\Delta] \cdot \Pr[\mathcal{E} \mid \mathcal{E}_{j+1}] \; . \tag{7.3}$$

By plugging (7.1) and (7.2) into (7.3) and using the induction hypothesis, we get that

$$\begin{aligned}
\Pr[\mathcal{E} \mid \mathcal{E}_j] &\leq \frac{k^{3/2}}{k^{3/2}-1}\left(e^{-\frac{a-\Delta/4}{\lambda}} + (1 - e^{-\frac{a-\Delta/4}{\lambda}})\left(1 + \frac{k-j}{k^{3/2}-1}\right)\right)(1 - e^{-20\gamma \ln k}) \\
&= \frac{k^{3/2}}{k^{3/2}-1}\left(1 + \frac{(1 - e^{-\frac{a-\Delta/4}{\lambda}})(k-j)}{k^{3/2}-1}\right)(1 - e^{-20\gamma \ln k}) \\
&\leq \left(1 + \frac{k-j+1}{k^{3/2}-1}\right)(1 - e^{-20\gamma \ln k}) \; ,
\end{aligned}$$

which proves Lemma 7.4. □ □

To complete the proof of Theorem 7.2, note that since $\gamma \leq 1/80$ and for $k \geq 3$,

$$\begin{aligned}
\Pr[\mathcal{E}] = \Pr[\mathcal{E} \mid \mathcal{E}_1] &\leq \left(1 + \frac{k}{k^{3/2}-1}\right)(1 - e^{-20\gamma \ln k}) \\
&\leq \left(1 + e^{-20\gamma \ln k}\right)(1 - e^{-20\gamma \ln k}) = (1 - e^{-40\gamma \ln k}) \; .
\end{aligned}$$

By setting $\beta = \frac{40 \ln k}{\ln 2} = O(\ln(p \ln n))$ we get that $\Pr[B(x, \gamma\Delta) \subseteq \Pi(x) \;] \geq 2^{-\beta\gamma}$, thus proving the last part of Theorem 7.2.

Part C
Cut Sparsifiers for Restricted Families of Cuts

8 Introduction : Counting Cuts in Restricted Cut Families

Our work studies the redundancy factor for the following generalizations of minimum st-cut. Let $G = (V, E, w)$ be an undirected edge-weighted graph.

- Group-Cut : Given two disjoint sets $A, B \subseteq V$ find a minimum (A, B)-cut, i.e., a set of edges of minimum weight that separates every vertex in A from every vertex in B.
- Multiway-Cut : Given $S \subseteq V$ find a minimum-weight set of edges, whose removal ensures that for every $s \neq s' \in S$ there is no s-s' path.
- Multicut : Given $Q \subseteq V \times V$ find a minimum-weight set of edges, whose removal ensures that for every $(q, q') \in Q$ there is no q-q' path.

In order to present our results about the redundancy in these cut problems in a streamlined way, we introduce next the terminology of vertex partitions and demand graphs.

Cut Problems via Demand Graphs. Denote by $\texttt{Par}(V)$ the set of all partitions of V, where a *partition* of V is, as usual, a collection of pairwise disjoint subsets of V whose union is V. Given a partition $\Pi \in \texttt{Par}(V)$ and a vertex $v \in V$, denote by $\Pi(v)$ the unique $S \in \Pi$ satisfying $v \in S$. Given a graph $G = (V, E, w)$, define the function $\texttt{Cut}_G : \texttt{Par}(V) \to \mathbb{R}^{\geq 0}$ to be $\texttt{Cut}_G(\Pi) = \sum_{uv \in E \,:\, \Pi(u) \neq \Pi(v)} w(uv)$. We shall usually omit the subscript G, since the graph will be fixed and clear from the context.

Cut problems as above can be defined by specifying the graph G and a collection D of *demands*, which are the vertex pairs that need to be separated. We can view (V, D) as an (undirected and unweighted) *demand graph*, and by slight abuse of notation, D will denote both this graph and its edges. For example, an instance of Group-Cut is defined by G and demands that form a complete bipartite graph $K_{A,B}$ (to formally view it as a graph on V, let us add that vertices outside of $A \cup B$ are isolated). We say that partition $\Pi \in \texttt{Par}(V)$ *agrees* with D if every $uv \in D$ satisfies $\Pi(u) \neq \Pi(v)$. The optimal cut-value for the instance defined by G and D is given by

$$\texttt{mincut}_G(D) := \min\{\texttt{Cut}_G(\Pi) : \Pi \in \texttt{Par}(V) \text{ agrees with } D\}.$$

Redundancy among Multiple Instances. We study multiple instances on the same graph $G = (V, E, w)$ by considering a family $\mathcal{D}$ of demand graphs. For example, all minimum st-cut instances in a single G corresponds to the family $\mathcal{D}$ of all demands of the form $D =$

$\{(s,t)\}$ (i.e., demand graph with one edge). The collection of optimal cut-values over the entire family $\mathcal{D}$ of instances in a single graph G, is simply $\{\texttt{mincut}(D) : D \in \mathcal{D}\}$. We are interested in the ratio between the size of this collection as a multiset and its size as a set, i.e., with and without counting multiplicities. Equivalently, we define the *redundancy factor* of a family $\mathcal{D}$ of demand graphs to be

$$\text{redundancy}(\mathcal{D}) := \frac{|\mathcal{D}|}{|\{\texttt{mincut}(D) : D \in \mathcal{D}\}|} ,$$

where throughout, $|A|$ denotes the size of A as a *set*, i.e., ignoring multiplicities.

Motivation and Potential Applications. A natural application of the redundancy factor is to construct small data structures that stores all relevant cut values. For the minimum *st*-cut problem, Gomory and Hu were able to collect all the cut values into a tree on the same vertex set V. This tree can easily support fast query time, or a distributed implementation (labeling scheme) [KKKP05].

In addition, large redundancy implies that there is a small collection of cuts that contains a minimum cut for each demand graph. Indeed, first make sure all cut values in G are distinct (e.g., break ties consistently by perturbing edge weights), and then pick for each cut-value in $\{\texttt{mincut}(D) : D \in \mathcal{D}\}$ just one cut that realizes it. This yields a data structure that reports, given demands $D \in \mathcal{D}$, a vertex partition that forms a minimum cut (see more in Section 8.2).

8.1 Main Results

Throughout, we denote $n = |V|$. We use the notation $O_\gamma(\cdot)$ to suppress factors that depend only on γ, and similarly for Ω and Θ.

The Group-Cut problem. In this problem, the demand graph is a complete bipartite graph $K_{A,B}$ for some subsets $A, B \subset V$. We give a tight bound on the redundancy factor of the family of all instances where A and B are of given sizes α and β, respectively. The special case $\alpha = \beta = 1$ is just all minimum *st*-cuts in G, and thus recovers the Gomory-Hu bound (Theorem 1.5). The following is a restatement of Theorem 1.6 using the terminology and notation defined in the previous section.

Theorem 8.1. *For every graph $G = (V, E, w)$ and $\alpha, \beta \in \mathbb{N}$, we have $|\{\texttt{mincut}(K_{A,B}) : |A| = \alpha, |B| = \beta\}| = O_{\alpha,\beta}(n^{\alpha+\beta-1})$, hence the family of (α,β)-group-cuts has redundancy factor $\Omega_{\alpha,\beta}(n)$. Furthermore, this bound is existentially tight (attained by some graph G) for all α, β and n.*

The Multiway-Cut problem. In this problem, the demand graph is a complete graph K_S for some subset $S \subseteq V$. We give a tight bound on the redundancy factor of the family of all instances where S is of a given size $k \geq 2$. Again, the Gomory-Hu bound is recovered by the special case $k = 2$.

Theorem 8.2. *For every graph $G = (V, E, w)$ and for every integer $k \in \mathbb{N}$, we have $|\{\textit{mincut}(K_S) : |S| = k\}| = O_k(n^{k-1})$, hence the family of k-multiway-cuts has redundancy factor $\Omega_k(n)$. Furthermore, this bound is existentially tight for all n and k.*

The MULTICUT **problem.** In this problem, the demand graph is a collection D of demand pairs. We give a tight bound on the redundancy factor of the family of all instances where D is of a given size $k \in \mathbb{N}$. Again, the Gomory-Hu bound is recovered by the special case $k = 1$.

Theorem 8.3. *For every graph $G = (V, E, w)$ and $k \in \mathbb{N}$, we have $|\{\textit{mincut}(D) : D \subseteq V \times V,\ |D| = k\}| = O_k(n^k)$, and hence the family of k-multicuts has redundancy factor $\Omega_k(n^k)$. Furthermore, this bound is existentially tight for all n and k.*

Theorem 8.3 is a bit surprising, since it shows a redundancy factor that is polynomial, rather than linear, in n (for fixed α, β and k), so in general MULTICUT has significantly larger redundancy than GROUP-CUT and MULTIWAY-CUT.

8.2 Extensions and Applications

Our main results above actually apply more generally and have algorithmic consequences, as discussed below briefly.

Terminals Version. In this version, the vertices to be separated are limited to a subset $T \subseteq V$ called *terminals*, i.e., we consider only demands inside $T \times T$. All our results above (Theorems 8.1, 8.2, and 8.3) immediately extend to this version of the problem — we simply need to replace $|V|$ by $|T|$ in all the bounds. As an illustration, the terminals version of Theorem 1.5 states that the $\binom{|T|}{2}$ minimum st-cuts (taken over all $s, t \in T$) attain at most $|T| - 1$ distinct values. (See also [CCPS98, Section 3.5.2] for this same version.) Extending our proofs to the terminals version is straightforward; for example, in Section 9.1 we need to consider polynomials in $|T|$ variables instead of $|V|$ variables.

Data Structures. Flow-equivalent or cut-equivalent trees, such as those constructed by Gomory and Hu [GH61], may be viewed more generally as succinct data structures that support certain queries, either for the value of an optimal cut, or for its vertex-partition, respectively. Motivated by this view, we define data structures, which we call as evaluation schemes, that preprocess an input graph G, a set of terminals T, and a collection of demand graphs $\mathcal{D}$, so as to answer a cut query given by a demand graph $D \in \mathcal{D}$. The scheme has two flavors, one reports the minimum cut-value, the second reports a corresponding vertex-partition. In Section 12 we initiate the study of such schemes, and provide constructions and lower bounds for some special cases.

Functions Different From Cuts. Recall that the value of the minimum st-cut equals $\min\{\texttt{Cut}_G(X, V \setminus X) : X \subseteq V,\ s \in X,\ t \notin X\}$. Cheng and Hu [CH91] extended the Gomory-Hu bound (Theorem 1.5) to a wider class of problems as follows. Instead of a graph

G, fix a ground set V and a function $f : 2^V \to \mathbb{R}$. Now for every $s, t \in V$, consider the optimal value $\min\{f(X) : X \subseteq V,\ |X \cap \{s,t\}| = 1\}$. They showed that ranging over all $s, t \in V$, the number of distinct optimal values is also at most $|V| - 1$. All our results above (Theorems 8.1, 8.2, and 8.3) actually extend to every function $f : \texttt{Par}(V) \to \mathbb{R}$. However, to keep the notation simple, we opted to present all our results only for the function `Cut`.

Directed Graphs. What happens if we ask the same questions for the directed variants of the three problems considered previously? Here, an $s \to t$ cut means a set of edges whose removal ensures that no $s \to t$ path exists. Under this definition, we can construct explicit examples for the directed variants of our three problems above where there is no *non-trivial redundancy*, i.e., the number of distinct cut values is asymptotically equal to the total number of instances. See Appendix 13 for more details.

8.3 Related Work

Gomory and Hu [GH61] showed how to compute a cut-equivalent tree, and in particular a flow-equivalent tree, using $|V| - 1$ minimum st-cut computations on graphs no larger than G. Gusfield [Gus90] has shown a version where all the cut computations are performed on G itself (avoiding contractions). For unweighted graphs, a faster (randomized) algorithm for computing a Gomory-Hu tree which runs in $\tilde{O}(|E| \cdot |V|)$ time was recently given by Bhalgat et al. [BHKP07].

We already mentioned that Cheng and Hu [CH91] extended Theorem 1.5 from cuts to an arbitrary function $f : 2^V \to \mathbb{R}$. They further showed how to construct a flow-equivalent tree for this case (but not a cut-equivalent tree). Benczúr [Ben95] showed a function f for which there is no cut-equivalent tree. In addition, he showed that for directed graphs, even flow-equivalent trees do not exist in general.

Another relevant notion here is that of mimicking networks, introduced by Hagerup, Katajainen, Nishimura, and Ragde [HKNR98]. A mimicking network for $G = (V, E, w)$ and a terminals set $T \subseteq V$ is a graph $G' = (V', E', w')$ where $T \subset V'$ and for every $X, Y \in T$, the minimum (X, Y)-cut in G and in G' have the exact same value. They showed that every graph has a mimicking network with at most $2^{2^{|T|}}$ vertices. Some improved bounds are known, e.g., for graphs that are planar or have bounded treewidth, as well as some lower bounds [CSWZ00, KR13, KR14]. Mimicking networks deal with the Group-Cut problem for all $A, B \subset V$; we consider A, B of bounded size, and thus typically achieve much smaller bounds.

9 Group-Cut: The Case of Complete Bipartite Demands

This section is devoted to proving Theorem 8.1. First we give two proofs, one in Section 9.1 via polynomials and the second in Section 9.2 via matrices, for the bound $|\{\texttt{mincut}(K_{A,B}) : |A| = \alpha, |B| = \beta\}| = O_{\alpha,\beta}(n^{\alpha+\beta-1})$. Then in Section 9.3 we construct examples of graphs for

which this bound is tight. Since $|\{K_{A,B} : |A| = \alpha, |B| = \beta\}| = \binom{n}{\alpha} \cdot \binom{n-\alpha}{\beta} = \Theta_{\alpha,\beta}(n^{\alpha+\beta})$, it follows that the redundancy factor is $\Omega_{\alpha,\beta}(n)$.

9.1 Proof via Polynomials

In this section we show the bound $|\{\texttt{mincut}(K_{A,B}) : |A| = \alpha, |B| = \beta\}| = O_{\alpha,\beta}(n^{\alpha+\beta-1})$ using polynomials. Let $r = \binom{n}{\alpha}\binom{n-\alpha}{\beta}$ and let $\{K_{A_1,B_1}, K_{A_2,B_2}, \ldots, K_{A_r,B_r}\}$ be the set of demand graphs for (α,β)-Group-Cut. For every vertex $v \in V$ we assign a boolean variable denoted by ϕ_v. Given an instance A, B we can assume that the optimal partition only contains two parts, one which contains A and other which contains B, since we can merge other parts into either of these parts.

Fix some $j \in [r]$. Recall that $\Pi = \{U, V \setminus U\} \in \texttt{Par}(V)$ agrees with, i.e., is a feasible solution for, the demand graph K_{A_j,B_j} if and only if the following holds: $\Pi(u) \neq \Pi(v)$ whenever $u \in A_j$ and $v \in B_j$ or vice versa.

Fix arbitrary $a_j \in A_j$ and $b_j \in B_j$. We associate with the demand graph K_{A_j,B_j} the formal polynomial P_j over the variables $\{\phi_v : v \in V\}$

$$P_j = \Pi_{b \in B_j}\left(\phi_{a_j} - \phi_b\right) \cdot \Pi_{a \in A_j \setminus \{a_j\}}\left(\phi_a - \phi_{b_j}\right).$$

Note that P_j is a polynomial of degree $\alpha+\beta-1$. Given $U \subseteq V$, we may think of $\Pi = \{U, V \setminus U\}$ as a vector in $\{0,1\}^n$. We denote by $P_j(\Pi)$ the value of the polynomial P_j (over $\mathbb{F}_2$) when instantiated on Π.

Lemma 9.1. *A partition Π is feasible for the demand graph K_{A_j,B_j} if and only if $P_j(\Pi) \neq 0$*

Proof. Suppose Π is feasible for the demand graph K_{A_j,B_j}. So $\Pi(u) \neq \Pi(v)$ if $u \in A_j, v \in B_j$ or vice versa. Since every term of P_j contains one variable from each of A_j and B_j, it follows that $P_j(\Pi) \neq 0$.

Conversely, assume $P_j(\Pi) \neq 0$. Let $u \in A_j$. Since $\Pi(u) \neq \Pi(b_j)$ and $\Pi(b_j) \neq \Pi(a_j)$ it follows that $\Pi(u) = \Pi(a_j)$. Similarly for every $v \in B_j$, $\Pi(v) = \Pi(b_j)$. Therefore, it follows that $\Pi(u) \neq \Pi(v)$ whenever $u \in A_j$ and $v \in B_j$ or vice versa, i.e., Π is feasible for K_{A_j,B_j}. □

Next we show that the polynomials corresponding to demand graphs with distinct values under `mincut` are linearly independent.

Lemma 9.2. *Reorder the demand graphs such that $\texttt{mincut}(K_{A_1,B_1}) < \ldots < \texttt{mincut}(K_{A_q,B_q})$. Then the polynomials $P_1, \ldots, P_q$ are linearly independent.*

Proof. Let $\Pi_1, \ldots, \Pi_q$ be the optimal partitions for the instances corresponding to the demand graphs $K_{A_1,B_1}, \ldots, K_{A_q,B_q}$ respectively, i.e., for each $i \in [q]$ we have that $\texttt{mincut}(K_{A_i,B_i}) = \texttt{Cut}(\Pi_i)$. Since $\texttt{mincut}(K_{A_i,B_i}) < \texttt{mincut}(K_{A_j,B_j})$ whenever $i < j$, it follows that Π_i is not feasible for the demand graph K_{A_j,B_j} for all $i < j$.

Suppose that the polynomials $P_1, P_2, \ldots, P_q$ are not linearly independent. Then there exist constants $\lambda_1, \ldots, \lambda_q \in \mathbb{R}$ which are not all zero such that $P = \sum_{j \in [q]} \lambda_j P_j$ is the zero

polynomial. We will now show that each of the constants $\lambda_1, \lambda_2, \ldots, \lambda_q$ is zero, leading to a contradiction. Instantiate P on Π_1. Recall that Π_1 is not feasible for any K_{A_i,B_i} with $i \geq 2$. Therefore, by Lemma 9.1, we have that $P_i(\Pi_1) = 0$ for all $i \geq 2$. Therefore $\lambda_1 P_1(\Pi_1) = 0$. Since Π_1 is an (optimal) feasible partition for instance corresponding to K_{A_1,B_1}, applying Lemma 9.1 we get that $P_1(\Pi_1) \neq 0$. This implies $\lambda_1 = 0$. Hence, we have $P = \sum_{2 \leq j \leq q} \lambda_j P_j$ is the zero polynomial. Now instantiate P on Π_2 to obtain $\lambda_2 = 0$ via a similar argument as above. In the last step, we will get that $\lambda_{q-1} = 0$ and hence $P = \lambda_q P_q$ is the zero polynomial. Instantiating on Π_q gives $0 = P(\Pi_q) = \lambda_q P_q(\Pi_q)$. Since Π_q is (optimal) feasible partition for the demand graph K_{A_q,B_q} it follows that $P_q(\Pi_q) \neq 0$, and hence $\lambda_q = 0$. □

Note that each of the polynomials $P_1, P_2, \ldots, P_q$ is contained in the vector space of polynomials with n variables and degree $\leq \alpha + \beta - 1$. This vector space is spanned by $\{\prod_{v \in V} \phi_v^{r_v} : \sum_{v \in V} r_v \leq \alpha + \beta - 1\}$ and therefore is of dimension $\binom{n+(\alpha+\beta-1)}{\alpha+\beta-1} = O_{\alpha,\beta}(n^{\alpha+\beta-1})$. From Lemma 9.2 and the fact that size of any set of linearly independent elements is at most the size of a basis, it follows that $\left| \{\texttt{mincut}(K_{A,B}) : |A| = \alpha, |B| = \beta\} \right| = O_{\alpha,\beta}(n^{\alpha+\beta-1})$.

9.2 Proof via Matrices

In this section we show the (slightly stronger) bound that $|\{\texttt{mincut}(K_{A,B}) : |A| \leq \alpha, |B| \leq \beta\}| = O_{\alpha,\beta}(n^{\alpha+\beta-1})$ using matrices. Let $\texttt{Par}_2(V) \subseteq \texttt{Par}(V)$ be the set of partitions of V into exactly two parts. Let $\mathcal{Q} := \{(A, B) : |A| \leq \alpha, \quad |B| \leq \beta\}$. Consider the matrix $\mathcal{M}$ over $\mathbb{F}_2$ with $|\mathcal{Q}|$ rows (one for each element from $\mathcal{Q}$) and $|\texttt{Par}_2(V)| = 2^n$ columns (one for each partition Π of V into two parts). We now define the entries of $\mathcal{M}$. Given $(A, B) \in \mathcal{Q}$ and $\Pi \in \texttt{Par}_2(V)$, we set $\mathcal{M}_{(A,B),\Pi} = 1$ if and only if the partition $\Pi \in \texttt{Par}_2(V)$ agrees with the demand graph $K_{A,B}$, which is equivalent to saying that $\Pi(u) \neq \Pi(v)$ whenever $u \in A$ and $v \in B$ or vice versa.

Fix a vertex $v_0 \in V$, and consider the set $\mathcal{R} := \{(A, B) \in \mathcal{Q} : v_0 \in A \cup B\}$.

Claim 9.3. *Over $\mathbb{F}_2$, the row space of $\mathcal{M}$ is spanned by the rows corresponding to elements from $\mathcal{R}$*

Proof. Consider $(A, B) \in \mathcal{Q}$ and $\Pi \in \texttt{Par}_2(V)$. If $v_0 \in A \cup B$ then $(A, B) \in \mathcal{R}$. Henceforth we assume that $v_0 \notin A \cup B$. Let

$$L(\Pi) := \mathcal{M}_{(A,B),\Pi} + \sum_{A' \subset A} \mathcal{M}_{(v_0 \cup A', B),\Pi} + \sum_{B' \subset B} \mathcal{M}_{(A, B' \cup v_0),\Pi} ,$$

where addition is over $\mathbb{F}_2$. Note that $(v_0 \cup A', B), (A, B' \cup v_0) \in \mathcal{R}$ for every $A' \subset A$ and $B' \subset B$, and therefore it is enough to show that $L(\Pi) \equiv 0 \pmod 2$.

Assume first that $\mathcal{M}_{(A,B),\Pi} = 1$, i.e. Π agrees with the demand graph $K_{A,B}$. Without loss of generality assume that $\Pi(v_0) = \Pi(a)$ for some $a \in A$. Then we have $\mathcal{M}_{(v_0 \cup A', B),\Pi} = 1$ for all $A' \subset A$, and $\mathcal{M}_{(A, v_0 \cup B'),\Pi} = 0$ for all $B' \subset B$. So, $L(\Pi) = 1 + (2^{|A|} - 1) \equiv 0 \pmod 2$.

Otherwise, we have $\mathcal{M}_{(A,B),\Pi} = 0$. If for every $v \in A \cup B$ it holds that $\Pi(v) \neq \Pi(v_0)$ then

$$L(\Pi) = 0 + \mathcal{M}_{(v_0,B),\Pi} + \mathcal{M}_{(A,v_0),\Pi} = 1 + 1 \equiv 0 \pmod 2 .$$

Hence suppose that there exists $v \in A \cup B$ such that $\Pi(v) = \Pi(v_0)$. Without loss of generality, assume $v \in A$. Then $\mathcal{M}_{(A,B'+v_0),\Pi} = 0$ for all $B' \subset B$. Note that if $A_1, A_2 \subset A$ satisfy $\mathcal{M}_{(v_0 \cup A_1, B),\Pi} = 1 = \mathcal{M}_{(v_0 \cup A_2, B),\Pi}$, then $\mathcal{M}_{(v_0 \cup A_1 \cup A_2, B),\Pi} = 1$. Hence there is an inclusion-wise maximal set $A^* \subset A$ such that $\mathcal{M}_{(v_0 \cup A^*, B),\Pi} = 1$. Since $\mathcal{M}_{(A,B),\Pi} = 0$, we conclude that $A^* \subset A$. Moreover $|A^*| \geq 1$ since $v \in A$. Therefore

$$L(\Pi) = \mathcal{M}_{(A,B),\Pi} + \sum_{A' \subset A} \mathcal{M}_{(v_0 \cup A', B),\Pi} = \sum_{A' \subseteq A^*} \mathcal{M}_{(v_0 \cup A', B),\Pi} = 2^{|A^*|} \equiv 0 \bmod (2)$$

□

An argument similar to Lemma 9.2 shows that rows corresponding to demand graphs with distinct values under `mincut` are linearly independent. Hence, we have $\left| \{\texttt{mincut}(K_{A,B}) : |A| \leq \alpha, |B| \leq \beta\} \right| \leq \texttt{rank}(\mathcal{M}) \leq |\mathcal{R}|$, where the last inequality follows from Claim 9.3. We now obtain the final bound

$$\begin{aligned} |\mathcal{R}| &= \sum_{i \leq \alpha - 1, j \leq \beta} \tbinom{n-1}{i} \cdot \tbinom{n-i-1}{j} + \sum_{j \leq \beta - 1, i \leq \alpha} \tbinom{n-1}{j} \cdot \tbinom{n-j-1}{i} \\ &= \sum_{i \leq \alpha - 1, j \leq \beta} O_{i,j}(n^{i+j}) + \sum_{j \leq \beta - 1, i \leq \alpha} O_{i,j}(n^{i+j}) \\ &= O_{\alpha,\beta}(n^{\alpha+\beta-1}) \end{aligned}$$

9.3 Lower Bound on Number of Distinct Cuts for (α, β)-Group-Cut

We now turn to prove that the bound given in Theorem 8.1 is existentially tight. To this end, we construct an infinite family $G_n^{\alpha,\beta}$ of graphs satisfying $|\{\texttt{mincut}(K_{A,B}) : |A| = \alpha, |B| = \beta\}| \geq \Omega_{\alpha,\beta}(n^{\alpha+\beta-1})$.

Let $n, \alpha, \beta \in \mathbb{N}$ be such that n is odd, and both α and $\beta - 1$ divide $(n-3)/2$. We define a graph $G_n^{\alpha,\beta}$ on n vertices as follows. $G_n^{\alpha,\beta}$ is composed of two graphs that share a common vertex H_n^α and J_n^β defined below.

- H_n^α has $(n+1)/2$ vertices, and is given by α parallel paths $P_1, \ldots, P_\alpha$ between two designated vertices s, t, each path having $(n-3)/2\alpha$ internal vertices. The edge weights are given by distinct powers of 2, monotonically decreasing from s to t. All edges in H_n^α incident on t have ∞ weight (see Figure 4).
- J_n^β has $(n+1)/2$ vertices, and is given by $(\beta - 1)$ parallel paths $Q_1, \ldots, Q_{\beta-1}$, between t and a designated vertex u, each having $(n-3)/2(\beta-1)$ internal vertices. As in H_n^α, edge weights are given by distinct powers of 2, monotonically decreasing from t to u, and all of which are strictly smaller than the weights of H_n^α. All edges in J_n^β incident on u have ∞ weight.

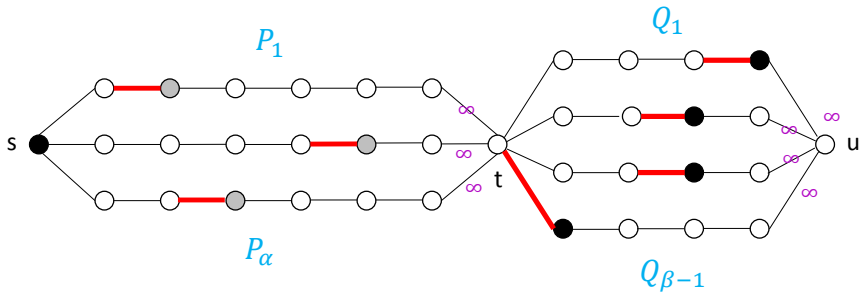


Figure 4: The graph $G_n^{\alpha,\beta}$ used in the lower bound of Section 9.3. The left part of the graph is H_n^α, consisting of α parallel s-t paths. The right part of the graph is J_n^β, consisting of $(\beta-1)$ parallel t-u paths. The gray vertices are in A, and the black ones are in B. The red edges represent the minimum cut for this choice of A and B.

The following claim implies the desired lower bound.

Claim 9.4. $|\{\mathit{mincut}(K_{A,B}) : |A| = \alpha, |B| = \beta\}| \geq \Omega_{\alpha,\beta}(n^{\alpha+\beta-1})$.

Proof. Pick one internal vertex from each P_i for $i \in [\alpha]$ to form A. Similarly for $\beta - 1$ elements in B, we pick one internal vertex from each Q_j for $j \in [\beta]$. In addition, $s \in B$ (as demonstrated in Figure 4). We claim that every such choice of A, B gives a distinct value for the minimum (A, B)-cut.

Indeed, for $i \in [\alpha]$ let a_i be the unique element in $A \cap P_i$. In order to separate A from B, we need to separate a_i from s. This implies that at least one edge on the segment of P_i between s and a_i has to be in the cut. By monotonicity of weights and minimality of the cut, this must be the edge incident to a_i. Similarly, for every $b \in B \setminus \{s\}$, the left edge incident to b must be cut. It can be easily seen (as demonstrated in Figure 4) that this set of edges is also enough to separate A and B.

By the choice of weights, each such cut has a unique value, and therefore $|\{\texttt{mincut}(K_{A,B}) : |A| = \alpha, |B| = \beta\}| \geq ((n-3)/2\alpha)^\alpha((n-3)/2(\beta-1))^{\beta-1} = \Omega_{\alpha,\beta}(n^{\alpha+\beta-1})$. □

10 Multiway-Cut: The Case of Clique Demands

This section is devoted to proving Theorem 8.2. In Section 10.1 we show that for every graph $G = (V, E, w)$ we have $|\{\texttt{mincut}(K_S) : |S| = k\}| = O_k(n^{k-1})$. The proof follows the lines of the proof from Section 9.2. In Section 10.2 we construct an infinite family of graphs for which this bound is tight. Since $|\{K_S : |S| = k\}| = \binom{n}{k} = \Theta_k(n^k)$, it follows that the redundancy factor is $\Omega_k(n)$.

10.1 Upper Bound on Number of Distinct Cuts for k-Multiway-Cut

In this section we show that $|\{\texttt{mincut}(K_S) : |S| = k\}| = O_k(n^{k-1})$. Let $\texttt{Par}_k(V) \subseteq \texttt{Par}(V)$ be the set of partitions of V into exactly k parts. Let $\mathcal{Q} := \{A \subseteq V : |A| = k\}$. Consider

the matrix $\mathcal{M}$ over $\mathbb{F}_2$ with $|\mathcal{Q}|$ rows (one for each element from $\mathcal{Q}$) and $|\texttt{Par}_k(V)|$ columns (one for each partition Π of V into k parts). We now define the entries of $\mathcal{M}$. Given $A \in \mathcal{Q}$ and $\Pi \in \texttt{Par}_k(V)$, we set $\mathcal{M}_{A,\Pi} = 1$ if and only if the partition $\Pi \in \texttt{Par}_k(V)$ agrees with the demand graph K_A. That is if and only if we have $\Pi(u) \neq \Pi(v)$ for every $u, v \in A$ such that $u \neq v$ Fix a vertex $v_0 \in V$, and consider the set $\mathcal{R} := \{A \in \mathcal{Q} : v_0 \in A\}$.

Claim 10.1. *Over $\mathbb{F}_2$, the row space of $\mathcal{M}$ is spanned by the rows corresponding to elements from $\mathcal{R}$*

Proof. Consider $A \in \mathcal{Q}$ and $\Pi \in \texttt{Par}_k(V)$. If $v_0 \in A$ then $A \in \mathcal{R}$. Henceforth we assume that $v_0 \notin A$. Let

$$L(\Pi) := \mathcal{M}_{A,\Pi} + \sum_{a \in A} \mathcal{M}_{\{v_0\} \cup A \setminus \{a\},\Pi}$$

where addition is over $\mathbb{F}_2$. Note that $(\{v_0\} \cup A \setminus \{a\}) \in \mathcal{R}$ for every $a \in A$, and hence it is enough to show that $L(\Pi) \equiv 0 \pmod 2$.

Assume first that $\mathcal{M}_{A,\Pi} = 1$. Since $\Pi \in \texttt{Par}_k(V)$, there is a unique element $a^* \in A$ such that $\Pi(v_0) = \Pi(a^*)$. Then $\mathcal{M}_{\{v_0\} \cup A \setminus \{a\},\Pi} = 0$ for all $a \in A \setminus \{a^*\}$ and $\mathcal{M}_{\{v_0\} \cup A \setminus \{a^*\},\Pi} = 1$. Therefore

$$L(\Pi) := \mathcal{M}_{A,\Pi} + \sum_{a \in A} \mathcal{M}_{\{v_0\} \cup A \setminus \{a\},\Pi} = 1 + 1 \equiv 0 \pmod 2 .$$

Next, assume $\mathcal{M}_{A,\Pi} = 0$. If $\mathcal{M}_{\{v_0\} \cup A \setminus \{a\},\Pi} = 0$ for all $a \in A$, then clearly $L(\Pi) \equiv 0 \pmod 2$. Otherwise, there exists $a' \in A$ such that $\mathcal{M}_{\{v_0\} \cup A \setminus \{a'\},\Pi} = 1$. Since $\mathcal{M}_{A,\Pi} = 0$, it follows that $\Pi(a') \neq \Pi(v_0)$. Therefore there is some $a'' \in A$ such that $\Pi(a') = \Pi(a'')$. Since $\mathcal{M}_{\{v_0\} \cup A \setminus \{a'\},\Pi} = 1$, it follows that $\mathcal{M}_{\{v_0\} \cup A \setminus \{a''\},\Pi} = 1$ and $\mathcal{M}_{\{v_0\} \cup A \setminus \{a\},\Pi} = 0$ for any $a \in A \setminus \{a', a''\}$. Therefore

$$L(\Pi) := \mathcal{M}_{A,\Pi} + \sum_{a \in A} M_{\{v_0\} \cup A \setminus \{a\},\Pi} = 0 + 1 + 1 \equiv 0 \pmod 2 .$$

□

An argument similar to Lemma 9.2 shows that rows corresponding to demand graphs with distinct values under `mincut` are linearly independent. Hence, we have $\left| \{\texttt{mincut}(K_S) : |S| = k\} \right| \leq \texttt{rank}(\mathcal{M}) \leq |\mathcal{R}|$, where the last inequality follows from Claim 10.1. We now obtain the final bound since $|\mathcal{R}| = \binom{n-1}{k-1} = O_k(n^{k-1})$

10.2 Lower Bound on Number of Distinct Cuts for k-Multiway-Cut

We now turn to prove that the bound given in Theorem 8.2 is existentially tight. To this end, we construct an infinite family P_n of graphs satisfying $|\{\texttt{mincut}(K_S) : |S| = k\}| \geq \Omega_k(n^{k-1})$.

For $n \in \mathbb{N}$ consider the path graph $P_n = (V_n, E_n, w)$ where $V_n = \{1, 2, \ldots, n\}$ and $E_n = \{\{i, i+1\} : 1 \leq i \leq n-1\}$. For each $i \in [n-1]$ we denote the edge $\{i, i+1\}$

by e_i and set $w(e_i) = 2^i$. By choice of the weights, it follows that any set of $k-1$ edges from E_n has different weight. Now consider a set $E^* \subseteq E_n$ of exactly $k-1$ edges. We will show that there is an set $S^* \subseteq V_n$ of size k such that E^* is the minimum solution for the k-Multiway-Cut instance with S^* as the input. Let $E^* = \{i_1, i_2, \ldots, i_{k-1}\}$. Then it is easy to see (by choice of weights) that E^* is the minimum weight solution for the instance with input $S^* = \{i_1, i_2 \ldots, i_{k-1}, i_{k-1}+1\}$. Note that $e_{i_{k-1}}$ is an edge implies $i_{k-1} \leq n-1$ and so $i_{k-1}+1$ is well-defined.

This implies that for the path graph P_n (with the weight function specified above) we have $\left|\{\texttt{mincut}(K_S) : |S| = k\}\right| \geq \binom{n-1}{k-1}$, and hence we have $\left|\{\texttt{mincut}(K_S) : |S| = k\}\right| = \Omega_k(n^{k-1})$.

11 Multicut: The Case of Demands with Fixed Number of Edges

This section is devoted to proving Theorem 8.3. In Section 11.1 we show that for every graph G, $|\{\texttt{mincut}(D) : D \subseteq V \times V, |D| = k\}| = O_k(n^k)$. This proof follows the lines of the proof from Section 9.1. Then in Section 11.2 we construct an infinite family of graphs for which this bound is tight. Since $|\{D : D \subseteq V \times V, |D| = k\}| = \binom{\binom{n}{2}}{k} = \Theta_k(n^{2k})$, it follows that the redundancy factor is $\Omega_k(n^k)$.

11.1 Upper Bound on Number of Distinct Cuts for k-Multicut

In this section we show that $|\{\texttt{mincut}(D) : D \subseteq V \times V, |D| = k\}| = O_k(n^k)$. Let $r = \binom{\binom{n}{2}}{k}$ and the set of demand graphs for k-Multicut be $\{D_1, \ldots, D_r\}$. For every vertex $v \in V$ we assign a variable denoted by ϕ_v which can take values from $[n]$. Fix some $j \in [r]$. Recall that $\Pi \in \texttt{Par}(V)$ agrees with (or equivalently, is feasible for) the demand graph D_j if and only if $u - v \in D_j$ implies $\Pi(u) \neq \Pi(v)$.

We associate with the demand graph D_j the formal polynomial

$$P_j = \Pi_{u-v \in D_j} \Big(\phi_u - \phi_v\Big)$$

Note that P_j is a polynomial of degree k. We denote by $P_j(\Pi)$ the value of the polynomial P_j (over $\mathbb{F}_2$) when instantiated on Π. The proof of the next lemma is straightforward.

Lemma 11.1. *A partition Π is feasible for the instance corresponding to the demand graph D_j if and only if $P_j(\Pi) \neq 0$*

Proof. Suppose Π is feasible for the instance corresponding to the demand graph D_j. So for every edge $u - v \in D_j$ we have $\Pi(u) \neq \Pi(v)$ and hence $P_j(\Pi) \neq 0$.

Conversely, assume $P_j(\Pi) \neq 0$. Hence, for each edge $u - v \in D_j$ we have $\Pi(u) \neq \Pi(v)$ which is exactly the condition for Π being feasible for the demand graph D_j. □

The proof of the following lemma is very similar to that of Lemma 9.2, and hence we omit the details.

Lemma 11.2. *Reorder the demand graphs such that* $\texttt{mincut}(D_1) < \texttt{mincut}(D_2) < \ldots < \texttt{mincut}(D_q)$. *Then the polynomials* $P_1, P_2, \ldots, P_q$ *are linearly independent.*

Note that each of the polynomials $P_1, P_2, \ldots, P_q$ is contained in the vector space of polynomials with n variables and degree $\leq k$. It is well known that the size of a basis of this vector space is $\binom{n+k}{k} = O_k(n^k)$. From Lemma 11.2 and the fact that size of any set of linearly independent elements is at most the size of a basis, it follows that $\Big|\{\texttt{mincut}(D) : D \subseteq V \times V, |D| = k\}\Big| = O_k(n^k)$.

11.2 Lower Bound on Number of Distinct Cuts for k-Multicut

We now turn to prove that the bound given in Theorem 8.3 is existentially tight. To this end, we construct an infinite family PM_n of graphs satisfying $|\{\texttt{mincut}(D) : D \subseteq V \times V, |D| = k\}| \geq \Omega_k(n^k)$.

For *even* $n \in \mathbb{N}$ consider the graph $PM_n = (V_n, D_n, w)$ which is a perfect matching on n vertices. Let $V_n = \{1, 2, \ldots, n\}$ and $D_n = \{\{2i-1, 2i\} : 1 \leq i \leq n/2\}$. For each $i \in [n]$ we denote the edge $\{2i-1, 2i\}$ by d_i and set $w(d_i) = 2^i$. By choice of the weights, it follows that any set of k edges from D_n has different total weight. Now consider a set $D^* \subseteq D_n$ of exactly k edges. We will now show that there is an set $D^{**} \subseteq D_n$ of size k such that D^* is the minimum solution for the k-Multicut instance whose demand graph is D^{**}. It is easy to see that D^* is the only solution (and hence of minimum weight too) for the instance whose demand graph is D^*. Hence taking $D^{**} = D^*$ suffices.

This implies that for the perfect matching graph $PM_n = (V_n, D_n)$ (with the weight function specified above) we have $|\{\texttt{mincut}(D) : D \subseteq V_n \times V_n, |D| = k\}| \geq \binom{n/2}{k} = \Omega_k(n^k)$.

12 Evaluation Schemes: Constructing Succinct Data Structures

Gomory and Hu [GH61] showed that for every undirected edge-weighted graph $G = (V, E, w)$ there is a tree $\mathcal{T} = (V, E', w')$ that represents the minimum st-cuts exactly both in terms of the *cut-values* and in terms of their *vertex-partitions*. The common terminology for the first property, probably due to Benczúr [Ben95], is to say that $\mathcal{T}$ is flow-equivalent to G. The second property, which is actually stronger, says that $\mathcal{T}$ is cut-equivalent to G. [2]

These (flow-equivalent and cut-equivalent) trees can be viewed more generally as succinct data structures that support certain queries, either for the value of an optimal cut, or for its vertex-partition.

[2] We say that $\mathcal{T}$ is flow-equivalent to G when for every $s, t \in V$ the minimum st-cut value in $\mathcal{T}$ is exactly the same as in G. We say it is cut-equivalent to G when every vertex-partitioning that attains a minimum st-cut in $\mathcal{T}$, also attains a minimum st-cut in G.

Motivated by this view, we define two types of data structures, which we call a *flow-evaluation scheme* and a *cut-evaluation scheme* (analogously to the common terminology in the literature). These schemes are arbitrary data structures (e.g., need not form a tree), and address the terminals version (of some cut problem). Both of these schemes, first preprocess an input that consists of a graph $G = (V, E, w)$, a terminals set $T \subset V$, and a collection of demand graphs $\mathcal{D}$. The preprocessed data can then be used (without further access to G) to answer a cut query given by a demand graph $D \in D$. The answer of a *flow-evaluation scheme* is the corresponding minimum cut-value $\texttt{mincut}(D)$. The answer of a *cut-evaluation scheme* is a vertex-partition that attains this cut-value $\texttt{mincut}(D)$. Formally, we define the following.

Definition 12.1. *A* flow-evaluation scheme *is a data structure that supports the following two operations.*

1. *Preprocessing P, which gets as input a graph $G = (V, E, w)$, a set of terminals $T \subseteq V$ and a family $\mathcal{D}$ of demand graphs on T and constructs a data structure $P(G, T, \mathcal{D})$.*
2. *Query Q, which gets as input $D \in \mathcal{D}$ and uses $P(G, T, \mathcal{D})$ to output $\texttt{mincut}(D)$. Note that Q has no access to G itself.*

A cut-evaluation scheme *also supports a third operation.*

3. *Query Q', which gets as input $D \in \mathcal{D}$ and uses $P(G, T, \mathcal{D})$ to output a partition $\Pi \in \texttt{Par}(T)$ which attains $\texttt{mincut}(D)$.*

We provide below some constructions and lower bounds for flow-evaluation schemes and cut-equivalent schemes, for the three cut problems in question, i.e. Group-Cut, Multiway-Cut and Multicut. Note that all our upper bounds are for the stronger version of cut-evaluation schemes, and our lower bound is for the weaker version of flow-evaluation schemes for the $(2, 1)$-Group-Cut problem. In order to measure bit complexity for the bounds, we assume hereafter that all weights are integers.

12.1 Upper Bounds for Cut-Evaluation Schemes

The next theorem follows from the *terminal version* of Theorem 8.1. Similar results also hold for the Multiway-Cut and Multicut problems; the proofs follow in the same manner from Theorem 8.2 and Theorem 8.3 respectively.

Theorem 12.2. *There exists a cut-evaluations scheme such that for every graph $G = (V, E, w)$, a set of terminals $T \subseteq V$ and $\alpha, \beta \in \mathbb{N}$, for the family $\mathcal{D} = \{K_{A,B} : A, B \subseteq T,\ |A| = \alpha,\ |B| = \beta\}$ of demand graphs at most $O_{\alpha,\beta}(|T|^{\alpha+\beta-1} \cdot (|T| + \log W))$ bits are stored, where $W = \sum_{e \in E} w(e)$, and such that the query time is $O_{\alpha,\beta}(|T|^{\alpha+\beta-1})$.*

Proof. Let q be the number of distinct values attained by demand graphs on T. Applying the upper bound of Theorem 8.1 adjusted for the terminals version, we get that $q \leq O_{\alpha,\beta}(|T|^{\alpha+\beta-1})$. Order the demand graphs $\{K_{A_1,B_1}, \ldots, K_{A_q,B_q}\}$ such that $\texttt{mincut}(K_{A_i,B_i}) <$

$\texttt{mincut}(K_{A_j,B_j})$ for all $i < j$. For every $j \in [q]$ we associate with K_{A_j,B_j} a partition $\Pi_j \in \texttt{Par}_2(T)$ which attains $\texttt{mincut}(K_{A_j,B_j})$. Representing Π_j as a bit vector of length $|T|$, we list all values in an increasing order. Each entry of this structure is of size $O(|T|+\log W)$. Therefore the size of the evaluation scheme is at most $O_{\alpha,\beta}(|T|^{\alpha+\beta-1} \cdot (|T|+\log W))$ bits. To see the bound of the query time, note that given $A, B \subseteq T$ such that $|A| = \alpha$ and $|B| = \beta$, the evaluation scheme holds $\texttt{mincut}(K_{A,B})$ and a partition Π that attains it. Moreover, going over the list, $\texttt{mincut}(K_{A,B})$ is the first value in the list for which Π agrees with the associated partition. □

12.2 Lower Bound on Flow-Evaluation Schemes for $(2,1)$-Group-Cut

Next we use an information-theoretic argument which shows a lower bound on the storage required by any flow-evaluation scheme for $(2,1)$-Group-Cut. Since a cut-evaluation scheme is stronger than a flow-evaluation scheme, this lower bound immediately extends also to cut-evaluation schemes.

Theorem 12.3. *For every $n \geq 3$, a flow-evaluation scheme for $(2,1)$-Group-Cut on graphs with n terminals (in which $T = V$) and with edge-weights bounded by a polynomial in n requires storage of $\Omega(n^2 \log n)$ bits.*

Let $3 \leq n \in \mathbb{N}$, let $\mathcal{D} = \{K_{A,B} : A, B \subset [n],\ |A| = 2,\ |B| = 1\}$ and let $G = (V, E)$ be the complete graph on $V = T = [n]$. For every $j \in [n-1]$, $(j, j+1) \in E$ are referred to as *path edges*, and the rest of the edges are referred to as *fork* edges as demonstrated in Figure 5.

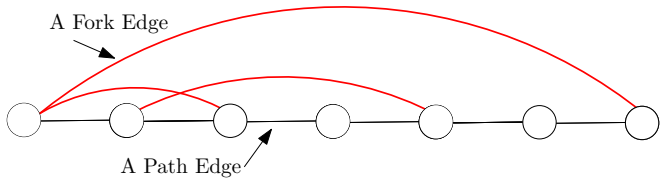


Figure 5: Edge types in G. Black edges correspond to *path* edges, while red edges correspond to *fork* edges (we illustrate only a few fork edges for simplicity).

To prove Theorem 12.3 we assign random edge weights to the graph in the range $\{1, \ldots, 2n^5\}$. We then show that given query access to $\{\texttt{mincut}(D) : D \in \mathcal{D}\}$ we can recover all edge weights. This, in turn, implies that we can recover at least $\Omega(n^2 \log n)$ bits, and thus implies Theorem 12.3.

We assign edge weights to the edges of G as follows:

(P1) For every $j \in [n]$, $w(j, j+1)$ is chosen uniformly at random in $[2(n-j)n^4, (2(n-j)+1)n^4]$. This ensures that the weights of the path edges are non-increasing as we go from left to right. Moreover, whenever $j > i$,

$$w(j, j+1) \leq (2(n-j)+1)n^4 = (2n-2j+1)n^4 < (2n-2i)n^4 - n^4 \leq w(i, i+1) - n^4 .$$

(P2) The weights of all fork edges are chosen uniformly at random from $\{0, 1, 2, \ldots, n-1\}$. Thus the total weight to all fork edges is at most n^3. Note that this is strictly smaller than the difference between weights of any two path edges (which is at least n^4).

Claim 12.4. *Let $1 \leq i < j \leq n$, then* $\texttt{mincut}(K_{\{1,j\},\{i\}}) = \sum_{e \in E : |e \cap \{i,\ldots,j-1\}|=1} w_e$.

Proof. By Property (P2) the total weigtht of all the fork edges is at most n^3, which is less than the difference between weights of any two path edges (which is at least n^4). Hence, the value $\texttt{mincut}(K_{\{1,j\},\{i\}})$ is determined only by which path edges we choose.

Note that the minimum cut separating $\{1, j\}$ from $\{i\}$ needs to pick at least one edge each from the paths $1 - i$ and $i - j$. By Property (P1) the path edges have non-increasing weights going from left to right, and hence the two cheapest edges on the paths $1-2-\ldots-i$ and $i-(i+1)-\ldots-j$ are $(i-1, i)$ and $(j-1, j)$ respectively. Therefore, it follows that $\texttt{mincut}(K_{\{1,j\},\{i\}}) = \texttt{Cut}(C, \overline{C})$, where $C = \{i, i+1, \ldots, j-1\}$. By definition of $\texttt{Cut}(C, \overline{C})$ it follows that

$$\texttt{mincut}(K_{\{1,j\},\{i\}}) = \sum_{e \in E : |e \cap \{i,\ldots,j-1\}|=1} w_e$$

□

Lemma 12.5. *Given access to queries Q as in Definition 12.1, we can recover w.*

Lemma 12.5 implies that we can recover $\Omega(n^2 \log n)$ random bits given access to queries Q as in Definition 12.1, and thus implies Theorem 12.3.

Proof. For every $1 \leq i < j \leq n$, we show that we can recover w_{ij}. The proof continues by induction on $j - i$, starting with the case $j = i + 1$. By Claim 12.4, we get that $\texttt{mincut}(K_{\{1,i+1\},\{i\}}) = \sum_{e \in E : i \in e} w_e$ and $\texttt{mincut}(K_{\{1,i+2\},\{i+1\}}) = \sum_{e \in E : i+1 \in e} w_e$. Therefore

$$\texttt{mincut}(K_{\{1,i+1\},\{i\}}) + \texttt{mincut}(K_{\{1,i+2\},\{i+1\}}) = 2w_{i,i+1} + \sum_{e \in E : |e \cap \{i,i+1\}|=1} w_e \tag{12.1}$$

In addition we have $\texttt{mincut}(K_{\{1,i+2\},\{i\}}) = \sum_{e \in E : |e \cap \{i,i+1\}|=1} w_e$. Plugging this into (12.1) we get that

$$w_{i,i+1} = \frac{1}{2}\left(\texttt{mincut}(K_{\{1,i+1\},\{i\}}) + \texttt{mincut}(K_{\{1,i+2\},\{i+1\}}) - \texttt{mincut}(K_{\{1,i+2\},\{i\}})\right) ,$$

and therefore we can recover $w_{i,i+1}$. Next, let $1 \leq i < j \leq n$, and assume that we can recover w_{pq} for all $1 \leq p < q \leq n$ such that $q - p < j - i$. In addition, we assume that $j < n$. The proof is similar for the case $j = n$. From Claim 12.4 we get that

$$\texttt{mincut}(K_{\{1,j+1\},\{i\}}) = \sum_{e \in E : |e \cap \{i,\ldots,j\}|=1} w_e = \sum_{k=i}^{j} \sum_{m \notin \{i,\ldots,j\}} w_{mk} \tag{12.2}$$

Let $i < k < j$, then by Claim 12.4

$$\sum_{m \notin \{i,\ldots,j\}} w_{mk} = \sum_{e \in E : k \in e} w_e - \sum_{m \in \{i,\ldots,j\}} w_{mk} = \texttt{mincut}(K_{\{1,k+1\},\{k\}}) - \sum_{m \in \{i,\ldots,j\}} w_{mk} \ .$$

For every $m \in \{i, \ldots, j\}$, $|k - m| < j - i$, and therefore we can recover w_{mk}. It follows that we can recover $\sum_{m \notin \{i,\ldots,j\}} w_{mk}$. Plugging this into (12.2), and rearranging we get that

$$\texttt{mincut}(K_{\{1,j+1\},\{i\}}) - \sum_{i<k<j} \sum_{m \notin \{i,\ldots,j\}} w_{mk} = \sum_{m \notin \{i,\ldots,j\}} w_{im} + \sum_{m \notin \{i,\ldots,j\}} w_{jm} \tag{12.3}$$

It remains to show that we can recover w_{ij} assuming we can recover $\sum_{m \notin \{i,\ldots,j\}} w_{im} + \sum_{m \notin \{i,\ldots,j\}} w_{jm}$. Applying Claim 12.4 once more, we get that

$$\sum_{m \notin \{i,\ldots,j\}} w_{im} = \sum_{e \in E : i \in e} w_e - \sum_{m \in \{i,\ldots,j\}} w_{im} = \texttt{mincut}(K_{\{1,i+1\},\{i\}}) - w_{ij} - \sum_{m \in \{i,\ldots,j-1\}} w_{im} \ ,$$

and similarly

$$\sum_{m \notin \{i,\ldots,j\}} w_{jm} = \texttt{mincut}(K_{\{1,j+1\},\{j\}}) - w_{ij} - \sum_{m \in \{i+1,\ldots,j\}} w_{im} \ .$$

By the induction hypothesis, we can recover w_{im} for all $m \in \{i, \ldots, j-1\}$ and w_{jm} for all $m \in \{i+1, \ldots, j\}$, and therefore we can recover w_{ij}. □

This completes the proof of Theorem 12.3. We note that similar arguments give a lower bound of $\Omega(n^3 \log n)$ by allowing weights which are exponential in n^3. Details omitted.

13 No Non-trivial Redundancy for Directed Graphs

In this section, we consider the directed versions of the three cut problems considered in this context, viz. the Group-Cut, Multiway-Cut, Multicut. Note that in directed graphs, an $s \to t$ cut is a set of edges whose removal ensures there is no $s \to t$ path. We construct an infinite family of graphs which have no non-trivial redundancy for any of these problems, i.e., the number of distinct cut values is asymptotically equal to the total number of instances.

Let $n \in \mathbb{N}$, and let X, Y be two disjoint n-element sets. Consider the graph $G_n := K_{X \to Y}$, which is the orientation of the complete bipartite graph $K_{X,Y}$ obtained by orienting each edge from a vertex of X towards a vertex of Y. We assign edge weights in G_n in such a manner that every set of edges has distinct weight (for example, we may assign each edge a distinct power of 2).

GROUP-CUT **in Directed Graphs.** In the directed version of the (α, β)-GROUP-CUT problem, given sets $A, B \subseteq V$ such that $|A| = \alpha$ and $|B| = \beta$, we want to find a set of edges of minimum weight whose removal ensures there is no path from any vertex of A to any vertex of B. The total number of demand graphs for G_n is therefore $|\{K_{A\to B} : |A| = \alpha, |B| = \beta\}| = \binom{2n}{\alpha} \cdot \binom{2n-\alpha}{\beta} = \Theta_{\alpha,\beta}(n^{\alpha+\beta})$. Let $A \subseteq X, B \subseteq Y$ be such that $|A| = \alpha, |B| = \beta$. A minimum (A, B)-cut must include all the edges of $K_{A\to B}$, and furthermore these edges are enough. By choice of weights, the value of the minimum (A, B)-cut is unique. This implies that in G_n we have $|\{\texttt{mincut}(K_{A\to B}) : |A| = \alpha, |B| = \beta\}| \geq \binom{n}{\alpha} \cdot \binom{n}{\beta} = \Omega_{\alpha,\beta}(n^{\alpha+\beta})$, and hence there is no non-trivial redundancy.

We note that for the special case of *st*-cuts in directed graphs (i.e. (α, β)-GROUP-CUT with $\alpha = \beta = 1$), Lacki *et al.* [LNSW12] show that there exists an infinite family of *planar* graphs, which have no non-trivial redundancy. That is, for every graph in the family there are $\Omega(|V|^2)$ distinct *st*-cuts.

MULTIWAY-CUT **in Directed Graphs.** In the directed version of the k-MULTIWAY-CUT problem, given a k-element set $S \subseteq V$ we want to find a set of edges of minimum weight whose removal ensures there is no $s \to s'$ path for any distinct $s, s' \in S$. Let $k \leq n$ be even. The number of instances S in G_n is $\binom{2n}{k} = \Theta_k(n^k)$. Let $A \subseteq X, B \subseteq Y$ be such that $|A| = |B| = k/2$, and let $S = A \cup B$. Then $|S| = k$, and therefore constitutes an instance for the directed k-MULTIWAY-CUT problem. For this instance any multiway cut must include all the edges of $K_{A\to B}$, and furthermore these edges are enough. Therefore the number of distinct cut values for the directed MULTIWAY-CUT problem is at least $|\{\texttt{mincut}(K_{A\to B}) : A \subseteq X, B \subseteq Y, |A| = |B| = k/2\}| = \binom{n}{k/2} \cdot \binom{n}{k/2} = \Omega_k(n^k)$.

MULTICUT **in Directed Graphs.** In the directed version of the k-MULTICUT problem, given a set of demands $D \subseteq V \times V$ such that $|D| = k$, we want to find a set of edges of minimum weight whose removal ensures there is no $s \to s'$ path for any $(s, s') \in D$. The total number of such demand graphs for G_n is $|\{D \subseteq V \times V : |D| = k\}| = \binom{2n(2n-1)}{k} = \Theta_k(n^{2k})$. Let D be a set of edges of G_n such that $|D| = k$. It is evident that the minimum set of edges satisfying D is, in fact, D itself. By the choice of weights, the number of distinct values is at least $|\{\texttt{mincut}(D) : D \subseteq V \times V, |D| = k\}| \geq \binom{n^2}{k} = \Omega_k(n^{2k})$.

14 Future Directions

A natural direction for future work is to construct better data structures for the problems discussed in this part. Our tight bounds on the number of distinct cut values (redundancy factor) yield straightforward schemes with improved storage requirement, as described in Section 12. But one may potentially improve these schemes in several respects. First, our storage requirement exceeds by a factor of $|T|$ the number of distinct cut values. The latter (number of distinct cut values) may be the "right bound" for storage requirement, and it is thus important to prove storage lower bounds; we only proved this for $(2, 1)$-GROUP-CUT. Second, it would be desirable to achieve fast query time, say sublinear in $|T|$ or perhaps even

constant. Third, one may ask for a distributed version of the data structure (i.e., a labeling scheme) that can report the same cut values; this would extend the known results [KKKP05] for minimum st-cuts. All these improvements require better understanding of the structure of the optimal vertex partitions (those that attain minimum cut values). Such structure is known for minimum st-cuts, where the Gomory-Hu tree essentially shows the existence of a family of minimum st-cuts, one for each $s, t \in V$, which is laminar.

Another very interesting question is to explore approximation to the minimum cut, i.e., versions of the above problems where we only seek for each instance a cut within a small factor of the optimal. For instance, the cut values of (α, β)-Group-Cut can be easily approximated within factor $\alpha \cdot \beta$ using Gomory-Hu trees, which requires storage that is linear in $|T|$, much below the aforementioned "right bound" $|T|^{\alpha+\beta-1}$. Can a better approximation be achieved using similar storage?

Part D
Batch Sparse Recovery

15 Introduction : Stable vs. Batch Sparse Recovey

We introduce a *batch* version of sparse recovery, where the goal is to construct $A'_1, \ldots, A'_m \in \mathbb{R}^n$ that estimate a sequence of unknown signals $A_1, \ldots, A_m \in \mathbb{R}^n$, using linear measurements, each involving exactly one signal vector, under an assumption of *average sparsity*. More precisely, given a matrix $A \in \mathbb{R}^{n \times m}$ and a parameter $k \in [n]$, the goal is to perform linear measurements to columns of A, one column in each measurement, and recover a matrix A' satisfying $\frac{\|A-A'\|_1}{\|A\|_1} \leq C\varepsilon$ for some constant $C \geq 1$, where

$$\varepsilon = \min_{(km)\text{-sparse } A^*} \frac{\|A - A^*\|_1}{\|A\|_1} .$$

The special case $m = 1$ (i.e. $A = A_1 \in \mathbb{R}^n$) is known as *stable sparse recovery*. Recall that a *scheme* for dimension n and sparsity bound $k \in [n]$, consists of (a) $t = t(n, k)$ non-adaptive linear measurements, arranged as the rows of a *sensing matrix* $S \in \mathbb{R}^{t \times n}$; and (b) a *recovery algorithm* that uses the measurements vector $SA = SA_1 \in \mathbb{R}^t$ to output an estimate $A'_1 \in \mathbb{R}^n$. Together, these should satisfy, for every signal $A \in \mathbb{R}^n$,

$$\|A - A'\|_p \leq C \min_{k\text{-sparse } A^*} \|A - A^*\|_p , \tag{15.1}$$

which is called an ℓ_p/ℓ_p guarantee, where $C \geq 1$ and p are some (predetermined) constants. The above is often called the "for all" model (or sometimes a uniform or deterministic guarantee). In contrast, in the "for each" model, the scheme (and in particular the matrix S) is random, and for every signal $A \in \mathbb{R}^n$ with high probability, the ℓ_p/ℓ_p guarantee (15.1) holds. The main goal is to minimize the number of measurements $t = t(n, k)$. For $C = \Theta(1)$, known schemes achieve the ℓ_1/ℓ_1 guarantee (15.1) in the "for all" model using $t = O(k \log(n/k))$ measurements [CRT06], and this bound is asymptotically tight even in the "for each" model [BIPW10]. A similar upper bound on t is known also for ℓ_2/ℓ_2 guarantee in the "for each" model [GLPS12], and again this bound is tight [PW11]. There are many other variants which focus on different considerations, for instance, when the measurements may be constructed adaptively, there is a scheme with $t = O(k \log \log(n/k))$ [IPW11]. In some schemes, the output A^* is itself sparse, and/or C can be made arbitrarily close to 1 (at the cost of increasing t).

15.1 Our Results

Our main result gives an adaptive algorithm that recovers a sequence of m vectors with average sparsity k using at most $\tilde{O}(km)$ linear measurements. Known sparse recovery lower bounds imply that this bound is asymptotically tight up to logarithmic factors, even in

the simple case where every vector in the sequence is exactly k-sparse. Our result and the ensuing discussion are stated in terms of the ℓ_1 norm, although all our results extend to the ℓ_2 norm in a standard manner. We repeat the formal statement for the convenience of the readers.

Theorem 1.8 *There is a randomized adaptive scheme for batch recovery that, for every input A and k, outputs a matrix A' such that with high probability $\frac{\|A-A'\|_1}{\|A\|_1} = O(\varepsilon)$, where ε is the optimum as in Definition 1.7. The algorithm performs $O(km \log n \log m)$ linear measurements in $O(\log m)$ adaptive rounds, and its output A' is $O(km \log m)$-sparse.*

The theorem is proved in Section 17. The algorithm extends to the scenario where instead of k we are given ε as a part of the input (k is implicit from x and ε), by the approach of repeatedly doubling the guess for k. More details are given in Section 15.2.

The main challenge in the batch recovery setting is to discover how the km heavy entries of A are distributed among the columns $A_1, \ldots, A_m$. Once this distribution is known, even approximately, the algorithm can perform robust sparse recovery separately on each column. Formally, let A^* be a (km)-sparse matrix such that $\|A - A^*\|_1 \leq \varepsilon \|A\|_1$, and denote the sparsity of its j-th column by $k_j := \|A^*_j\|_0$. If the values $\{k_j\}_{j\in[m]}$ are known, then the columns can be recovered near-optimally using $\sum_{j\in[m]} O(k_j \log(n/k_j)) = O(km \log n)$ linear measurements, all in one additional round.

The key innovation in our algorithm (presented in Section 17) is that in addition to traditional sparse recovery separately on each column, our algorithm adopts a broader view and constructs a matrix A' that estimates A.

Intuitively, every algorithm for batch recovery must employ such a broader view and learn how the heavy entries are distributed across the columns, before it can successfully reconstruct A. Indeed, our second result shows that this is more than mere intuition, and proves that in the *non-adaptive* setting, batch recovery is significantly harder than standard sparse recovery. Specifically, in Section 18 we prove the following theorem, which shows that non-adaptive algorithms for reconstructing A require $\Omega(nm)$ measurements, even in the noise-free case $\varepsilon = 0$.

Theorem 15.1. *For every $m > n$, every non-adaptive randomized scheme for batch recovery must make $\Omega(mn)$ linear measurements in the worst case, even when $k = 2$ and the input is $(2m)$-sparse (i.e., $\varepsilon = 0$).*

15.2 A Noise-Capped Variant of Sparse Recovery

In addition to our main result we give a variant of stable sparse recovery, where the input specifies an intended noise level $\varepsilon \in [0, 1]$ instead of an intended sparsity bound $k \in [n]$. While each of these two parameters completely determines the other one, which of them is given explicitly does matter algorithmically. Specifically, in our *noise-capped* variant, the input is $x \in \mathbb{R}^n$ and $\varepsilon \in [0, 1]$, and the goal is to recover $x' \in \mathbb{R}^n$ that satisfies

$$\|x - x'\|_1 \leq O(\varepsilon)\|x\|_1 \,, \tag{15.2}$$

using a small number of linear measurements to x, where small is with respect to the minimal sparsity k needed to approximate x with relative error ε. We prove the following theorem in Appendix 19 using once more the approach of repeatedly doubling the guess for k (and applying standard sparse recovery).

Theorem 15.2 (Noise-Capped Sparse Recovery)**.** *There is an adaptive randomized scheme that, for every input $x \in \mathbb{R}^n$ and $\varepsilon \in [0,1]$, outputs $x' \in \mathbb{R}^n$, such that with high probability: (i) x' satisfies* (15.2)*; (ii) the algorithm performs $O(k \log n)$ linear measurements in $O(\log k)$ adaptive rounds, where $k = k(x,\varepsilon) := \min\{\|x^*\|_0 : \|x - x^*\|_1 \leq \varepsilon \|x\|_1\}$; and moreover (iii) the output x' is $(2k)$-sparse.*

We additionally show that in the non-adaptive setting, noise-capped recovery is significantly harder than stable sparse recovery, and requires at least n measurements, even in very simple settings. Since even the naive algorithm makes only n measurements, we obtain that a (non-trivial) noise-capped recovery scheme must be adaptive.

Proposition 15.3. *Every non-adaptive randomized scheme for noise-capped sparse recovery (randomized means that, for every input $x \in \mathbb{R}^n$ and $\varepsilon \in [0,1]$, the output x' with high probability satisfies* (15.2)*) must perform, with high probability at least n linear measurements, even in the case $\varepsilon = 0$ and $k(x,0) = 1$.*

The validity of Proposition 15.3 is easily verified. For consider such a scheme in the case $\varepsilon = 0$. Since it is non-adaptive, the number of measurements t depends only on n and the scheme's coins, and it thus must be independent of $k(x,0)$. As the scheme succeeds on *every* input $x \in \mathbb{R}^n$ with high probability, including non-sparse inputs, the scheme must perform with high probability at least n measurements.

15.3 Related Work

The recovery of a matrix from partial or corrupted measurements has numerous applications in theoretical fields such as streaming and sublinear algorithms as well as practical ones, such as signal processing, communication-networks analysis, computer vision and machine learning. In many of these natural settings the matrix is typically sparse. Such settings include covariance matrices [DSBN15], adjacency matrices of sparse or random graphs [McG09, DSBN15], image and video processing for facial recognition [WYG+09] and medical imaging [Mal08], in addition to traffic analysis of large communication networks [CQZ+14]. Woodruff and Zhang [WZ12, WZ13] considered a distributed model known as the *message-passing* model, similar to that described in the previous section. m servers, each holding partial information regarding an unknown matrix A (not necessarily a column, though), need to communicate with a designated coordinator in order to compute some function of A. The communication they consider is not restricted to linear measurements. They show communication complexity lower bounds in terms of bit-complexity for several designated functions (e.g. $\|A\|_0$ or $\|A\|_\infty$).

Considerable work has been made on the reconstruction of a matrix from linear measurements performed on the matrix rather than on each column separately. That is, each

measurement is of the form $A \mapsto B \bullet A = \sum A_{ij}B_{ij}$, where $B \in \mathbb{R}^{n\times m}$ is a sensing "vector". This model offers a much richer set of linear measurements, and, in fact, in terms of sparse recovery reduces matrix reconstruction to stable sparse recovery, albeit in some models, e.g. the message-passing model, such measurements are infeasible. Waters, Sankaranarayanan and Baraniuk [WSB11] give an adaptive sensing algorithm for recovering a matrix which is the sum of a low-rank matrix and a sparse matrix using linear measurements on A.

Dasarathy *et al.* [DSBN15] recently considered matrix recovery from *bilinear* measurements (also called *tensor products*), i.e., measurements of the form $A \mapsto v^tAu = \sum A_{ij}v_iu_j$, where v, u are sensing vectors. This model can be viewed as a restriction of the aforementioned model, in which every sensing vector B is a matrix of rank 1. They show how to reconstruct a sparse matrix using bilinear measurements, in the special case where the heavy entries of A are not concentrated in a few columns.

16 Preliminaries

It is well known [CRT06] that given a sparsity parameter s, there exist sensing matrices and associated recovery algorithms for the stable sparse recovery problem such that the number of linear measurements is $O(s \log n)$.

Claim 16.1 ([CRT06])**.** *Let $s \in [n]$, there exists a sensing matrix $S \in \mathbb{R}^{t\times n}$ for $t = O\left(s \log \frac{n}{s}\right)$, and an associated recovery algorithm such that for every $x \in \mathbb{R}^n$, given Sx, produces an s-sparse vector $x'(s)$ that satisfies $\|x - x'(s)\|_1 \leq 3 \min_{x^*\ s-sparse} \|x - x^*\|_1$. Moreover, such S can be found efficiently with high probability.*

We note that the choice of a "for all" guarantee in Claim 16.1 is not required in our case. The role of Claim 16.1 in our algorithms can be replaced by any (adaptive or non-adaptive) stable sparse recovery guarantee either in the "for all" or "for each" models.

Indyk [Ind06] showed that one can construct a matrix $S \in \mathbb{R}^{t\times n}$, where $t = O(\log n)$ such that for every $x \in \mathbb{R}^n$, given Sx, we can estimate $\|x\|_1$ up to a constant factor with high probability.

Claim 16.2 ([Ind06])**.** *There exists a sensing matrix $S \in \mathbb{R}^{t\times n}$ for $t = O(\log n)$, and an associated algorithm such that for every $x \in \mathbb{R}^n$, given Sx, produces a number $\varrho(x)$ that satisfies $\frac{1}{2}\|x\|_1 \leq \varrho(x) \leq 2\|x\|_1$ with probability at least $1 - \frac{1}{n^{\Omega(1)}}$. Moreover, such S can be found efficiently.*

17 Batch Reconstruction

In this section we prove Theorem 1.8 by presenting an iterative algorithm that, given access to $A_1, \ldots, A_m$ via linear measurements, in addition to a parameter k as in Definition 1.7, constructs vectors $A_1^{alg}, \ldots, A_m^{alg}$ satisfying $\|A^{alg}\|_0 \leq O(km \log m)$ and $\|A - A^{alg}\|_1 \leq O(\varepsilon)\|A\|_1$. We will additionally show that the algorithm performs a total of at most $O(km \log m \log n)$ linear measurements.

The algorithm, described in detail as Algorithm 17.1, performs $\log(2m)$ iterations. Throughout the execution, it maintains a set $I \subseteq [m]$, of the indices of all columns not yet fixed by the algorithm. Initially I is the entire set $[m]$. At every iteration $\ell \in [\log(2m)]$, the algorithm performs standard sparse recovery on each of the columns indexed by I with sparsity parameter $2^{\ell+1}k$, and constructs vectors $\{A_j^{tmp}\}_{j \in I}$. Applying Claim 16.2, the algorithm additionally estimates the residual error $\|A_j - A_j^{tmp}\|_1$ for all $j \in I$, and then chooses the $\frac{1}{2}|I|$ indices for which the residual error is smallest. For each such index j, the algorithm fixes A_j^{alg} to be A_j^{tmp}, and removes j from I. After $\log(2m)$ iterations, the algorithm returns $A_1^{alg}, \ldots, A_m^{alg}$.

```
initialize I ← [m].
for ℓ = 1 to log(2m) do
   for all j ∈ I do
      let A_j^tmp ← A'_j(2^{ℓ+1}k) // by applying Claim 16.1
      ρ_j ← ρ(A_j − A_j^tmp) // by applying Claim 16.2
      let I_ℓ ⊆ I be the set of ⌈m/2^ℓ⌉ indices j ∈ I with smallest ρ_j.
      let I ← I \ I_ℓ
      for all j ∈ I_ℓ do
         let A_j^alg ← A_j^tmp // fix the columns {A_j^tmp : j ∈ I_ℓ}
return A_1^alg, ..., A_n^alg.
```

Algorithm 17.1: Algorithm for Batch Reconstruction

Prior to analyzing Algorithm 17.1 in the next section, let us note that we can easily bound the number of times it invokes Claims 16.2 and 16.1, and show that with probability at least $1 - \frac{1}{n^{\Omega(1)}}$ all invocations succeed. We therefore condition on that event. For sake of simplicity, we assume that m is a power of 2. The next claim follows by simple induction.

Proposition 17.1. *For every $\ell \in [\log(2m)]$, at the beginning of the ℓth iteration, $|I| = \frac{m}{2^{\ell-1}}$.*

It follows that at the end of the last iteration of the main loop, $I = \emptyset$, and thus the output columns are all well-defined.

17.1 Controlling the Noise and the Number of Measurements

The main challenge is to bound the relative error by $O(\varepsilon)$. We remark that if one is willing to pay an extra factor of $\log m$, then there is a simple analysis for this algorithm, as follows. Let A^* denote a (km)-sparse matrix satisfying $\|A - A^*\|_1 \leq \varepsilon\|A\|_1$. By straightforward averaging, for every $\ell \in [\log(2m)]$, at most $\frac{m}{2^{\ell+1}}$ columns A_j^* have $\|A_j^*\|_1 > 2^{\ell+1}k$, and at most $\frac{m}{2^{\ell+1}}$ columns A_j^* have $\|A_j - A_j^*\|_1 > \frac{2^{\ell+1}}{m}\varepsilon\|A\|_1$. The total number of columns in these two groups is at most $2 \cdot \frac{m}{2^{\ell+1}} = \frac{1}{2}|I|$, and thus at least $\lceil \frac{1}{2}|I| \rceil = |I_\ell|$ columns in I are not in these two groups. By the sparse recovery guarantees on these columns and the choice of I_ℓ,

$$\forall j \in I_\ell, \qquad \|A_j - A_j^{alg}\|_1 \leq \frac{3 \cdot 2^{\ell+1}}{m}\varepsilon\|A\|_1. \tag{17.1}$$

Summing these over all values of ℓ we get that $\|A - A^{alg}\|_1 \leq O(\varepsilon \log m)$. To improve this guarantee to $O(\varepsilon)$, we need to tighten the bound in (17.1). We replace the term $\|A\|_1/m$, which represents the norm of an average column, with the norm of a specific column, and the crux is that they sum up (over all iterations ℓ) very nicely, because these summands correspond (essentially) to distinct columns.

We start the analysis by first bounding the number of linear measurements performed by the algorithm, as well as the sparsity of A^{alg}.

Lemma 17.2. *During the ℓth iteration of the main loop, Algorithm 17.1 performs $O(km \log n)$ linear measurements, and moreover $\sum_{j \in I_\ell} \|A_j^{alg}\|_0 \leq O(km)$.*

Proof. At the beginning of the ℓth iteration, $|I| \leq \frac{m}{2^{\ell-1}}$. For every $j \in I$, the algorithm performs $O(2^{\ell+1} k \log n)$ linear measurements on A_j, and by Claim 16.1 we have $\|A_j^{tmp}\|_0 \leq O(2^{\ell+1}k)$. Thus the total number of measurements performed during the ℓth iteration is $O(2^{\ell+1} k \log n) \cdot |I| \leq O(mk \log n)$. Similarly we get that $\sum_{j \in I_\ell} \|A_j^{alg}\|_0 = \sum_{j \in I_\ell} \|A_j^{tmp}\|_0 \leq O(km)$. □

The algorithm performs $\log(2m)$ iterations, and thus at the end of execution, $\|A^{alg}\|_0 \leq O(km \log m)$. Moreover, the total number of linear measurements performed throughout the execution is at most $O(km \log n \log m)$. It remains to show that $\|A - A^{alg}\|_1 \leq O(\varepsilon)\|A\|_1$.

To this end, denote for every $j \in [m]$, $\varepsilon_j = \frac{\|A_j - A_j^*\|_1}{\|A\|_1}$, and note that $\sum_{j \in [m]} \varepsilon_j = \varepsilon$. Let $\varepsilon_{(1)} \geq \varepsilon_{(2)} \geq \ldots \geq \varepsilon_{(m)}$ be a non-increasing ordering of $\{\varepsilon_j\}_{j \in [m]}$.

Lemma 17.3. *For every $\ell \in [\log(m/2)]$ and $j \in I_\ell$, $\frac{\rho_j}{\|A\|_1} \leq 6\varepsilon_{(2^{-\ell-1}m)}$.*

Proof. Fix some $\ell \in [\log(m/2)]$, and consider the set I at the beginning of the ℓth iteration. Recall that $|I| = m/2^{\ell-1}$, and I_ℓ is the set of $m/2^\ell$ indices $j \in I$ with smallest ρ_j. Observe that to prove the claim, it is enough to show that

$$\Pr_{j \in I}\left[\frac{\rho_j}{\|A\|_1} > 6\varepsilon_{(2^{-\ell-1}m)}\right] \leq \frac{1}{2} .$$

To this end, consider an arbitrary $j \in I$ with $\frac{\rho_j}{\|A\|_1} > 6\varepsilon_{(2^{-\ell-1}m)}$. If, in addition, $\|A_j^*\|_0 \leq 2^{\ell+1}k$, then by Claim 16.1

$$\|A_j - A_j^{tmp}\|_1 = \|A_j - A_j'(2^{\ell+1}k)\|_1 \leq 3 \min_{x^* \ 2^{\ell+1}k-sparse} \|A_j - x^*\|_1 \leq 3\|A_j - A_j^*\|_1 = 3\varepsilon_j \|A\|_1 .$$

By Claim 16.2, $\|A_j - A_j^{tmp}\|_1 \geq \frac{1}{2}\rho(A_j - A_j^{tmp}) = \frac{1}{2}\rho_j$, and therefore

$$\varepsilon_j \|A\|_1 \geq \frac{1}{3}\|A_j - A_j^{tmp}\|_1 \geq \frac{1}{6}\rho_j > \varepsilon_{(2^{-\ell-1}m)}\|A\|_1 .$$

We conclude that for every $j \in I$, if $\frac{\rho_j}{\|A\|_1} > 6\varepsilon_{(2^{-\ell-1}m)}$, then either $\|A_j^*\|_0 > 2^{\ell+1}k$ or $\varepsilon_j > \varepsilon_{(2^{-\ell-1}m)}$. By definition, $\varepsilon_j > \varepsilon_{(2^{-\ell-1}m)}$ occurs for at most $\frac{m}{2^{\ell+1}}$ indices $j \in [m]$. In addition, since $\mathbb{E}_{j \in [m]}[\|A_j^*\|_0] = k$, then at most $\frac{m}{2^{\ell+1}}$ indices $j \in [m]$ satisfy $\|A_j^*\|_0 > 2^{\ell+1}k$. Thus at most $\frac{m}{2^\ell} = \frac{1}{2}|I|$ indices $j \in [m]$ satisfy $\frac{\rho_j}{\|A\|_1} > 6\varepsilon_{(2^{-\ell-1}m)}$. The claim follows. □

Lemma 17.4. *For every $\ell \in [\log(m/2)]$,*

$$\sum_{j\in I_\ell} \|A_j - A_j^{alg}\|_1 \leq 24\|A\|_1 \sum_{j\in\left[\frac{m}{2^{\ell+1}},\frac{m}{2^\ell}-1\right]} \varepsilon_{(j)} \ .$$

Proof. Fix some $\ell \in [\log(m/2)]$, and let $j \in I_\ell$. Then A_j^{alg} was fixed in the ℓth iteration, and thus by the previous claim, $\|A_j - A_j^{alg}\|_1 \leq 2\rho(A_j - A_j^{alg}) = 2\rho_j \leq 12\varepsilon_{(2^{-\ell-1}m)}\|A\|_1$. Since $|I_\ell| = \frac{m}{2^\ell}$ and $\{\varepsilon_{(j)}\}_{j\in[m]}$ is non-increasing

$$\sum_{j\in I_\ell} \|A_j - A_j^{alg}\|_1 \leq 12\|A\|_1 \cdot \tfrac{m}{2^\ell} \cdot \varepsilon_{(2^{-\ell-1}m)} \leq 12\|A\|_1 \cdot 2 \sum_{j\in\left[\frac{m}{2^{\ell+1}},\frac{m}{2^\ell}-1\right]} \varepsilon_{(j)} \ .$$

□

Corollary 17.5. $\|A - A^{alg}\|_1 \leq O(\varepsilon)\|A\|_1$.

Proof. Since $\bigcup_{\ell\in[\log(2m)]} I_\ell = [m]$, and the sets $\{I_\ell\}_{\ell\in\log(2m)}$ are pairwise disjoint, then

$$\|A - A^{alg}\|_1 = \sum_{\ell=1}^{\log(2m)} \sum_{j\in I_\ell} \|A_j - A_j^{alg}\|_1 = \sum_{j\in I_{\log(2m)}\cup I_{\log m}} \|A_j - A_j^{alg}\|_1 + \sum_{\ell=1}^{\log m} \sum_{j\in I_\ell} \|A_j - A_j^{alg}\|_1 \ .$$

Observe first that $|I_{\log(2m)} \cup I_{\log m}| = 3$. Moreover, for every $j \in I_{\log(2m)} \cup I_{\log m}$, A_j^{alg} is constructed by applying standard sparse recovery on A_j with sparsity parameter $\geq mk$. Since $\|A_j^*\|_0 \leq km$, then similarly to the previous proof we get that $\|A_j - A_j^{alg}\|_1 \leq 3\varepsilon_j\|A\|_1 \leq 3\varepsilon\|A\|_1$. Therefore by the previous lemma

$$\|A - A^{alg}\|_1 \leq 9\varepsilon\|A\|_1 + 24\|A\|_1 \sum_{\ell=1}^{\log m} \sum_{j\in\left[\frac{m}{2^{\ell+1}},\frac{m}{2^\ell}-1\right]} \varepsilon_{(j)} \leq 9\varepsilon\|A\|_1 + 24\|A\|_1 \sum_{j=1}^{m} \varepsilon_{(j)} = O(\varepsilon)\|A\|_1 \ .$$

□

Theorem 1.8 follows from Lemma 17.2 and Corollary 17.5.

18 Adaptivity is Necessary Even for Noise-Free Signals

To prove Theorem 15.1, we first note that every non-adaptive scheme ALG to the problem of reconstructing a km-sparse matrix $A \in \mathbb{R}^{n\times m}$ can be viewed as the concatenation of two algorithms. ALG^s constructs sensing matrices $S_1, \ldots, S_m$, and ALG^r is given the measurements $S_1A_1, \ldots, S_mA_m$, and recovers A. For every $j \in [m]$, let t_j denote the number of rows of (i.e. measurements performed by) S_j. The total number of linear measurements performed by ALG^s is therefore $t_{\text{ALG}} := \sum_{j\in[m]} t_j$.

Assume that for every $A \in \mathbb{R}^{n\times m}$, ALG reconstructs A with success probability $\geq \frac{1}{2}$. Since ALG is non-adaptive, the number of measurements r_{ALG} depends only on k, m, n.

By Yao's minimax principle, it suffices to show a distribution $\mathcal{D}$ over $(2m)$-sparse matrices in $\mathbb{R}^{n\times m}$ such that for every deterministic algorithm ALG_{det}, if $\Pr_{A\sim\mathcal{D}}[\text{ALG}_{det} \ succeeds] \geq \frac{1}{2}$, then $t_{\text{ALG}_{det}} \geq \Omega(mn)$. Consider a matrix A constructed as follows. Choose uniformly at random $i^* \in [m]$. For every $j \in [m] \setminus \{i^*\}$, let $A_j = (1, 0, 0, \ldots, 0)^t$, and let $A_{i^*} = (x_1, \ldots, x_n)^t$, where $x_1, \ldots, x_n \sim N(0, 1)$ i.i.d. ALG^s_{det} constructs sensing matrices $S_1, \ldots, S_m$. The following lemma implies Theorem 15.1.

Lemma 18.1. *Fix $j \in [m]$, and assume that $t_j \leq n-1$, then conditioned on $i^* = j$, ALG^r_{det} fails to recover A_j with probability 1.*

Proof. Conditioned on $i^* = j$, the distribution of A_j is independent of $\{A_i\}_{i\neq j}$. We can therefore analyze the success probability of ALG^r_{det} on A_j as if ALG^r_{det} receives only (S_j, S_jA_j) as input, and attempts to recover A_j. Denote $U = \mathsf{Ker}(S_j)$ and $\nu = \dim U$, then since S_j is underdetermined, $\nu \geq 1$, and there is an orthonormal basis $u_1, \ldots, u_n$ to $\mathbb{R}^n$ satisfying that $u_1, \ldots, u_\nu$ is a basis of U, and $u_{\nu+1}, \ldots, u_n$ is a basis of $U^\perp$. For every $y \in \mathsf{Im}(S_j)$ there exists a unique $x' \in U^\perp$ satisfying $S_jx = y$. Since ALG^r_{det} is deterministic, there exists a unique $x \in \mathbb{R}^n$ such that ALG^r_{det} returns x when invoked on (S_j, y).

Denote $A_j = \sum_{\ell\in[n]} y_\ell u_\ell$, then since $\{u_\ell\}_{\ell\in[n]}$ is orthonormal, then $y_1, \ldots, y_n \sim N(0, 1)$ i.i.d. Following the above discussion, the value of S_jA_j is independent of $y_1, \ldots, y_\nu$. Moreover, for every $y_{\nu+1}, \ldots, y_n \in \mathbb{R}$, there exist unique values $y^*_1, \ldots, y^*_\nu$ such that $\text{ALG}^r_{det}(S_j, S_jA_j)$ is correct if and only if $y_\ell = y^*_\ell$ for all $\ell \in [\nu]$. Therefore,

$$\Pr[\text{ALG}^r_{det} \ fails | i^* = j] = \int_{y_{\nu+1},\ldots,y_n\in\mathbb{R}} \int_{(y_1,\ldots,y_\nu)\neq(y^*_1,\ldots,y^*_\nu)} \left(\prod_{\ell\in[n+1]} f(y_\ell)\right) dy_1 \ldots dy_n = 1 \ ,$$

where f is the pdf of the standard normal distribution. □

Proof of Theorem 15.1. Denote $J := \{j \in [m] : t_j \geq n\}$. From Lemma 18.1, whenever $j \notin J$, $\Pr[\text{ALG}^r_{det} \ recovers \ A_j | i^* = j] = 0$. Since ALG_{det} reconstructs A with probability $\geq \frac{1}{2}$, then

$$\frac{1}{2} \leq \Pr[\text{ALG}^r_{det} \ recovers \ A] = \Pr[\text{ALG}^r_{det} \ recovers \ A | i^* \in J] \Pr[i^* \in J] \leq \Pr[i^* \in J] \ .$$

Therefore,

$$t_{\text{ALG}_{det}} = \sum_{j\in[m]} t_j \geq \sum_{j\in[m]:t_j\geq n} t_j \geq n \cdot \frac{m}{2} \geq \Omega(mn) \ .$$

□

19 Noise-Capped Sparse Recovery

We now prove Theorem 15.2 by presenting an adaptive algorithm that constructs a sensing matrix $S \in \mathbb{R}^{k^*\times n}$ and, given Sx, can recover a $(2k^*)$-sparse vector x' close to x, where

$k^* = k(x,\varepsilon) = \min\{\|x^*\|_0 : \|x - x^*\|_1 \leq \varepsilon\|x\|_1\}$. The algorithm is iterative, and, given $x \in \mathbb{R}^n$ and a parameter $\varepsilon \in [0,1]$, performs with high probability $O(k^* \log n)$ linear measurements on x, and constructs a $(2k^*)$-sparse vector x^{alg} satisfying $\|x - x^{alg}\|_1 \leq O(\varepsilon)\|x\|_1$.

Loosely speaking, starting with $k = 1$, during each iteration, the algorithm invokes Claim 16.1: namely it performs $O(k \log \frac{n}{k})$ linear measurements on x and thus constructs $x'(k)$. Ideally, we wish to proceed as follows. If $\|x - x'(k)\|_1 \leq O(\varepsilon)\|x\|_1$ the algorithm halts and then returns $x'(k)$. Otherwise, it doubles k and performs another iteration. However, since we only have access to x via linear measurements, we cannot compute $\|x - x'(k)\|_1$ or $\|x\|_1$ exactly (using a small number of measurements). Not all is lost, though. Applying Claim 16.2, the algorithm can perform additional $O(\log n)$ linear measurements on x and estimate $\|x\|_1$ with high probability. Moreover, since $x'(k)$ is known, and by utilizing the linearity of the measurements, we can also estimate $\|x - x'(k)\|_1$ with high probability. The algorithm is given in detail as Algorithm 19.1. We now turn to prove Theorem 15.2.

Input: $x \in \mathbb{R}^n$ accessed by linear measurements and an upper bound $\varepsilon \geq \min_{x^* \ k^*-sparse} \frac{\|x-x^*\|_1}{\|x\|_1}$.
Output: A $2k^*$-sparse $x' \in \mathbb{R}^n$ satisfying $\|x - x'\|_1 \leq O(\varepsilon)\|x\|_1$.
1: $\ell \leftarrow 0$
2: $x^{alg} \leftarrow 0$.
3: **while** $\varrho(x - x^{alg}) \geq 12\varepsilon\varrho(x)$ **do**
4: $\quad \ell \leftarrow \ell + 1$
5: $\quad x^{alg} \leftarrow x'(2^\ell)$ as in Claim 16.1.
6: **return** x^{alg}

Algorithm 19.1: Sparse Recovery Given $\varepsilon \geq 0$

Fix $x \in \mathbb{R}^n$ and $\varepsilon \in [0,1]$. By the definition of k^* there exists a k^*-sparse $x^* \in \mathbb{R}^n$ satisfying $\|x - x^*\|_1 \leq \varepsilon\|x\|_1$. Denote by Λ the number of iterations performed by the algorithm, when invoked on x, ε. Fix some $\ell \in \Lambda$. Let $\mathcal{E}_\ell$ denote the event that the algorithm fails to estimate $\|x - x'(2^\ell)\|_1$. That is, $\varrho(x - x'(2^\ell)) \notin \left[\frac{1}{2}\|x - x'(2^\ell)\|_1 \ , \ 2\|x - x'(2^\ell)\|_1\right]$. Similarly, let $\mathcal{E}_0$ denote the event that the algorithm failed to estimate $\|x\|_1$. By Claim 16.2, $\Pr[\mathcal{E}_\ell] \leq \frac{1}{n^{\Omega(1)}}$ for every $0 \leq \ell \leq \Lambda$. Since by performing n linear measurements we can uniquely characterize x, that is $x'(n) = x$, then $\Lambda \leq \log n$ with certainty. Applying a union bound we get that $\Pr\left[\bigcup_{0 \leq \ell \leq \Lambda} \mathcal{E}_\ell\right] \leq \frac{1}{n^{\Omega(1)}}$. We can therefore condition on $\bigcap_{0 \leq \ell \leq \Lambda} \bar{\mathcal{E}}_\ell$.

Lemma 19.1. *Algorithm 19.1 performs at most $O(k^* \log n)$ linear measurements in $O(\log k^*)$ adaptive rounds. Furthermore, in the end of the execution $\|x^{alg}\|_0 \leq 4k^*$.*

Proof. Consider the ℓth iteration for $\ell = \lceil \log k^* \rceil$. Then $k^* \leq 2^\ell \leq 2k^*$. Since x^* is k^*-sparse, we get that

$$\|x - x'(2^\ell)\|_1 \leq 3 \min_{y \ 2^\ell-sparse} \|x - y\|_1 \leq 3\|x - x^*\|_1 \leq 3\varepsilon\|x\|_1 \ .$$

Therefore, $\varrho(x - x'(2^\ell)) \leq 6\varepsilon\|x\|_1 \leq 12\varepsilon\varrho(x)$, and the algorithm halts. It follows that the algorithm performs at most $\lceil \log k^* \rceil$ iterations. The total number of linear measurements

performed is therefore at most $O\left(\sum_{\ell=1}^{\lceil \log k^* \rceil} 2^\ell \log n\right) = O(k^* \log n)$. Moreover, $\|x^{alg}\|_0 \leq 2^{\lceil \log k^* \rceil} = 2k^*$. □

To finish the proof of Theorem 15.2, it remains to show that $\|x - x^{alg}\|_1 \leq O(\varepsilon)\|x\|_1$. It is enough to see that by the halting condition of the algorithm, in the end of the execution

$$\|x - x^{alg}\|_1 \leq 2\varrho(x - x^{alg}) \leq 24\varepsilon\varrho(x) \leq O(\varepsilon)\|x\|_1 \ .$$

This completes the proof of Theorem 15.2.

20 Future Directions

A natural important question that arises from the main results, and is not completely resolved within the context of our previous work is whether we can obtain tight bounds on the number of adaptive rounds needed for a batch sparse recovery scheme that performs $\tilde{O}(km)$ linear measurements. A more refined question asks for the correct tradeoff between the number of adaptive rounds and number of linear measurements. We note that both questions are not resolved even for the simpler special case of a single vector (i.e. the noise-capped recovery).

We showed in Theorem 15.1 that every random non-adaptive scheme for batch sparse recovery must perform $\Omega(mn)$ measurements for every A and k with high probability, even for the case $\varepsilon = 0$. As it turns out, for the case $\varepsilon = 0$, two adaptive rounds are indeed enough to get an optimal number of linear measurements. Indeed, in the first round the algorithm estimates, up to a constant factor, the "correct" sparsity bound of each column, i.e., $\|A_j\|_0$, which can be done by performing $O(\log n)$ linear measurements on every column [KNW10]. In the second round of measurements, the algorithm employs stable sparse recovery (e.g. Claim 16.1) to reconstruct the unknown entries while using the correct sparsity bound, up to a constant factor. The total number of measurements is therefore $O(mk \log n)$, which is optimal by known sparse recovery lower bounds.

For arbitrary values of $\varepsilon \in (0, 1)$, however, the scheme we have presented uses $O(\log m)$ adaptive rounds of measurements in order to bound the number of measurements by $\tilde{O}(km)$. We suspect that a doubling approach similar to that in Algorithm 17.1 is, in a sense, required and therefore every batch recovery scheme that performs $\tilde{O}(km)$ measurements must perform $\Omega(\log m)$ adaptive rounds.

References

[AG06] I. Abraham and C. Gavoille. Object location using path separators. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing*, PODC '06, pages 188–197, 2006.

[AGG+14] I. Abraham, C. Gavoille, A. Gupta, O. Neiman, and K. Talwar. Cops, robbers, and threatening skeletons: Padded decomposition for minor-free graphs. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 79–88, New York, NY, USA, 2014. ACM.

[AGMW10] I. Abraham, C. Gavoille, D. Malkhi, and U. Wieder. Strong-diameter decompositions of minor free graphs. *Theor. Comp. Sys.*, 47(4):837–855, November 2010.

[AKK17] A. Andoni, L. Kamma, and R. Krauthgamer. Batch sparse recovery, or how to leverage the average sparsity. 2017. Submitted for publication.

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows - theory, algorithms and applications*. Prentice Hall, 1993.

[AP90] B. Awerbuch and D. Peleg. Sparse partitions. In *31st Annual IEEE Symposium on Foundations of Computer Science*, pages 503–513, 1990.

[Ass83] P. Assouad. Plongements lipschitziens dans $\mathbf{R}^n$. *Bull. Soc. Math. France*, 111(4):429–448, 1983.

[Bar96] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *37th Annual Symposium on Foundations of Computer Science*, pages 184–193. IEEE, 1996.

[Bar04] Y. Bartal. Graph decomposition lemmas and their role in metric embedding methods. In *12th Annual European Symposium on Algorithms*, volume 3221 of *LNCS*, pages 89–97. Springer, 2004.

[Ben95] A. A. Benczúr. Counterexamples for directed and node capacitated cut-trees. *SIAM J. Comput.*, 24(3):505–510, 1995.

[BG08] A. Basu and A. Gupta. Steiner point removal in graph metrics. Unpublished Manuscript, available from `http://www.math.ucdavis.edu/~abasu/papers/SPR.pdf`, 2008.

[BHKP07] A. Bhalgat, R. Hariharan, T. Kavitha, and D. Panigrahi. An $\tilde{O}(mn)$ Gomory-Hu tree construction algorithm for unweighted graphs. In *39th Annual ACM Symposium on Theory of Computing*, STOC'07, pages 605–614. ACM, 2007.

[BIPW10] K. D. Ba, P. Indyk, E. Price, and D. P. Woodruff. Lower bounds for sparse recovery. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 1190–1197, 2010.

[BK96] A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In *28th Annual ACM Symposium on Theory of Computing*, pages 47–55. ACM, 1996.

[BSS09] J. D. Batson, D. A. Spielman, and N. Srivastava. Twice-Ramanujan sparsifiers. In *41st Annual ACM symposium on Theory of computing*, pages 255–262. ACM, 2009.

[CCPS98] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial optimization*. John Wiley & Sons Inc., New York, 1998.

[CGMZ05] H. T.-H. Chan, A. Gupta, B. M. Maggs, and S. Zhou. On hierarchical routing in doubling metrics. In *16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 762–771. SIAM, 2005.

[CH91] C. Cheng and T. Hu. Ancestor tree for arbitrary multi-terminal cut functions. *Annals of Operations Research*, 33(3):199–213, 1991.

[CKK16] R. Chitnis, L. Kamma, and R. Krauthgamer. Tight bounds for Gomory-Hu-like cut counting. In *Graph-Theoretic Concepts in Computer Science - 42nd International Workshop, WG 2016*, pages 133–144, 2016.

[CKR01] G. Calinescu, H. Karloff, and Y. Rabani. Approximation algorithms for the 0-extension problem. In *12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 8–16. SIAM, 2001.

[CKR04] G. Calinescu, H. J. Karloff, and Y. Rabani. Approximation algorithms for the 0-extension problem. *SIAM J. Comput.*, 34(2):358–372, 2004.

[CQZ+14] Y. Chen, L. Qiu, Y. Zhang, G. Xue, and Z. Hu. Robust network compressive sensing. In *20th Annual International Conference on Mobile Computing and Networking*, MobiCom'14, pages 545–556, 2014.

[CRT06] E. J. Cands, J. K. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223, 2006.

[CSWZ00] S. Chaudhuri, K. V. Subrahmanyam, F. Wagner, and C. D. Zaroliagis. Computing mimicking networks. *Algorithmica*, 26:31–49, 2000.

[CW08] E. J. Candes and M. B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21–30, 2008.

[CXKR06] T. Chan, D. Xia, G. Konjevod, and A. Richa. A tight lower bound for the Steiner point removal problem on trees. In *9th International Workshop on Approximation, Randomization, and Combinatorial Optimization*, volume 4110 of *Lecture Notes in Computer Science*, pages 70–81. Springer, 2006.

[DG10] E. Diot and C. Gavoille. Path separability of graphs. In *Frontiers in Algorithmics, 4th International Workshop, FAW*, pages 262–273, 2010.

[Don06] D. L. Donoho. Compressed sensing. *IEEE Trans. Inf. Theor.*, 52(4):1289–1306, 2006.

[DSB97] B. Das, R. Sivakumar, and V. Bharghavan. Routing in ad hoc networks using a spine. In *Sixth International Conference on Computer Communications and Networks.*, pages 1–20, 1997.

[DSBN15] G. Dasarathy, P. Shah, B. N. Bhaskar, and R. D. Nowak. Sketching sparse matrices, covariances, and graphs via tensor products. *IEEE Trans. Information Theory*, 61(3):1373–1388, 2015.

[EGK+10] M. Englert, A. Gupta, R. Krauthgamer, H. Räcke, I. Talgam-Cohen, and K. Talwar. Vertex sparsifiers: New results from old techniques. In *13th International Workshop on Approximation, Randomization, and Combinatorial Optimization*, volume 6302 of *Lecture Notes in Computer Science*, pages 152–165. Springer, 2010.

[EGK+14] M. Englert, A. Gupta, R. Krauthgamer, H. Rcke, I. Talgam-Cohen, and K. Talwar. Vertex sparsifiers: New results from old techniques. *SIAM Journal on Computing*, 43(4):1239–1262, 2014.

[EK12] Y. C. Eldar and G. Kutyniok, editors. *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012.

[FF56] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

[FR13] S. Foucart and H. Rauhut. *A Mathematical Introduction to Compressive Sensing*. Birkhäuser Base, 2013.

[FRT04] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.

[FT03] J. Fakcharoenphol and K. Talwar. Improved decompositions of graphs with forbidden minors. In *6th International workshop on Approximation algorithms for combinatorial optimization*, pages 36–46, 2003.

[GH61] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9:551–570, 1961.

[GI10] A. Gilbert and P. Indyk. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, 98(6):937–947, 2010.

[GKL03] A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *44th Annual IEEE Symposium on Foundations of Computer Science*, pages 534–543, October 2003.

[GLPS12] A. C. Gilbert, Y. Li, E. Porat, and M. J. Strauss. Approximate sparse recovery: Optimizing time and measurements. *SIAM J. Comput.*, 41(2):436–453, 2012.

[GNR10] A. Gupta, V. Nagarajan, and R. Ravi. Improved approximation algorithms for requirement cut. *Operations Research Letters*, 38(4):322–325, 2010.

[Gup01] A. Gupta. Steiner points in tree metrics don't (really) help. In *12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 220–227. SIAM, 2001.

[Gus90] D. Gusfield. Very simple methods for all pairs network flow analysis. *SIAM J. Comput.*, 19(1):143–155, 1990.

[GVY96] N. Garg, V. V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM Journal on Computing*, 25(2):235–251, 1996.

[HKNR98] T. Hagerup, J. Katajainen, N. Nishimura, and P. Ragde. Characterizing multiterminal flow networks and computing flows in networks of small treewidth. *J. Comput. Syst. Sci.*, 57(3):366–375, 1998.

[Ind06] P. Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006.

[IPW11] P. Indyk, E. Price, and D. P. Woodruff. On the power of adaptivity in sparse recovery. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 285–294, 2011.

[KK16] L. Kamma and R. Krauthgamer. Metric decompositions of path-separable graphs. *to appear in Algorithmica*, http://dx.doi.org/10.1007/s00453-016-0213-0, 2016.

[KKKP05] M. Katz, N. A. Katz, A. Korman, and D. Peleg. Labeling schemes for flow and connectivity. *SIAM J. Comput.*, 34(1):23–40, 2005.

[KKN15] L. Kamma, R. Krauthgamer, and H. L. Nguyen. Cutting corners cheaply, or how to remove steiner points. *SIAM J. Comput.*, 44(4):975–995, 2015.

[KL06] R. Krauthgamer and J. R. Lee. Algorithms on negatively curved spaces. In *47th Annual IEEE Symposium on Foundations of Computer Science*, pages 119–132. IEEE, 2006.

[KLMN05] R. Krauthgamer, J. R. Lee, M. Mendel, and A. Naor. Measured descent: A new embedding method for finite metrics. *Geometric And Functional Analysis*, 15(4):839–858, 2005.

[KNW10] D. M. Kane, J. Nelson, and D. P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM Symposium on Principles of Database Systems, PODS 2010*, pages 41–52, 2010.

[KNZ14] R. Krauthgamer, H. Nguyen, and T. Zondiner. Preserving terminal distances using minors. *SIAM Journal on Discrete Mathematics*, 28(1):127–141, 2014.

[KPR93] P. Klein, S. A. Plotkin, and S. Rao. Excluded minors, network decomposition, and multicommodity flow. In *25th Annual ACM Symposium on Theory of Computing*, pages 682–690, May 1993.

[KR11] R. Krauthgamer and T. Roughgarden. Metric clustering via consistent labeling. *Theory of Computing*, 7(5):49–74, 2011.

[KR13] R. Krauthgamer and I. Rika. Mimicking networks and succinct representations of terminal cuts. In *SODA*, pages 1789–1799, 2013.

[KR14] A. Khan and P. Raghavendra. On mimicking networks representing minimum terminal cuts. *Inf. Process. Lett.*, 114(7):365–371, 2014.

[LN04] J. R. Lee and A. Naor. Metric decomposition, smooth measures, and clustering, 2004. Unpublished manuscript.

[LN05] J. R. Lee and A. Naor. Extending lipschitz functions via random metric partitions. *Inventiones Mathematicae*, 1:59–95, 2005.

[LNSW12] J. Lacki, Y. Nussbaum, P. Sankowski, and C. Wulff-Nilsen. Single source - all sinks max flows in planar digraphs. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 599–608, 2012.

[LR99] T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

[LS93] N. Linial and M. Saks. Low diameter graph decompositions. *Combinatorica*, 13(4):441–454, 1993.

[LS10] J. R. Lee and A. Sidiropoulos. Genus and the geometry of the cut graph. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 193–201. Society for Industrial and Applied Mathematics, 2010.

[Mal08] S. Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way.* Academic Press, 3rd edition, 2008.

[McG09] A. McGregor. Graph mining on streams. In L. Liu and M. T. Özsu, editors, *Encyclopedia of Database Systems*, pages 1271–1275. Springer, 2009.

[MN07] M. Mendel and A. Naor. Ramsey partitions and proximity data structures. *J. Eur. Math. Soc.*, 9(2):253–275, 2007.

[PS89] D. Peleg and A. A. Schäffer. Graph spanners. *J. Graph Theory*, 13(1):99–116, 1989.

[PW11] E. Price and D. P. Woodruff. (1 + eps)-approximate sparse recovery. In *52nd Annual Symposium on Foundations of Computer Science*, FOCS'11, pages 295–304, 2011.

[Rao99] S. Rao. Small distortion and volume preserving embeddings for planar and Euclidean metrics. In *Proceedings of the 15th Annual Symposium on Computational Geometry*, pages 300–306. ACM, 1999.

[Tal04] K. Talwar. Bypassing the embedding: Algorithms for low dimensional metrics. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 281–290, 2004.

[Tho04] M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, November 2004.

[TZ05] M. Thorup and U. Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005.

[WSB11] A. E. Waters, A. C. Sankaranarayanan, and R. G. Baraniuk. SpaRCS: Recovering low-rank and sparse matrices from compressive measurements. In *25th Annual Conference on Neural Information Processing Systems*, NIPS'11, pages 1089–1097, 2011.

[WYG+09] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):210–227, 2009.

[WZ12] D. P. Woodruff and Q. Zhang. Tight bounds for distributed functional monitoring. In *44th Annual ACM Symposium on Theory of Computing*, STOC'12, pages 941–960. ACM, 2012.

[WZ13] D. P. Woodruff and Q. Zhang. When distributed computation is communication expensive. In *Distributed Computing: 27th International Symposium*, DISC'13, pages 16–30. Springer, 2013.

[Yu14] S. Yu. *Distributed Denial of Service Attack and Defense*. Springer Briefs in Computer Science. Springer, 2014.