On Sketching Quadratic Forms[*]

Alexandr Andoni[†] Jiecao Chen[‡] Robert Krauthgamer[§] Bo Qin[¶]
David P. Woodruff[||] Qin Zhang[**]

December 18, 2015

Abstract

We undertake a systematic study of sketching a quadratic form: given an $n \times n$ matrix A, create a succinct sketch $\mathrm{sk}(A)$ which can produce (without further access to A) a multiplicative $(1+\varepsilon)$-approximation to $x^T Ax$ for any desired query $x \in \mathbb{R}^n$. While a general matrix does not admit non-trivial sketches, positive semi-definite (PSD) matrices admit sketches of size $\Theta(\varepsilon^{-2}n)$, via the Johnson-Lindenstrauss lemma, achieving the "for each" guarantee, namely, for each query x, with a constant probability the sketch succeeds. (For the stronger "for all" guarantee, where the sketch succeeds for all x's simultaneously, again there are no non-trivial sketches.)

We design significantly better sketches for the important subclass of graph Laplacian matrices, which we also extend to symmetric diagonally dominant matrices. A sequence of work culminating in that of Batson, Spielman, and Srivastava (SIAM Review, 2014), shows that by choosing and reweighting $O(\varepsilon^{-2}n)$ edges in a graph, one achieves the "for all" guarantee. Our main results advance this front.

1. For the "for all" guarantee, we prove that Batson et al.'s bound is optimal even when we restrict to "cut queries" $x \in \{0,1\}^n$. Specifically, an arbitrary sketch that can $(1+\varepsilon)$-estimate the weight of *all* cuts $(S, \bar{S})$ in an n-vertex graph must be of size $\Omega(\varepsilon^{-2}n)$ bits. Furthermore, if the sketch is a cut-sparsifier (i.e., itself a weighted graph and the estimate is the weight of the corresponding cut in this graph), then the sketch must have $\Omega(\varepsilon^{-2}n)$ edges. In contrast, previous lower bounds showed the bound only for *spectral-sparsifiers*.

[*]A preliminary version of the paper will soon appear in the Proceedings of the 7th Innovations in Theoretical Computer Science (ITCS 2016).

[†]Columbia University. Work done in part while the author was at Microsoft Research Silicon Valley. Email: andoni@cs.columbia.edu

[‡]Indiana University Bloomington. Work supported in part by NSF CCF-1525024, and IU's Office of the Vice Provost for Research through the Faculty Research Support Program. Email: jiecchen@indiana.edu

[§]Weizmann Institute of Science, Israel. Work supported in part by a US-Israel BSF grant #2010418, an Israel Science Foundation grant #897/13, and by the Citi Foundation. Part of the work was done at Microsoft Research Silicon Valley. Email: robert.krauthgamer@weizmann.ac.il

[¶]Hong Kong University of Science and Technology. Work of this author partially supported by Hong Kong RGC GRF grant 16208415. Email: bqin@cse.ust.hk

[||]IBM Almaden Research. Work supported in part by the XDATA program of the Defense Advanced Research Projects Agency (DARPA), administered through Air Force Research Laboratory contract FA8750-12-C-0323. Email: dpwoodru@us.ibm.com

[**]Indiana University Bloomington. Work supported in part by NSF CCF-1525024, and IU's Office of the Vice Provost for Research through the Faculty Research Support Program. Email: qzhangcs@indiana.edu

2. For the "for each" guarantee, we design a sketch of size $\tilde{O}(\varepsilon^{-1}n)$ bits for "cut queries" $x \in \{0,1\}^n$. We apply this sketch to design an algorithm for the distributed minimum cut problem. We prove a nearly-matching lower bound of $\Omega(\varepsilon^{-1}n)$ bits. For general queries $x \in \mathbb{R}^n$, we construct sketches of size $\tilde{O}(\varepsilon^{-1.6}n)$ bits.

Our results provide the first separation between the sketch size needed for the "for all" and "for each" guarantees for Laplacian matrices.

1 Introduction

Sketching emerges as a fundamental building block used in numerous algorithmic contexts to reduce memory, runtime, or communication requirements. Here we focus on sketching *quadratic forms*, defined as follows: Given a matrix $A \in \mathbb{R}^{n\times n}$, compute a sketch of it, $\mathrm{sk}(A)$, which suffices to estimate the quadratic form x^TAx for every query vector $x \in \mathbb{R}^n$. Typically, we aim at $(1+\varepsilon)$-approximation, i.e., the estimate is in the range $(1 \pm \varepsilon)x^TAx$, and sketches that are randomized. The randomization guarantee comes in two flavors. The first one requires that the sketch $\mathrm{sk}(A)$ succeeds (produces a $(1+\varepsilon)$-approximation) on all queries x simultaneously. The second one requires that for every fixed query x, the sketch succeeds with high probability. The former is termed the "for all" guarantee and the latter the "for each" guarantee, following the prevalent terminology in compressive sensing. The main goal is then to design a sketch $\mathrm{sk}(A)$ of small size.

Sketching quadratic forms is a basic task with many applications. In fact, the definition from above abstracts several specific concepts studied before. One important example is the sparsification of a graph G, where we take the matrix A to be the Laplacian of G and restrict the sketch to be of a specific form, namely, a Laplacian of a sparse subgraph G'. Then a cut-sparsifier corresponds to the setting of query vectors $x \in \{0,1\}^n$, in which case x^TAx describes the weight of the corresponding cut in G. Also, a spectral-sparsifier corresponds to query vectors $x \in \mathbb{R}^n$ in which case x^TAx is a Laplacian Rayleigh quotient. Cut queries to a graph have been studied in the context of privacy in databases [GRU12, JT12, BBDS13, Upa13, Upa14] where, for example, vertices represent users and edges represent email correspondence between users, and email correspondences between groups of users are of prime interest. These papers study also directional covariance queries on a matrix, which correspond to evaluating the quadratic form of a positive semidefinite (PSD) matrix, as well as evaluating the quadratic form of a low-rank matrix, which could correspond to, e.g., a user-movie rating matrix. Finally, sketching quadratic forms has appeared and has been studied in other contexts [AHK05, AGM12a, AGM12b, KLM+14, McG14].

Quadratic form computations also arise in numerical linear algebra. Consider the least squares regression problem of minimizing $\|By - c\|_2^2$ for an input matrix B and vector c. Writing the input as an adjoined matrix $M = [B, c]$ and denoting $x = (y, -1)$, the objective is just $\|By - c\|_2^2 = \|Mx\|_2^2 = x^TM^TMx$, and thus regression queries can be modeled by a quadratic form over the PSD matrix $A = M^TM$. Indeed, for a concrete example where a small-space sketch $\mathrm{sk}(A)$ leads to memory savings (in the data-stream model) in regression problems, see [CW09].

To simplify the exposition, let us assume that the matrix A is of size $n \times n$ and its entries are integers bounded by a polynomial in n, and fix the success probability to be 90%. When we consider a graph G, we let n denote its number of vertices, with edge-weights that are positive integers bounded by a polynomial in n. We use $\tilde{O}(f)$ to denote $f \cdot (\log f)^{O(1)}$, which suppresses the distinction between counting bits and machine words.

The general quadratic forms, i.e., when the square matrix A is arbitrary, require a sketch of size $\tilde{\Omega}(n^2)$ bits, even in the "for each" model (see Appendix A).

Hence we restrict our attention to the class of PSD matrices A, and its subclasses like graph Laplacians, which occur in many applications. We provide tight or near-tight bounds for these classes, as detailed in Table 1. Overall, our results show that the specific class of matrices as well as the model ("for each" vs. "for all" guarantee) can have a dramatic effect on the sketch size, namely, quadratic vs. linear dependence on n or on ε.

1.1 Our Contributions

We start by characterizing the sketching complexity for general PSD matrices A, in both the "for all" and "for each" models. First, we show that, for the "for all" model, sketching an arbitrary PSD matrix A requires $\Omega(n^2)$ bits (Theorem 5.1); i.e., storing the entire matrix is essentially optimal. In contrast, for the "for each" model, we show that the Johnson-Lindenstrauss lemma immediately yields a sketch of size $O(n\varepsilon^{-2}\log n)$ bits and this is tight up to the logarithmic factor (see Section 5.2). We conclude that the bounds for the two models are quite different: quadratic vs. linear in n.

Surprisingly, one can obtain significantly smaller sketches when A is the Laplacian of a graph G, a subclass of PSD matrices that occurs in many applications. Specifically, we refer to a celebrated result of Batson, Spielman, and Srivastava [BSS14], which is the culmination of a rich line of research on graph sparsification [BK96, ST04, ST11, SS11, FHHP11, KP12]. They show that every graph Laplacian A admits a sketch in the "for all" model whose size is $O(n\varepsilon^{-2}\log n)$ bits. This stands in contrast to the $\tilde{\Omega}(n^2)$ lower bound for general PSD matrices. Their sketch has a particular structure: it is itself a graph, consisting of a reweighted subset of edges in G and works in the "for all" model. Batson et al. [BSS14] also prove a lower bound for the case of *spectral sparsification* (for cut sparsifiers, the bound remained open).

The natural question is whether there are qualitatively better sketches we can construct by relaxing the guarantees or considering more specific cases. Indeed, we investigate this research direction by pursuing the following concrete questions:

Q1. Can we improve the "for all" upper bound $O(n\varepsilon^{-2})$ by using an arbitrary data structure?

Q2. Can we improve the bound by restricting attention to *cut* queries? Specifically, can the optimal size of *cut-sparsifiers* be smaller than that of spectral-sparsifier?

Q3. Can we improve the "for each" bound beyond the $\tilde{O}(n\varepsilon^{-2})$ bound that follows from general PSD matrices result (and also from the "for all" model via [BSS14])?

We make progress on all of the above questions, often providing (near) tight results.

In all of these questions, the main quantitative focus is the dependence on the accuracy parameter ε. We note that improving the dependence on ε is important for a variety of reasons. From a theoretical angle, a quadratic dependence is common for estimates with two-sided error, and hence sub-quadratic dependence elucidates new interesting phenomena. From a practical angle, we can set ε to be the smallest value for which the sketch still fits in memory (i.e., we can get *better* estimates with the same memory). In general, quadratic dependence might be prohibitive for large-scale matrices: if, say, ε is 1% then $1/\varepsilon^2 = 10000$.

We answer Q1 negatively by showing that every sketch that satisfies the "for all" guarantee requires $\Omega(n\varepsilon^{-2})$ bits of space, even if the sketch is an arbitrary data structures (see Section 3). This

Matrix family	"for all" model upper bound	"for all" model lower bound	"for each" model upper bound	"for each" model lower bound
General	$\tilde{O}(n^2)$	$\Omega(n^2)$	$\tilde{O}(n^2)$	$\Omega(n^2)$ App. A
PSD	$\tilde{O}(n^2)$	$\Omega(n^2)$ Sec. 5.1	$\tilde{O}(n\epsilon^{-2})$ Sec. 5.2	$\Omega(n\varepsilon^{-2})$ Sec. 5.2
Laplacian, SDD	$\tilde{O}(n\varepsilon^{-2})$ [BSS14]	$\Omega(n\varepsilon^{-2})$ [BSS14]	$\tilde{O}(n\varepsilon^{-1.6})$ Sec. 4.2	$\Omega(n\varepsilon^{-1})$ Sec. 4.1
edge-count:	$O(n\varepsilon^{-2})$ [BSS14]	$\Omega(n\varepsilon^{-2})$ [BSS14]		
Laplacian+cut queries	$\tilde{O}(n\varepsilon^{-2})$ [BSS14]	$\Omega(n\varepsilon^{-2})$ Sec. 3	$\tilde{O}(n\varepsilon^{-1})$ Sec. 4.1	$\Omega(n\varepsilon^{-1})$ Sec. 4.1
edge-count:	$O(n\varepsilon^{-2})$ [BSS14]	$\Omega(n\varepsilon^{-2})$ Sec. 3		

Table 1: Bounds for sketching quadratic forms, expressed in bits, except when counting edges.

matches the upper bound of [BSS14] (up to a logarithmic factor, which stems from the difference between counting words and bits).

Our answer to Q1 essentially answers Q2 as well: our lower bound actually holds even if we only consider cut queries $x \in \{0,1\}^n$. Indeed, an immediate consequence of the $\Omega(n\varepsilon^{-2})$ bits lower bound is that a cut-sparsifier G' must have $\Omega(n\varepsilon^{-2}/\log n)$ edges. We strengthen this further and obtain a tight lower bound of $\Omega(n\varepsilon^{-2})$ edges (even in the case when the cut-sparsifier G' is a not necessarily a subgraph of G). Such an edge lower bound was not known before. The previous lower bound for a cut-sparsifier G', due to Alon [Alo97], uses two additional requirements — that the sparsifier G' has *regular degrees* and *uniform edge weights* — to reach the same conclusion that G' has $\Omega(n/\varepsilon^2)$ edges. Put differently, Alon's lower bound is *quantitatively* optimal — it concludes the tight lower bound of $\Omega(n/\varepsilon^2)$ edges — but it is unsatisfactory *qualitatively*, as it does not cover a cut-sparsifier G' that has edge weights or has non-regular degrees, which may potentially lead to a smaller sparsifier. Similarly, the results of [Nil91, BSS14] apply to spectral-sparsification, which is a harder problem than cut-sparsification. Our result subsumes all of these bounds, and for cut sparsifiers it is in fact the first lower bound under no assumption. Our lower bound holds even for input graphs G that are unweighted.

On the upside, we answer Q3 positively by showing how to achieve the "for each" guarantee using $n\varepsilon^{-1}$ polylog(n) bits of space (see Section 4.1). This bound can be substantially smaller than in the "for all" model when ε is small: e.g., when $\varepsilon = 1/\sqrt{n}$ we obtain size $n^{3/2}$ polylog(n) instead of the $O(n^2)$ needed in the "for all" model. We also show that $\Omega(n\varepsilon^{-1})$ bits of space is necessary for the "for each" guarantee (Theorem 4.13).

We then give an application for the "for each" sketch to showcase that it is useful algorithmically despite having a guarantee that is is weaker than that of a "for all" cut-sparsifier. In particular, we show how to $(1+\varepsilon)$-approximate the global minimum cut of a graph whose edges are distributed across multiple servers (see Section 1.3).

Finally, we consider a "for each" sketch of a Laplacian matrix under arbitrary query vectors $x \in \mathbb{R}^n$, which we refer to as *spectral queries* on the graph G. Such spectral queries give more flexibility than cut queries. For example, if the graph corresponds to a physical system, e.g., the edges correspond to electrical resistors, then spectral queries can evaluate the total heat dissipation of the system for a given set of potentials on the vertices. Also, a spectral query x that is a permutation of $\{1,2,\dots,n\}$ gives the average squared distortion of a line embedding of G. We design in Section 4.2 a sketch for spectral queries that uses $n\varepsilon^{-1.6}$ polylog(n) bits of space. These upper bounds also apply to the symmetric diagonally-dominant (SDD) matrices.

Our results and previous bounds are summarized in Table 1.

1.2 Highlights of Our Techniques

In this section we give technical overviews for our three main results: (1) the lower bound for cut queries on Laplacian matrices (answering Q1 and Q2); (2) the upper bound for cut queries on Laplacian matrices; and (3) the upper bound for spectral queries on Laplacian matrices (answering Q3). We always use G to denote the corresponding graph of the considered Laplacian matrix.

1.2.1 Lower Bound for Sketching Laplacian Matrices with Cut Queries, "For All" Model

We first prove our $\Omega(n\varepsilon^{-2})$-bit lower bound using communication complexity for arbitrary data structures. We then show how to obtain an $\Omega(n\varepsilon^{-2})$ edge lower bound for cut sparsifiers by encoding a sparsifier in a careful way so that if it had $o(n/\varepsilon^2)$ edges, it would violate an $\Omega(n\varepsilon^{-2})$ bit lower bound in the communication problem.

For the $\Omega(n\varepsilon^{-2})$ bit lower bound, the natural thing to do would be to give Alice a graph G, and Bob a cut S. Alice produces a sketch of G and sends it to Bob, who must approximate the capacity of S. The communication cost of this problem lower bounds the sketch size. However, as we just saw, Alice has an upper bound with only $\tilde{O}(n\varepsilon^{-1})$ bits of communication. We thus need for Bob to solve a much harder problem which uses the fact that Alice's sketch preserves all cuts.

We let G be a disjoint union of $\varepsilon^2 n/2$ graphs G_i, where each G_i is a bipartite graph with $\frac{1}{\varepsilon^2}$ vertices in each part. Each vertex in the left part is independently connected to a random subset of half the vertices in the right part. Bob's problem is now, given a vertex v in the left part of one of the G_i, as well as a subset T of half of the vertices in the right part of that G_i, decide if $|N(v) \cap T| > \frac{1}{4\varepsilon^2} + \frac{c}{\varepsilon}$ ($N(v)$ is the set of neighboring vertices of v), or if $|N(v) \cap T| < \frac{1}{4\varepsilon^2} - \frac{c}{\varepsilon}$, for a small constant $c > 0$. Most vertices v will satisfy one of these conditions, by anti-concentration of the binomial distribution. Note that this problem is not a cut query problem, and so *a priori* it is not clear how Bob can use Alice's sketch to solve it.

To solve the problem, Bob will do an exhaustive enumeration on cut queries, and here is where we use that Alice's sketch preserves all cuts. Namely, for each subset S of half of the vertices in the left part of G_i, Bob queries the cut $S \cup T$. As Bob ranges over all (exponentially many) such cuts, what will happen is that for most vertices u in the left part for which $|N(u) \cap T| > \frac{1}{4\varepsilon^2} + \frac{c}{\varepsilon}$, the capacity of $S \cup T$ is a "little bit" larger if u is excluded from S. This little bit is not enough to be detected, since $|N(u) \cap T| = \Theta\left(\frac{1}{\varepsilon^2}\right)$ while the capacity of $S \cup T$ is $\Theta\left(\frac{1}{\varepsilon^4}\right)$. However, as Bob range over all such S, he will eventually get lucky in that S contains all vertices u for which $|N(u) \cap T| > \frac{1}{4\varepsilon^2} + \frac{c}{\varepsilon}$, and now since there are about $\frac{1}{2\varepsilon^2}$ such vertices, the little $\frac{c}{\varepsilon}$ bit gets "amplified" by a factor of $\frac{1}{2\varepsilon^2}$, which is just enough to be detected by a $(1+\varepsilon)$-approximation to the capacity of $S \cup T$. If Bob finds the S which maximizes the (approximate) cut value $S \cup T$, he can check if his v is in S, and this gives him a correct answer with large constant probability.

We believe our main contribution is in designing a communication problem which requires Alice's sketch to preserve all cuts instead of only a single cut. There are also several details in the communication lower bound for the problem itself, including a direct-sum theorem for a constrained version of the Gap-Hamming-Distance problem, which could be independently useful.

For the $\Omega(n\varepsilon^{-2})$ edge lower bound for cut sparsifiers, the straightforward encoding would encode each edge using $O(\log n)$ bits, and cause us to lose a $\log n$ factor in the lower bound. Instead, we show how to randomly round each edge weight in the sparsifier to an adjacent *integer*, and observe that the integer weights sum up to a small value in our communication problem. This ultimately allows to transmit, in a communication-efficient manner, all the edge weights together with the edge identities.

1.2.2 Upper Bound for Sketching Laplacian Matrices with Cut Queries, "For Each" Model

To discuss the main ideas behind our $\tilde{O}(n\varepsilon^{-1})$-bit sketch construction for Laplacian matrices with queries $x \in \{0,1\}^n$, let us first give some intuition on why the previous algorithms cannot yield a $\tilde{O}(n\varepsilon^{-1})$ bound, and show how our algorithm circumvents these roadblocks on a couple of illustrative examples. For concreteness, it is convenient to think of $\varepsilon = 1/\sqrt{n}$.

All existing cut (and spectral) sparsifiers algorithms construct the sparsifier by taking a subgraph of the original graph G, with the "right" re-weightening of the edges [BK96, SS11, BSS14, FHHP11, KP12]. In fact, except for [BSS14], they all proceed by sampling edges independently, each with its own probability (that depends on the graph).

Consider for illustration the complete graph. In this case, these sampling schemes employ a uniform probability $p \approx \frac{1/\varepsilon^2}{n}$ of sampling every edge. It is not hard to see that one cannot sample edges with probability less than p, as otherwise anti-concentration results suggest that even the degree of a vertex (i.e., the cut of a "singleton") is not preserved within $1+\varepsilon$ approximation. Perhaps a more interesting example is a random graph $\mathcal{G}_{n,1/2}$; if edges are sampled independently with (roughly) uniform probability, then again it cannot be less than p, because of singleton cuts. However, if we aim for a sketch for the complete graph or $\mathcal{G}_{n,1/2}$, we can just store the degree of each vertex using only $O(n)$ space, and this will allow us to report the value of every singleton cut (which is the most interesting case, as the standard deviation for these cut values have multiplicative order roughly $1 \pm \varepsilon$). These observations suggest that *sketching* a graph may go beyond considering a subgraph (or a different graph) to represent the original graph G.

Our general algorithm proceeds in several steps. The core of our algorithm is a procedure for handling cuts of value $\approx 1/\varepsilon^2$ in a graph with unweighted edges, which proceeds as follows. First, repeatedly partition the graph along every *sparse* cut, namely, any cut whose sparsity is below $1/\varepsilon$. This results with a partition of the vertices into some number of parts. We store the cross-edges (edge connecting different parts) explicitly. We show the number of such edges is only $\tilde{O}(n\varepsilon^{-1})$, and hence they fit into the space allocated for the sketch. Obviously, the contribution of these edges to any desired cut $w(S,\bar{S})$ is easy to compute from this sketch.

The sketching algorithm still needs to estimate the contribution (to a cut $w(S,\bar{S})$ for a yet unknown $S \subset V$) from edges that are inside any single part P of the partition. To accomplish this, we sample $\approx 1/\varepsilon$ edges out of each vertex, and also store the exact degrees of all vertices. Then, to estimate the contribution of edges inside a part P to $w(S,\bar{S})$, we take the sum of (exact) degrees of all vertices in $S \cap P$, *minus* an estimate for (twice) the number of edges inside $S \cap P$ (estimated from the edge sample). This "difference-based" estimate has a smaller variance than a direct estimate for the number edges in $(S \cap P, \bar{S} \cap P)$ (which would be the "standard estimate", in some sense employed by previous work). The smaller variance is achieved thanks to the facts that (1) the assumed cut is of size (at most) $1/\varepsilon^2$; and (2) there are no sparse cuts in P.

Overall, we achieve a sketch size of $\tilde{O}(n\varepsilon^{-1})$. We can construct the sketch in polynomial time by employing an $O(\sqrt{\log n})$-approximation algorithm for sparse cut [ARV09, She09] or faster algorithms with $(\log^{O(1)} n)$-approximation [Mad10].

1.2.3 Upper Bound for Sketching Laplacian Matrices with Spectral Queries, "For Each" Model

Now we consider spectral queries $x \in \mathbb{R}^n$, starting first with a space bound of $n\varepsilon^{-1.66}\text{polylog}(n)$ bits, and then discuss how to improve it further to $n\varepsilon^{-1.6}\,\text{polylog}(n)$.

We start by making several simplifying assumptions. The first is that the total number of edges is $O(n\varepsilon^{-2})$. Indeed, we can first compute a spectral sparsifier [BSS14]. It is useful to note that if all edges weights were between 1 and $\text{poly}(n)$, then after spectral sparsification the edge weights are between 1 and $\text{poly}(n)$, for a possibly larger polynomial. Next, we can assume all edge weights are within a factor of 2. Indeed, by linearity of the Laplacian, if all edge weights are in $[1, \text{poly}(n)]$, then we can group the weights into powers of 2 and sketch each subset of edges separately, incurring an $O(\log n)$ factor blowup in space. Third, and most importantly, we assume that Cheeger's constant h_G of each resulting graph $G = (V, E)$ satisfies $h_G > \varepsilon^{1/3}$, where recall that $h_G = \inf_{S \subset V} \Phi_G(S)$ where

$$\Phi_G(S) = \frac{w(S, \bar{S})}{\min\{\text{vol}(S), \text{vol}(\bar{S})\}} \quad \text{and} \quad \text{vol}(S) = \sum_{u \in S} w(\{u\}, V \setminus \{u\}).$$

We can assume $h_G > \varepsilon^{1/3}$ because if it were not, then by definition of h_G there is a sparse cut, that is, $\Phi_G(S) \leq \varepsilon^{1/3}$. We can find a sparse cut (a polylogarithmic approximation suffices), store all sparse cut edges in our data structure, and remove them from the graph G. We can then recurse on the two sides of the cut. By a charging argument we can bound the total number of edges stored across all sparse cuts.

As for the actual data structure achieving our $n\varepsilon^{-1.66}\text{polylog}(n)$ upper bound, we first store the weighted degree $\delta_u(G) = \sum_{v:(u,v)\in E} w(u, v)$ of each node (as that for the cut queries). A difference is that we now partition vertices into "heavy" and "light" classes V_L and V_H, where V_H contains those vertices whose weighted degree exceeds a threshold, and light consists of the remaining vertices. We include all edges incident to light vertices in the data structure. The remaining edges have both endpoints heavy and for each heavy vertex, we randomly sample about $\varepsilon^{-5/3}$ of its neighboring heavy edges; edge u, v is sampled with probability $\frac{w(u,v)}{\delta_u(G_H)}$ where $\delta_u(G_H)$ is the sum of weighted edges from the heavy vertex u to neighboring heavy vertices v.

For the estimation procedure, we write $x^T L x = \sum_{(u,v)\in E} (x_u - x_v)^2 w(u, v)$ as

$$\begin{aligned} x^T L x = &\sum_{u \in V} \delta_u(G) x_u^2 - \sum_{u \in V_L, v \in V} x_u x_v w(u, v) \\ &- \sum_{u \in V_H, v \in V_L} x_u x_v w(u, v) - \sum_{u \in V_H} \sum_{v \in V_H} x_u x_v w(u, v) \end{aligned}$$

and observe that our data structure has the first three summations on the right exactly; error can only come from estimating $\sum_{u\in V_H} \sum_{v \in V_H} x_u x_v w(u, v)$, for which we use our sampled heavy edges. Since this summation has only heavy edges, we can control its variance and upper bound it by $\varepsilon^{10/3}\|D^{1/2}x\|_2^4$, where D is a diagonal matrix with the degrees of G on the diagonal. We can then upper bound this norm by relating it to the first non-zero eigenvalue $\lambda_1(\tilde{L})$ of the normalized Laplacian $\tilde{L}$, which cannot be too small, since by Cheeger's inequality, $\lambda_1(\tilde{L}) \geq h_G^2/2$, and we have ensured that h_G is large.

To improve the upper bound to $n\varepsilon^{-1.6}\text{polylog}(n)$ bits, we partition the edges of G into more refined groups, based on the degrees of their endpoints. More precisely, we classify edges e by the minimum degree of their two endpoints, call this number $m(e)$, and two edges e, e' are in the same class if the nearest power of 2 of $m(e)$ and of $m(e')$ is the same. We note that the total number of vertices with degree in $\omega(\varepsilon^{-2})$ is $o(n)$, since we are starting with a graph with only $O(n\varepsilon^{-2})$ edges; therefore, all edges e with $m(e) = \omega(\varepsilon^{-2})$ can be handled by applying our entire procedure recursively on say, at most $n/2$ nodes. Thus, it suffices to consider $m(e) \leq \varepsilon^{-2}$.

The intuition now is that as $m(e)$ increases, the variance of our estimator decreases since the two endpoints have even larger degree now and so they are even "heavier" than before. Hence, we need fewer edge samples when processing a subgraph restricted to edges with large $m(e)$. On the other hand, a graph on edges e for which every value of $m(e)$ is small simply cannot have too many edges; indeed, every edge is incident to a low degree vertex. Therefore, when we partition the graph to ensure that Cheeger's constant h_G is small, since there are fewer total edges (before we just assumed this number was upper bounded by $n\varepsilon^{-2}$), now we pay less to store all edges across sparse cuts. Thus, we can balance these two extremes, and doing so we arrive at our overall $n\varepsilon^{-1.6}\text{polylog}(n)$ bit space bound.

Several technical challenges arise when performing this more refined partitioning. One is that when doing the sparse cut partitioning to ensure the Cheeger's constant is small, we destroy the minimum degree of endpoints of edges in the graph. Fortunately we can show that for our setting of parameters, the total number of edges removed along sparse cuts is small, and so only a small number of vertices have their degree drop by more than a factor of 2. For these vertices, we can afford to store all edges incident to them directly, so they do not contribute to the variance. Another issue that arises is that to have small variance, we would like to "assign" each edge $\{u, v\}$ to one of the two endpoints u or v. If we were to assign it to both, we would have higher variance. This involves creating a companion or "buddy graph" which is a directed graph associated with the original graph. This directed graph assists us with the edge partitioning, and tells us which edges to potentially sample from which vertices.

1.3 Application to Distributed Minimum Cut

We now illustrate how a "for each" sketch can be useful algorithmically despite its relaxed guarantees compared to a cut sparsifier. In particular, we show how to $(1+\varepsilon)$-approximate the global minimum cut of a graph whose edges are distributed across multiple servers. Distributed large-scale graph computation has received recent attention, where protocols for distributed minimum spanning tree, breadth-first search, shortest paths, and testing connectivity have been studied, among other problems, see, e.g., [KNPR15, WZ13]. In our case, each server locally computes the "for each" data structure of Sec. 4.1 on its subgraph (for accuracy ε), and sends it to a central server. Each server also computes a classical cut sparsifier, with fixed accuracy $\varepsilon' = 0.2$, and sends it to the central server. Using the fact that cut-sparsifiers can be merged, the central server obtains a $(1 \pm \varepsilon')$-approximation to all cuts in the union of the graphs. By a result of Henzinger and Williamson [HW96] (see also Karger [Kar00]), there are only $O(n^2)$ cuts strictly within factor 1.5 of the minimum cut, and they can be found efficiently from the sparsifier (see [Kar00] for an $\tilde{O}(n^2)$ time way of implicitly representing all such cuts). The central server then evaluates each "for each" data structure on each of these cuts, and sums up the estimates to evaluate each such cut up to factor $1+\varepsilon$, and eventually reports the minimum found. Note that the "for each" data structures can be assumed, by independent repetitions, to be correct with probability $1 - 1/n^4$ for any fixed cut (and at any server), and therefore correct with high probability on all $O(n^2)$ candidate cuts.

2 Preliminaries

Let $G = (V, E, w)$ be an undirected graph with weight function $w : V \times V \to \mathbb{R}_{\geq 0}$. Denote $n = |V|$ and $m = |E|$, and assume by convention that $w(u, v) > 0$ whenever $(u, v) \in E$, and $w(u, v) = 0$ otherwise. Let $w_{\max}$ and $w_{\min}$ be the maximum and minimum (positive) weights of edges in E respectively.

For a vertex $u \in V$, let $\delta_u(G)$ be the weighted degree of u in G, i.e. $\delta_u(G) = \sum_{v \in V} w(u, v)$, and let $d_u(G)$ be the unweighted degree of u in G, i.e. $d_u(G) = |\{v \in V \mid w(u, v) > 0\}|$.

For two disjoint vertex sets A and B, let $\partial(A, B) = \{(u, v) \in E \mid u \in A, v \in B\}$ be the set of edges across two vertex sets A and B. Abusing the notation a little bit, let $\partial(A, A)$ denotes the set of edges whose endpoints are both inside A. Given a cut $(S, \bar{S})$ with $S \subset V$, we denote the cut weight in G as $w_G(S, \bar{S}) = \sum_{e \in \partial(S, \bar{S})} w(e)$ (we often omit the subscript G when it is clear from the context).

Let $\vec{G} = (V, \vec{E}, w)$ be a directed positively weighted graph. For a vertex $u \in V$, define weighted in- and out-degree $\delta_u^{\texttt{in}}(\vec{G}) = \sum_{v:(v,u) \in \vec{E}} w(v, u)$, $\delta_u^{\texttt{out}}(\vec{G}) = \sum_{v:(u,v) \in \vec{E}} w(u, v)$, and unweighted in- and out-degree $d_u^{\texttt{in}}(\vec{G}) = |\{v \mid (v, u) \in \vec{E}, w(v, u) > 0\}|$, and $d_u^{\texttt{out}}(\vec{G}) = |\{v \mid (u, v) \in \vec{E}, w(u, v) > 0\}|$.

For a vertex set $S \subset V$, let $G(S)$ be the vertex-induced subgraph of G, and $E(S)$ be the edge set of $G(S)$. And for an edge set $F \subset E$, let $G(F)$ be the edge-induced subgraph of G, and $V(F)$ be the vertex set of $G(F)$; definitions will be the same if edges are directed.

Let $L(G)$ be the unnormalized Laplacian of G, and let $\tilde{L}(G)$ be the normalized Laplacian of G.

Let $\lambda_0(L), \lambda_1(L), \ldots, \lambda_{n-1}(L)$ be the eigenvaules of a Laplacian matrix L such that $\lambda_0(L) \leq \lambda_1(L) \leq \ldots \leq \lambda_{n-2}(L) \leq \lambda_{n-1}(L)$. Recall that the smallest eigenvalue of L is always $\lambda_0(L) = 0$. We can always assume G is connected, since otherwise we can sketch each connected component separately. We thus have $\lambda_1(L) > 0$.

A random variable X is called a $(1 + \varepsilon, \delta)$-approximation of Y if $\mathbf{Pr}[X \in (1 \pm \varepsilon)Y] \geq 1 - \delta$. We usually assume $\varepsilon > 1/n$ (or similar) since otherwise the sketch can just store the whole graph.

We define cut and spectral sketch of a graph as follows.

Definition 2.1 **($(1+\varepsilon, \delta)$-cut-sketch)** *A sketch of $G = (V, E, w)$, denoted sk(G), is called a* $(1+\varepsilon, \delta)$-cut-sketch *of G if there is a reconstruction function that given sk(G) and $S \subset V$, outputs a $(1 + \varepsilon, \delta)$-approximation to the weight of the cut $(S, \bar{S})$, i.e., $w(S, \bar{S})$.*

Definition 2.2 **($(1+\varepsilon, \delta)$-spectral-sketch)** *A sketch of $G = (V, E, w)$, denoted sk(G), is called a* $(1+\varepsilon, \delta)$-spectral-sketch *of G if there is a reconstruction function that given sk(G) and $x \in \mathbb{R}^n$, outputs a $(1 + \varepsilon, \delta)$-approximation to $x^T L(G)x$.*

We will need Cheeger's constant and Cheeger's inequality.

Definition 2.3 **(Cheeger's constant)** *Given a graph $G = (V, E, w_G)$, for any $S \subset V$, let vol$_G(S) = \sum_{u \in S} \delta_u(G)$ be the weighted volume of S in G. Let $\Phi_G(S) = \frac{w_G(S,\bar{S})}{\min\{vol_G(S), vol_G(\bar{S})\}}$ be the* conductance *of the cut $(S, \bar{S})$. We define* Cheeger's constant *to be $h_G = \min_{S \subset V} \Phi_G(S)$.*

Lemma 2.4 **(Cheeger's inequality)** *Let $G = (V, E, w)$ be an undirected positively weighted graph. Let $\tilde{L}$ be the normalized Laplacian of G. Let h_G be the Cheeger's constant of graph G. The following* *inequality holds,*

$$\lambda_1(\tilde{L}) \geq \frac{h_G^2}{2}. \tag{1}$$

We introduce another parameter of a graph.

Definition 2.5 (Expansion constant) *For any $S \subset V$, we define* expansion constant *of graph $G = (V, E)$ to be $\Gamma_G = \min_{|S| \leq n/2} \frac{|\partial(S,\bar{S})|}{|S|}$.*

3 Symmetric Diagonally Dominant Matrices, "For All" Model

In this section we prove a lower bound $\Omega(n/\varepsilon^2)$ on the size of "for all" sketches for symmetric diagonally dominant (SDD) matrices. Our lower bound in fact applies even in the the special case of a graph Laplacian A with query vectors $x \in \{0,1\}^n$. Concretely, we prove the following two theorems in Sections 3.1 and 3.3, respectively. The first holds for arbitrary sketches, and the second holds for sketches that take the form of a *graph*.

Theorem 3.1 *Fix an integer n and $\varepsilon \in (1/\sqrt{n}, 1)$, and let $\mathbf{sk} = \mathbf{sk}_{n,\varepsilon}$ and $\mathbf{est} = \mathbf{est}_{n,\varepsilon}$ be possibly randomized sketching and estimation algorithms for unweighted graphs on vertex set $[n]$. Suppose that for every such graph $G = ([n], E)$, with probability at least $3/4$ we have*[1]

$$\forall S \subset [n], \quad \mathbf{est}\left(S, \mathbf{sk}(G)\right) \in \ (1 \pm \varepsilon) \cdot |\partial(S, \bar{S})|.$$

Then the worst-case size of $\mathbf{sk}(G)$ is $\Omega(n/\varepsilon^2)$ bits.

Theorem 3.2 *For every integer n and $\varepsilon \in (1/\sqrt{n}, 1)$, there is an n-vertex graph G for which every $(1+\varepsilon)$-cut sparsifier H has $\Omega(n/\varepsilon^2)$ edges, even if H is not required to be a subgraph of G.*

In addition, we show in Appendix B how to reduce the quadratic form of an SDD matrix can to that of a Laplacian matrix (with only a modest increase in the matrix size, from order n to order $2n$). Thus, the upper bound of sketching SDD matrices in both "for each" and "for all" cases will be the same as that for Laplacians. Since in the "for all" case, we can build the cut (or spectral) sparsifier for a graph using $\tilde{O}(n/\varepsilon^2)$ bits (using e.g. [BSS14]), we can also construct a "for all" sketch for an SDD matrix using $\tilde{O}(n/\varepsilon^2)$ bits of space. This means that our $\Omega(n/\varepsilon^2)$ lower bound is tight up to some logarithmic factors.

3.1 Sketch-size Lower Bound

We prove Theorem 3.1 using the following communication lower bound for a version of the Gap-Hamming-Distance problem, whose proof appears in section 3.2. Throughout, we fix $c = 10^{-3}$ (or a smaller positive constant), and assume $\varepsilon \leq c/10$.

Theorem 3.3 *Consider the following distributional communication problem: Alice has as input $n/2$ strings $s_1, \ldots, s_{n/2} \in \{0,1\}^{1/\varepsilon^2}$ of Hamming weight $\frac{1}{2\varepsilon^2}$, and Bob has an index $i \in [n/2]$ together with one string $t \in \{0,1\}^{1/\varepsilon^2}$ of Hamming weight $\frac{1}{2\varepsilon^2}$, drawn as follows:*[2]

[1]The probability is over the randomness of the two algorithms; equivalently, the two algorithms have access to a common source of random bits.

[2]Alice's input and Bob's input are *not* independent, but the marginal distribution of each one is uniform over its domain, namely, $\{0,1\}^{(n/2)\times(1/\varepsilon^2)}$ and $[n] \times \{0,1\}^{1/\varepsilon^2}$, respectively.

- *i is chosen uniformly at random;*
- *s_i and t are chosen uniformly at random but conditioned on their Hamming distance $\Delta(s_i, t)$ being, with equal probability, either $\geq \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$ or $\leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$;*
- *the remaining strings $s_{i'}$ for $i' \neq i$ are chosen uniformly at random.*

Consider a (possibly randomized) one-way protocol, in which Alice sends to Bob an m-bit message, and then Bob determines, with success probability at least $2/3$, whether $\Delta(s_i, t)$ is $\geq \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$ or $\leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$. Then Alice's message size is $m \geq \Omega(n/\varepsilon^2)$ bits.

We can interpret the lower bound of Theorem 3.3 as follows: Consider a (possibly randomized) algorithm that produces an m-bit sketch of Alice's input $(s_1, \ldots, s_{n/2}) \in \{0,1\}^{n/2\varepsilon^2}$, and suppose that the promise about $\Delta(s_i, t)$ can be decided correctly (with probability at least 3/4) given (only) the sketch and Bob's input $(i, t) \in [n/2] \times \{0,1\}^{1/\varepsilon^2}$. Then $m \geq \Omega(n/\varepsilon^2)$.

We now prove Theorem 3.1 by a reduction to the above communication problem, interpreting the one-way protocol as a sketching algorithm, as follows. Given the instance $(s_1, \ldots, s_{n/2}, i, t)$, define an n-vertex graph G that is a disjoint union of the graphs $\{G_j : j \in [\varepsilon^2 n/2]\}$, where each G_j is a bipartite graph, whose two sides, denoted $L(G_j)$ and $R(G_j)$, are of size

$$|L(G_j)| = |R(G_j)| = 1/\varepsilon^2.$$

The edges of G are determined by $s_1, \ldots, s_{n/2}$, where each string s_u is interpreted as a vector of indicators for the adjacency between vertex $u \in \cup_{j \in [\varepsilon^2 n/2]} L(G_j)$ and the respective $R(G_j)$.

Observe that Alice can compute G without any communication, as this graph is completely determined by her input. She then builds a sketch of this graph, that with probability $\geq 99/100$, succeeds in simultaneously approximating all cut queries within factor $1 \pm \gamma\varepsilon$, where $\gamma > 0$ is a small constant to be determined later. This sketch is obtained from the theorem's assumption about m-bit sketches by standard amplification of the success probability from 3/4 to 0.99 (namely, repeating $r = O(1)$ times independently and answering any query with the median value of the r answers). Alice then sends this $O(m)$-bit sketch to Bob.

Bob then uses his input i to compute $j = j(i) \in [\varepsilon^2 n/2]$ such that the graph G_j contains vertex i (i.e., the vertex whose neighbors are determined by s_i). Bob also interprets his input string t as a vector of indicators determining a subset $T \subseteq R(G_j)$. By construction of G_j , the neighbor sets $N(v)$ of the vertices $v \in L(G_j) \setminus \{i\}$ are uniformly distributed, independently of T and of each other; in particular, each $|N(v) \cap T|$ has a Binomial distribution $B(\frac{1}{\varepsilon^2}, \frac{1}{4})$.

Lemma 3.4 *Using the $O(m)$-bit sketch he received from Alice, Bob can compute a "list" $B \subset L(G_j)$ of size $|B| = \frac{1}{2}|L(G_j)| = \frac{1}{2\varepsilon^2}$, and with probability at least 0.96, this list contains at least $\frac{4}{5}$-fraction of the vertices in the set*

$$L_{\text{high}} = \{v \in L(G_j) : \ |N(v) \cap T| \geq \tfrac{1}{4\varepsilon^2} + \tfrac{c}{\varepsilon}\}. \tag{2}$$

Moreover, Bob uses no information about his input i other than $j = j(i)$.

Before proving the lemma, let us show how to use it to decide about $\Delta(s_i, t)$ and derive the theorem. We will need also the following simple claim, which we prove further below.

Claim 3.5 *With probability at least 0.98, the relative size of L_{high} is $\frac{|L_{\text{high}}|}{|L(G_j)|} \in [\frac{1}{2} - 10c, \frac{1}{2}]$.*

We assume henceforth that the events described in the above lemma and claim indeed occur, which happens with probability at least 0.94. Notice that $\Delta(s_i, t) = \deg(i) + |T| - 2|N(i) \cap T|$.

Now suppose that $\Delta(s_i, t) \le \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$. Then $|N(i) \cap T| \ge \frac{1}{4\varepsilon^2} + \frac{c}{2\varepsilon}$, and because Bob's list B is independent of the vertex $i \in L(G_j)$, we have $\mathbf{Pr}[i \in B] \ge \frac{4}{5}|L_{\text{high}}|/|L_{\text{high}}| = \frac{4}{5}$.

Next, suppose that $\Delta(s_i, t) \ge \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$. Then $|N(i) \cap T| \le \frac{1}{4\varepsilon^2} - \frac{c}{2\varepsilon}$, and using Claim 3.5,

$$\mathbf{Pr}[i \in B] \le \frac{|B| - \frac{4}{5}|L_{\text{high}}|}{|L(G_j)|} \le \frac{1}{4}.$$

Thus, Bob can decide between the two cases with error probability at most 1/4. Overall, it follows that Bob can solve the Gap-Hamming-Distance problem for (s_i, t), with overall error probability at most $1/4 + 0.06 < 1/3$, as required to prove the theorem.

Proof:[Proof of Claim 3.5] By basic properties of the Binomial distribution (or the Berry-Esseen Theorem), there are absolute constants $\frac{1}{5} \le K_1 \le K_2 \le 5$ such that for each vertex $v \in L(G_j)$,

$$\mathbf{Pr}\Big[v \in L_{\text{high}}\Big] = \mathbf{Pr}\Big[|N(v) \cap T| \ge \tfrac{1}{4\varepsilon^2} + \tfrac{c}{\varepsilon}\Big] \in [\tfrac{1}{2} - K_2 c, \tfrac{1}{2} - K_1 c].$$

Denoting $Z = |L_{\text{high}}|$, then $\mathbf{E}[Z] \in [\frac{1}{2\varepsilon^2} - \frac{K_2 c}{\varepsilon^2}, \frac{1}{2\varepsilon^2} - \frac{K_1 c}{\varepsilon^2}]$. We have by Hoeffding's inequality,

$$\mathbf{Pr}\Big[\,|Z - \mathbf{E}[Z]| > \tfrac{K_1 c}{\varepsilon^2}\Big] \le 2e^{-\frac{1}{2}(K_1 c)^2 (1/\varepsilon^2)} \le 0.02.$$

Thus, with probability at least 0.98, we have both bounds

$$Z \le \mathbf{E}[Z] + \tfrac{K_1 c}{\varepsilon^2} \le \tfrac{1}{2\varepsilon^2} - \tfrac{K_1 c}{\varepsilon^2} + \tfrac{K_1 c}{\varepsilon^2} = \tfrac{1}{2\varepsilon^2}, \text{ and}$$
$$Z \ge \mathbf{E}[Z] - \tfrac{K_1 c}{\varepsilon^2} \ge \tfrac{1}{2\varepsilon^2} - \tfrac{K_1 c}{\varepsilon^2} - \tfrac{K_2 c}{\varepsilon^2} \ge (\tfrac{1}{2} - 2K_2 c)\tfrac{1}{\varepsilon^2} \ge (\tfrac{1}{2} - 10c)\tfrac{1}{\varepsilon^2}.$$

The claim follows by noting $|L(G_j)| = 1/\varepsilon^2$. □

Proof:[Proof of Lemma 3.4] We now show how Bob creates the "list" $B \subset L(G_j)$ of size $|B| = \frac{1}{2\varepsilon^2}$. Bob estimates the cut value for $S \cup T$ for every subset $S \subseteq L(G_j)$ of size exactly $\frac{1}{2\varepsilon^2}$. Observe that the cut value for a given S is

$$\delta(S \cup T) = \sum_{v \in S} \deg(v) + \sum_{u \in T} \deg(u) - 2\sum_{v \in S} |N(v) \cap T|.$$

The cut value is bounded by the number of edges in G_j, which is at most $1/\varepsilon^4$, and since the data structure maintains all the cut values within factor $1 + \gamma\varepsilon$ for an arbitrarily small constant $\gamma > 0$, the additive error on each cut value is at most γ/ε^3 . Further, we can assume Bob knows the exact degrees of all vertices (by adding them to the sketch, using $O(n \log \frac{1}{\varepsilon})$ bits), which he can subtract off, and since scaling by $-1/2$ can only shrink the additive error, we can define the "normalized" cut value

$$n(S, T) = \sum_{v \in S} |N(v) \cap T|,$$

which Bob can estimate within additive error γ/ε^3 . Bob's algorithm is to compute these estimates for all the values $n(S, T)$, and output a set S that maximizes his estimate for $n(S, T)$ as the desired list $B \subset L(G_j)$.

Let us now analyze the success probability of Bob's algorithm. For each vertex $v \in L(G_j)$, let $f(v) = |N(v) \cap T|$. Observe that each $f(v)$ has a Binomial distribution $B(\frac{1}{\varepsilon^2}, \frac{1}{4})$, and they are independent of each other. We will need the following bounds on the typical values of some order statistics when taking multiple samples from such a Binomial distribution. Recall that the *k-th order statistic* of a sample (multiset) $x_1, \ldots, x_m \in \mathbb{R}$ is the k-th smallest element in that sample. The following claim is proved further below.

Claim 3.6 *Let $\{X_j\}_{j=1,\ldots,m}$ be independent random variables with Binomial distribution $B(t, \frac{1}{4})$. Let $\alpha \in (0, \frac{1}{2})$ such that $(\frac{1}{2} + \alpha)m$ is integral, and both $t, m \geq 10/\alpha^2$. Then*

$$\mathbf{Pr}\Big[\text{the } (\tfrac{1}{2} - \alpha)m \text{ order statistic of } \{X_j\} \text{ is } \leq \tfrac{1}{4}t - \tfrac{\alpha}{10}\sqrt{t}\Big] \geq 0.99, \text{ and}$$
$$\mathbf{Pr}\Big[\text{the } (\tfrac{1}{2} + \alpha)m \text{ order statistic of } \{X_j\} \text{ is } \geq \tfrac{1}{4}t + \tfrac{\alpha}{10}\sqrt{t}\Big] \geq 0.99.$$

Sort the vertices $v \in L(G_j)$ by their $f(v)$ value, and denote them by $v_1, \ldots, v_{1/\varepsilon^2}$ such that $f(v_i) \leq f(v_{i+1})$. Applying the claim (for $\alpha = 0.05$ and $t, m = \frac{1}{\varepsilon^2}$), we see that with probability at least 0.98, the difference

$$f(v_{0.55/\varepsilon^2}) - f(v_{0.45/\varepsilon^2}) \geq 0.01/\varepsilon. \tag{3}$$

We assume henceforth this event indeed occurs. Let S^* include the $\frac{1}{2\varepsilon^2}$ vertices $v \in L(G_j)$ with largest $f(v)$, i.e., $S^* = \{v_j\}_{j>0.5/\varepsilon^2}$, and let $S' \subset L(G_j)$ be any subset of the same size such that at least $\frac{1}{10}$-fraction of its vertices are not included in S^* (i.e., their order statistic in $L(G_j)$ is at most $\frac{1}{2\varepsilon^2}$). Then we can write

$$n(S^*, T) = \sum_{j \in S^*} f(v) = \sum_{j > 0.5/\varepsilon^2} f(v_j),$$
$$n(S', T) = \sum_{j \in S'} f(v) \leq \sum_{j > 0.6/\varepsilon^2} f(v_j) + \sum_{0.4/\varepsilon^2 < j \leq 0.5/\varepsilon^2} f(v_j).$$

Now subtract them

$$n(S', T) - n(S^*, T) = \sum_{0.5/\varepsilon^2 < j \leq 0.6/\varepsilon^2} f(v_j) - \sum_{0.4/\varepsilon^2 < j \leq 0.5/\varepsilon^2} f(v_j),$$

observe that elements in the normalized interval $(0.5, 0.55]$ dominate those in $(0.45, 0.5]$,

$$\geq \sum_{0.55/\varepsilon^2 < j \leq 0.6/\varepsilon^2} f(v_j) - \sum_{0.4/\varepsilon^2 < j \leq 0.45/\varepsilon^2} f(v_j)$$

and bound the remaining elements using (3),

$$\geq (0.05/\varepsilon^2)\big[f(v_{0.55/\varepsilon^2}) - f(v_{0.45/\varepsilon^2})\big] \geq 0.0005/\varepsilon^3.$$

Bob's estimate for each of the values $n(S^*, T)$ and $n(S', T)$ has additive error at most γ/ε^3, and therefore for suitable $\gamma = 10^{-4}$, the list B Bob computes cannot be this set S'. Thus, Bob's list B must contain at least 9/10-fraction of S^*, i.e., the $\frac{1}{2\varepsilon^2}$ vertices $v \in L(G_j)$ with highest $f(v)$.

Recall from Claim 3.5 that with probability at least 0.98, we have $\frac{1}{4\varepsilon^2} \le |L_{\text{high}}| \le \frac{1}{2\varepsilon^2}$, and assume henceforth this event occurs. Since S^* includes the $\frac{1}{2\varepsilon^2}$ vertices with highest f-value, it must contain all the vertices of L_{high}, i.e., $L_{\text{high}} \subseteq S^*$. We already argued that Bob's list B contains all but at most $\frac{1}{10}|S^*| = \frac{1}{20\varepsilon^2}$ vertices of S^*, and thus

$$\frac{|L_{\text{high}} \setminus B|}{|L_{\text{high}}|} \le \frac{|S^* \setminus B|}{|L_{\text{high}}|} \le \frac{\frac{1}{20\varepsilon^2}}{\frac{1}{4\varepsilon^2}} = \frac{1}{5}.$$

This bound holds with probability at least 0.96 (because of two events that we ignored, each having probability at most 0.02) and this proves Lemma 3.4. □

Proof:[Proof of Claim 3.6] The $(\frac{1}{2}-\alpha)m$ order statistic of $\{X_j\}$ is smaller or equal to $T = \frac{1}{4}t - \frac{\alpha}{10}\sqrt{t}$ if and only if at least $(\frac{1}{2}-\alpha)m$ elements are smaller or equal to T, which can be written as $\sum_{j=1}^{m} \mathbb{1}_{\{X_j \le T\}} \ge (\frac{1}{2}-\alpha)m$.

Now fix $j \in \{1, \dots, t\}$. Then

$$\mathbf{Pr}[X_j \le T] = \mathbf{Pr}[X_j \le \tfrac{1}{4}t] \cdot \mathbf{Pr}[X_j \le T \mid X_j \le \tfrac{1}{4}t], \tag{4}$$

and by the Binomial distribution's relationship between mean and median, $\mathbf{Pr}[X_j \le \frac{1}{4}t] \ge \frac{1}{2}$. Elementary but tedious calculations (or the Berry-Esseen Theorem) show there is an absolute constant $K \in (0,5)$ such that

$$\mathbf{Pr}\Big[\tfrac{1}{4}t - \tfrac{\alpha}{10}\sqrt{t} < X_j \le \tfrac{1}{4}t\Big] \le K\tfrac{\alpha}{10} \cdot \mathbf{Pr}\Big[X_j \le \tfrac{1}{4}t\Big],$$

and plugging into (4), we obtain $\mathbf{Pr}[X_j \le T] \ge \frac{1}{2}(1 - K\frac{\alpha}{10}) \ge \frac{1}{2} - \frac{1}{2}\alpha$.

Now bound the expectation by $\mathbf{E}[\sum_{j=1}^{m} \mathbb{1}_{\{X_j \le T\}}] \ge (\frac{1}{2} - \frac{1}{2}\alpha)m$, and apply Hoeffding's inequality,

$$\mathbf{Pr}\Big[\sum_j \mathbb{1}_{\{X_j \le T\}} < (\tfrac{1}{2}-\alpha)m\Big] \le e^{-\frac{1}{2}(\frac{1}{2}\alpha)^2 m} = e^{-\alpha^2 m/8} \le 0.01,$$

where the last inequality follows since $\alpha^2 m$ is sufficiently large. □

3.2 The Communication Lower Bound

We now prove Theorem 3.3 (see Theorem 3.10 below), by considering distributional communication problems between two parties, Alice and Bob, as defined below. We restrict attention to the one-way model, in which Alice sends to Bob a single message M that is a randomized function of her input (using her private randomness), and Bob outputs the answer.

3.2.1 Distributional Versions of Gap-Hamming-Distance.

Recall that our analysis is asymptotic for ε tending to 0, and let $0 < c < 1$ be a parameter, considered to be a sufficiently small constant. Alice's input is $S \in \{0,1\}^{\frac{1}{\varepsilon^2}}$, Bob's input is $T \in \{0,1\}^{\frac{1}{\varepsilon^2}}$, where the Hamming weights are $\text{wt}(S) = \text{wt}(T) = \frac{1}{2\varepsilon^2}$, and Bob needs to evaluate the partial function

$$f_c(S,T) = \begin{cases} 1 & \text{if } \Delta(S,T) \ge \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}; \\ 0 & \text{if } \Delta(S,T) \le \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}. \end{cases}$$

The distribution μ we place on the inputs (S,T) is the following: S is chosen uniformly at random with $\mathrm{wt}(S) = \frac{1}{2\varepsilon^2}$, and then with probability $\frac{1}{2}$, we choose T uniformly at random with $\mathrm{wt}(T) = \frac{1}{2\varepsilon^2}$ subject to the constraint that $\Delta(S,T) \geq \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$, while with the remaining probability $\frac{1}{2}$, we choose T uniformly at random with $\mathrm{wt}(T) = \frac{1}{2\varepsilon^2}$ subject to the constraint that $\Delta(S,T) \leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$. We say Alice's message $M = M(S)$ is δ-error for (f_c, μ) if Bob has a reconstruction function R for which

$$\mathbf{Pr}_{(S,T)\sim\mu,\text{ private randomness}}[R(M,T) = f_c(S,T)] \geq 1 - \delta.$$

Now consider a related but different distributional problem. Alice and Bob have $S, T \in \{0,1\}^{\frac{1}{\varepsilon^2}}$, respectively, each of Hamming weight exactly $\frac{1}{2\varepsilon^2}$, and Bob needs to evaluate the function

$$g(S,T) = \begin{cases} 1 & \text{if } \Delta(S,T) > \frac{1}{2\varepsilon^2}; \\ 0 & \text{if } \Delta(S,T) \leq \frac{1}{2\varepsilon^2}. \end{cases}$$

We place the following distribution ζ on the inputs (S,T): S and T are chosen independently and uniformly at random among all vectors with Hamming weight exactly $\frac{1}{2\varepsilon^2}$. We say a message M is δ-error for (g,ζ) if Bob has a reconstruction function R for which

$$\mathbf{Pr}_{(S,T)\sim\zeta,\text{ private randomness}}[R(M,T) = g(S,T)] \geq 1 - \delta.$$

Let $I(S;M) = H(S) - H(S|M)$ be the mutual information between S and M, where H is the entropy function. Define $\mathrm{IC}_{\mu,\delta}(f_c) = \min_{\delta\text{-error } M \text{ for } (f_c,\,\mu)} I(S;M)$ and $\mathrm{IC}_{\zeta,\delta}(g) = \min_{\delta\text{-error } M \text{ for } (g,\,\zeta)} I(S;M)$.

Lemma 3.7 *For all $\delta > 0$, $\mathrm{IC}_{\mu,\delta}(f_c) \geq \mathrm{IC}_{\zeta,\delta+O(c)}(g)$.*

Proof: It suffices to show that if M is δ-error for (f_c,μ), then M is $(\delta + O(c))$-error for (g,ζ). Since M is δ-error for (f_c,μ), Bob has a reconstruction function R for which

$$\mathbf{Pr}_{(S,T)\sim\mu,\text{ private randomness}}[R(M,T) = f_c(S,T)] \geq 1 - \delta.$$

Now consider $\mathbf{Pr}_{(S,T)\sim\zeta,\text{ private randomness}}[R(M,T) = g(S,T)]$. Observe that whenever (S,T) lies in the support of μ, if $R(M,T) = f_c(S,T)$, then $R(M,T) = g(S,T)$. The probability that (S,T) lies in the support of μ is $1 - O(c)$, by standard anti-concentration arguments of the Binomial distribution (or the Berry-Esseen Theorem), and conditioned on this event we have that (S,T) is distributed according to μ. Hence, $\mathbf{Pr}_{(S,T)\sim\zeta,\text{ private randomness}}[R(M,T) = g(S,T)] \geq [1 - O(c)][1-\delta] \geq 1 - O(c) - \delta$. $\square$

We now lower bound $\mathrm{IC}_{\zeta,\delta}(g)$.

Lemma 3.8 *For $\delta_0 > 0$ a sufficiently small constant, $IC_{\zeta,\delta_0}(g) = \Omega\left(\frac{1}{\varepsilon^2}\right)$.*

Proof: We use the following lower bound of Braverman, Garg, Pankratov and Weinstein [BGPW13] for the following $h_c(S,T)$ problem. Like before, Alice has $S \in \{0,1\}^{\frac{1}{\varepsilon^2}}$, Bob has $T \in \{0,1\}^{\frac{1}{\varepsilon^2}}$, and needs to evaluate the partial function

$$h_c(S,T) = \begin{cases} 1 & \text{if } \Delta(S,T) \geq \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}; \\ 0 & \text{if } \Delta(S,T) \leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}. \end{cases}$$

However, now $\mathrm{wt}(S)$ and $\mathrm{wt}(T)$ may be arbitrary. Moreover, S and T are chosen independently and uniformly at random from $\{0,1\}^{\frac{1}{\varepsilon^2}}$. Denote this by $(S,T) \sim \eta$. Now it may be the case that $\left|\Delta(S,T) - \frac{1}{2\varepsilon^2}\right| < \frac{c}{\varepsilon}$, in which case Bob's output is allowed to be arbitrary. A message M is δ-error for (h_c, η) if Bob has a reconstruction function R for which

$$\mathbf{Pr}_{(S,T)\sim\eta,\text{ private randomness}}\left[(R(M,T) = h_c(S,T)) \wedge \left(\left|\Delta(S,T) - \frac{1}{2\varepsilon^2}\right| \geq \frac{c}{\varepsilon}\right)\right] \geq 1 - \delta.$$

It was proved in [BGPW13] that for a sufficiently small constant $\delta > 0$,

$$\mathrm{IC}_{\eta,\delta}(h_1) = \min_{\delta\text{-error } M \text{ for } (h_1,\eta)} I(S;M) \geq \frac{C}{\varepsilon^2},$$

for an absolute constant $C > 0$. We show how to apply this result to prove the lemma.

An immediate corollary of this result is that $\mathrm{IC}_{\eta,\delta}(g) = \min_{\delta\text{-error } M \text{ for } (g,\eta)} I(S;M) \geq \frac{C}{\varepsilon^2}$. Indeed, if M is δ-error for (g,η), then it is also δ-error for (h_1,η).

Now let M be a δ-error protocol for (g,ζ). Consider the following randomized protocol M' for g with inputs distributed according to η. Given S, Alice computes $s = \mathrm{wt}(S)$. If $s < \frac{1}{2\varepsilon^2}$, Alice randomly chooses $\frac{1}{2\varepsilon^2} - s$ coordinates in S that are equal to 0 and replaces them with a 1, otherwise she randomly chooses $s - \frac{1}{2\varepsilon^2}$ coordinates in S that are equal to 1 and replaces them with a 0. Let S' be the resulting vector. Alice sends $M(S')$ to Bob, i.e., $M'(S) = M(S')$. Given the message $M(S')$ and his input T, Bob first computes $t = \mathrm{wt}(T)$. If $t < \frac{1}{2\varepsilon^2}$, Bob randomly chooses $\frac{1}{2\varepsilon^2} - t$ coordinates in T which are equal to 0 and replaces them with a 1, otherwise he randomly chooses $t - \frac{1}{2\varepsilon^2}$ coordinates in T which are equal to 1 and replaces them with a 0. Let T' be the resulting vector. Suppose R is such that $\mathbf{Pr}_{(S',T')\sim\zeta,\text{ private randomness}}[R(M(S'),T') = g(S',T')] \geq 1 - \delta$. Bob outputs $R(M(S'),T')$.

We now lower bound $\mathbf{Pr}[g(S',T') = g(S,T)]$, where the probability is over $(S,T) \sim \eta$ and the random choices of Alice and Bob for creating S', T' from S, T, respectively. First, the number of coordinates changed by Alice or Bob is $r = \Theta(1/\varepsilon)$ with arbitrarily large constant probability. Since S and T are independent and uniformly random, after performing this change, the Hamming distance on these r coordinates is $\frac{r}{2} \pm O(\sqrt{r})$ with arbitrarily large constant probability. Finally, $\left|\Delta(S',T') - \frac{1}{2\varepsilon^2}\right| = \omega(\sqrt{r})$ with arbitrarily large constant probability. Hence, with arbitrarily large constant probability, $g(S',T') = g(S,T)$. It follows that $\mathbf{Pr}[g(S',T') = g(S,T)] \geq 1 - \gamma$ for an arbitrarily small constant $\gamma > 0$, and therefore if R' describes the above reconstruction procedure of Bob, then $\mathbf{Pr}_{(S,T)\sim\eta,\text{ private randomness}}[R'(M'(S),T) = g(S,T)] \geq 1 - \gamma - \delta$.

Hence, M' is a $(\delta+\gamma)$-error protocol for (g,η). We now bound $I(M';S)$ in terms of $I(M;S')$. Let J be an indicator random variable for the event $\mathrm{wt}(S) \in \left[\frac{1}{2\varepsilon^2} - \frac{1}{\varepsilon^{3/2}}, \frac{1}{2\varepsilon^2} + \frac{1}{\varepsilon^{3/2}}\right]$. Then $\mathbf{Pr}[J = 1] = 1 - o(1)$, where $o(1) \to 0$ as $\varepsilon \to 0$. Since conditioning on a random variable Z can change the mutual information by at most $H(Z)$, we have

$$I(M';S) \leq I(M';S \mid J) + H(J) \leq I(M';S \mid J=1) + 1. \tag{5}$$

S is a probabilistic function of S', which if $J = 1$, is obtained by changing at most $1/\varepsilon^{3/2}$ randomly chosen coordinates $A_1, \ldots, A_{1/\varepsilon^{3/2}}$ of S' from 0 to 1 or from 1 to 0. By the data processing inequality and the chain rule for mutual information,

$$\begin{aligned} I(M'; S \mid J = 1) &\leq I(M'; S', A_1, \ldots, A_{1/\varepsilon^{3/2}} \mid J = 1) \\ &= I(M'; S' \mid J = 1) + \sum_{\ell=1}^{1/\varepsilon^{3/2}} I(M'; A_\ell \mid J = 1, A_1, \ldots, A_{\ell-1}) \\ &\leq I(M'; S' \mid J = 1) + O\left(\frac{\log(1/\varepsilon)}{\varepsilon^{3/2}}\right). \end{aligned} \tag{6}$$

Observe that the joint distribution of $M'(S')$ and S' is independent of J, and moreover is equal to the joint distribution of $M(S')$ and $S' \sim \zeta$. We can take M to be a δ-error protocol for (g, ζ) for which $I(M(S'); S') = \mathrm{IC}_{\zeta,\delta}(g)$. Combining this with (5) and (6), $I(M'; S) \leq \mathrm{IC}_{\zeta,\delta}(g) + O\left(\frac{\log(1/\varepsilon)}{\varepsilon^{3/2}}\right)$. Now since M' is a $(\delta + \gamma)$-error protocol for (g, η), we have $I(M'; S) \geq \mathrm{IC}_{\eta,\delta+\gamma}(g) \geq \frac{C}{\varepsilon^2}$, provided δ and γ are sufficiently small constants. It follows that $\mathrm{IC}_{\zeta,\delta}(G) \geq \frac{C}{\varepsilon^2} - O\left(\frac{\log(1/\varepsilon)}{\varepsilon^{3/2}}\right) \geq \frac{C}{2\varepsilon^2}$, as desired. □

Corollary 3.9 *For sufficiently small constants $\delta, c > 0$, $\mathrm{IC}_{\mu,\delta}(f_c) = \Omega(1/\varepsilon^2)$.*

Proof: This follows by combining Lemmas 3.7 and 3.8. □

3.2.2 n-fold Version of Gap-Hamming-Distance.

We now consider the n-fold problem in which Alice is given n strings $S_1, \ldots, S_n \in \{0,1\}^{1/\varepsilon^2}$, and Bob has an index $I \in [n]$ together with one string $T \in \{0,1\}^{1/\varepsilon^2}$. Here $(S_I, T) \sim \zeta$, while S_j for $j \neq I$, are chosen independently and uniformly at random from all Hamming weight $\frac{1}{2\varepsilon^2}$ vectors. Thus the joint distribution of $S_1, \ldots, S_n$ is n i.i.d. strings drawn uniformly from $\{0,1\}^{1/\varepsilon^2}$ subject to each of their Hamming weights being $\frac{1}{2\varepsilon^2}$. Here I is drawn independently and uniformly from $[n]$. We let ν denote the resulting input distribution.

We consider the one-way two-party model in which Alice sends a single, possibly randomized message M of her inputs $S_1, \ldots, S_n$, and Bob needs to evaluate $h(S_1, \ldots, S_n, T) = f_c(S_I, T)$. We say M is δ-error for (h, ν) if Bob has a reconstruction function R for which

$$\mathbf{Pr}_{\text{inputs}\sim\nu,\text{ private randomness}}\left[(R(M,T,I) = f_c(S_I,T)) \wedge \left(\left|\Delta(S_I,T) - \frac{1}{2\varepsilon^2}\right| \geq \frac{c}{\varepsilon}\right)\right] \geq 1 - \delta.$$

Let $\mathrm{IC}_{\nu,\delta}(h) = \min_{\delta\text{-error } M \text{ for } (h,\nu)} I(S_1, \ldots, S_n; M)$.

Theorem 3.10 *For a sufficiently small constant $\delta > 0$, $\mathrm{IC}_{\nu,\delta}(h) = \Omega(n/\varepsilon^2)$. In particular, the distributional one-way communication complexity of h under input distribution ν is $\Omega(n/\varepsilon^2)$.*

Proof: Say an index $i \in [n]$ is *good* if

$$\mathbf{Pr}_{\text{inputs}\sim\nu,\text{ private randomness}}\left[(R(M,T,I) = f_c(S_I,T)) \wedge \left(\left|\Delta(S_I,T) - \frac{c}{2\varepsilon^2}\right| \geq \frac{1}{\varepsilon}\right) \mid I = i\right] \geq 1 - 2\delta.$$

By a union bound, there are at least $n/2$ good $i \in [n]$. By the chain rule for mutual information and using that the S_i are independent and conditioning does not increase entropy,

$$I(M; S_1, \ldots, S_n) \geq \sum_{i=1}^{n} I(M; S_i) \geq \sum_{\text{good } i} I(M; S_i).$$

We claim that for each good i, $I(M; S_i) \geq \mathrm{IC}_{\mu,2\delta}(f_c)$. Indeed, consider the following protocol M_i for f_c under distribution μ. Alice, given her input S for f_c, uses her private randomness to sample S_j for all $j \neq i$ independently and uniformly at random from $\{0,1\}^{1/\varepsilon^2}$ subject to each of their Hamming weights being $\frac{1}{2\varepsilon^2}$. Bob sets $I = i$ and uses his input T for f_c as his input for h. Since i is good, it follows that M_i is 2δ-error for (f_c, ζ). Hence $I(M; S_i) \geq \mathrm{IC}_{\mu,2\delta}(f_c)$, which by Corollary 3.9, is $\Omega(1/\varepsilon^2)$ provided $\delta > 0$ is a sufficiently small constant. Hence, $\mathrm{IC}_{\nu,\delta} = \Omega(n/\varepsilon^2)$. Since $\mathrm{IC}_{\nu,\delta}(h) \leq I(M; S_1, \ldots, S_n)$ for each δ-error M for (h, ν), and $I(M; S_1, \ldots, S_n) \leq H(M)$ which is less than the length of M, the communication complexity lower bound follows. $\square$

3.3 Cut-sparsifiers Lower Bound

We now prove Theorem 3.2. Fix n and ε, and assume that every n-vertex graph has a $(1+\gamma\varepsilon/2)$-cut sparsifier with at most s edges, for a small constant $\gamma > 0$ to be determined later. We wish to prove a lower bound on s. Consider then the random graph G constructed in the proof of Theorem 3.1, as a disjoint union of graphs $\{G_j : j \in [\varepsilon^2 n/2]\}$, each being a bipartite graph with $1/\varepsilon^2$ vertices in each side. By our assumption above, such G (always) has a subgraph H which is a $(1+\gamma\varepsilon/2)$-cut sparsifier having at most s edges. By Theorem 3.1, answering all possible cut queries correctly (in the sense of approximation factor $1 \pm \varepsilon$ with probability 1) requires sketch size $\Omega(n/\varepsilon^2)$ bits. In fact, by inspecting the proof (specifically, of Lemma 3.4) the above holds even if the correctness (1) holds only for cut queries contained in a single G_j, i.e., queries $S \cup T$ for $S \subset L(G_j)$ and $T \subset R(G_j)$; and (2) allows for each cut value an additive error of γ/ε^3, where $\gamma = 10^{-4}$. The idea now is to encode H using $m \approx s$ bits in a way that suffices to correctly answer all such cut queries i.e., in the context of Lemma 3.4, Alice will encode H and send it as her sketch to Bob. The sketch-size bound $m \geq \Omega(n/\varepsilon^2)$ we proved for G would then imply a similar bound on s.

Consider then the sparsifier H, which is an edge-weighted graph, while the edges of G all have unit weight. Observe that H must be a union of disjoint graphs H_j on $L(G_j) \cup R(G_j)$ for $j \in [\varepsilon^2 n/2]$, because H must preserve the corresponding cut, which has value zero. Let s_j denote the number of edges in H_j. It will be convenient to consider each such graph separately, so fix for now some $j \in [\varepsilon^2 n/2]$.

Consider first the case $s_j \leq \gamma^2/(6\varepsilon^4)$, and let us show how to encode H_j. Construct from H_j another graph H'_j by rounding every non-integral edge weight to one of its two nearby integers independently at random, in an unbiased manner. Specifically, each edge weight $w > 0$ is rounded upwards to $w' = \lceil w \rceil$ with probability $w - \lfloor w \rfloor$, and downwards to $w' = \lfloor w \rfloor$ otherwise. Now consider a fixed cut query $S \cup T$ in G_j, denoting by $\delta_H(S \cup T)$ its cut value in H, and similarly for H'. Then $\mathbf{E}[\delta_{H'}(S \cup T)] = \delta_H(S \cup T)$, and since the number of edges participating in this cut (in H_j) is at most $s_j \leq \gamma^2/(6\varepsilon^4)$, by Hoeffding's inequality for $t = \gamma/(2\varepsilon^3)$,

$$\mathbf{Pr}\Big[\, |\delta_{H'}(S \cup T) - \delta_H(S \cup T)| > t \Big] \leq e^{-2t^2/s_j} \leq e^{-3/\varepsilon^2}.$$

Applying a union bound over at most $2^{2/\varepsilon^2}$ possible cut queries $S \cup T$, we see that there exists H'_j (it is in fact obtained with high probability) such that for every cut query, the cut value in H'_j is within additive $\gamma/(2\varepsilon^3)$ from the cut value in H_j, which in turn is within additive $\gamma\varepsilon/2 \cdot 1/\varepsilon^4 = \gamma/(2\varepsilon^3)$ from the cut value in G_j. Hence, such H'_j approximates all the cut values in G_j sufficiently well for our intended application, and Alice's sketch will thus encode H'_j instead of H_j. To simplify the description, let us include in H'_j also edges of weight zero, and then H'_j has exactly s_j edges (same as in H_j). We further claim that the total edge-weight in H'_j is at most $2/\varepsilon^4$. To see this, observe that (i) the total edge-weight in H'_j (and similarly for G_j) is exactly twice the expected cut value of a random query in that graph; and (ii) this expected cut value in H'_j differs from the respective expectation in G_j by at most additive $\gamma/(2\varepsilon^3)$. It follows that the total edge-weight in H'_j is at most $1/\varepsilon^4$ larger than that in G_j, which in turn is at most $1/\varepsilon^4$.

The encoding of H'_j has two parts, which describe separately the edges of H'_j (without their weights), and their weights (assuming the edges are known). Since H'_j has $1/\varepsilon^2$ vertices in each side, the number of possibilities for s_j edges (without their weights) among $\binom{2/\varepsilon^2}{2} \leq 2/\varepsilon^4$ vertex pairs is at most $\binom{2/\varepsilon^4}{s_j}$. Given the identity of s_j edges in H'_j, the number of possibilities for their weights (recall the weights are integral and add up to at most $2/\varepsilon^4$) is at most $\binom{s_j+2/\varepsilon^4}{s_j} \leq \binom{4/\varepsilon^4}{s_j}$. We conclude that H'_j can be encoded, on its two parts, using $O(\log \binom{4/\varepsilon^4}{s_j})$ bits.

The second case $s_j > \gamma^2/(6\varepsilon^4)$ is very simple, and just encodes the original G_j (instead of encoding H_j), which trivially provides the value of every cut query inside G_j exactly. A straightforward encoding of G_j takes $1/\varepsilon^4$ bits.

Concatenating these encodings over all $j \in [\varepsilon^2 n/2]$, yields a sketch that can approximate the value of all the needed cut queries (those that are inside a single G_j) within additive error γ/ε^3. It remains to bound the size of this sketch. The number of j's that fall into the second case $s_j > \gamma^2/(6\varepsilon^4)$ is at most $\frac{\sum_j s_j}{\gamma^2/(6\varepsilon^4)} = \frac{6\varepsilon^4 s}{\gamma^2}$, and thus the total size of their encodings is at most $6s/\gamma^2$ bits. For j's that fall into the first case, we use the fact $\binom{p}{k} \cdot \binom{p'}{k'} \leq \binom{p+p'}{k+k'}$, and get that the total size of their encodings is at most $\sum_j O(\log \binom{4/\varepsilon^4}{s_j}) \leq O(\log \binom{2n/\varepsilon^2}{s}) = O(s \log(\varepsilon^{-2} n/s))$ bits. Altogether, there is a sketch of size $m = O(s(\gamma^{-2} + \log(\varepsilon^{-2} n/s)))$ bits that encodes all the relevant cuts in G within the desired accuracy. Recalling that γ is a constant and our sketch-size lower bound $m \geq \Omega(n/\varepsilon^2)$ (from Lemma 3.4 and Theorem 3.3), we conclude that the number of edges in H is $s \geq \Omega(n/\varepsilon^2)$, which proves Theorem 3.2.

4 Symmetric Diagonally Dominant Matrices, "For Each" Model

In this section, we consider sketching SDD matrices with "for each" guarantee. That is, we want a sketch $\mathrm{sk}(A)$ of an $n \times n$ SDD matrix A, which can be used to produce a $(1+\varepsilon)$-approximation of $x^T A x$ for any fixed $x \in \mathbb{R}^n$ with constant probability (say 0.9). Note that we can always use standard amplification argument (run the query on $O(\log n)$ independent copies of sketch, and return the median outcome) to boost the success probability to $1 - 1/n^{100}$.

Recall that we have a reduction from SDD matrices to Laplacian matrices (see Appendix B). We thus only need to sketch Laplacian matrices, which we first do for *cut queries* $x \in \{0,1\}^n$ in Section 4.1, and then for *spectral queries* $x \in \mathbb{R}^n$ in Section 4.2 (for the main results, see Theorems 4.12 and 4.28, respectively).

Algorithm 1: Cut-S1(G, ε)

```
  Input: An S1-Graph $G = (V, E, w)$; a quality control parameter $\varepsilon \in [1/n, 1/30]$
  Output: A $(1 + 21\varepsilon, 1/9)$-cut-sketch sk$(G)$ of $G$
1 Let $s = 1/\varepsilon$;
2 sk$(G) \leftarrow \emptyset$;
3 Add $\{\delta_u(G) \mid u \in V\}$ to sk$(G)$;
4 for $u \in V$ do
5     $E_u \leftarrow \{(u, v) \mid v \in V\}$;
6     Sample (with replacement) $s$ edges from $E_u$, where each time the probability of sampling
      $e = (u, v) \in E_u$ is $p_e = 1/d_u(G)$;
7     Add the sampled $s$ edges to sk$(G)$;
8 return $sk(G)$;
```

4.1 Laplacian Matrices with Cut Queries

Given a Laplacian matrix L, let $G = G(L) = (V, E, w)$ be the corresponding graph with $n = |V|$ vertices. In this section we will construct a sketch for G that can produce the weight of each *cut* up to factor $1 + \varepsilon$ with constant probability. Our work plan has three parts, each uses the previous one as a building block. We start with an important special case in Section 4.1.1, then extend it to all graphs with polynomial edge weights in Section 4.1.2, and finally extend that to all edge weights in Section 4.1.3.

We then turn to proving a nearly matching lower bound in Section 4.1.4.

4.1.1 Special Graphs and Special Queries

We start by sketching a class of special graphs with a special set of queries, which is sufficient to illustrate our basic ideas.

Definition 4.1 (S1-graph) *We say an undirected weighted graph $G = (V, E, w)$ is an* S1-graph *(reads "simple type-1 graph") if it satisfies the followings.*

1. *All edge weights are within factor 2 of each other, i.e., there is $\gamma > \varepsilon^2$ such that all $e \in E$ satisfy $w(e) \in [\gamma, 2\gamma)$.*
2. *The expansion constant of G is $\Gamma_G \geq \frac{1}{\varepsilon}$.*

We consider a special set of cut queries where $w(S, \bar{S}) \leq 5$.

The sketching algorithm for S1-graph and the special cut queries is described in Algorithm 1, which is fairly simple: we first add all weighted degrees of vertices to the sketch, and then for each vertex we sample a set of adjacent edges and store them in the sketch.

We first show the correctness of Algorithm 1. Let Y_u^v be the random variable denoting the number of times edge (u, v) is sampled at Line 6 in Algorithm 1. It is easy to see that

$$\mathbf{E}[Y_u^v] = \frac{s}{d_u(G)} \quad \text{and} \quad \mathbf{Var}\,[Y_u^v] = s\left(1 - \frac{1}{d_u(G)}\right)\frac{1}{d_u(G)} \leq \frac{s}{d_u(G)}. \tag{7}$$

Given a cut $(S,\bar{S})$ such that $|S| \le |V|/2$ and $w(S,\bar{S}) \le 5$, we can approximate the cut weight $w(S,\bar{S})$ by the following estimator:

$$I_G = \sum_{u\in S}\Big[\delta_u(G) - \frac{d_u(G)}{s}\sum_{v\in S,\ (u,v)\in E} Y_u^v w(u,v)\Big]. \tag{8}$$

Lemma 4.2 *Let $G=(V,E,w)$ be an S1-Graph and $(S,\bar{S})$ be a cut of G such that $|S| \le |V|/2$ and $w(S,\bar{S}) \le 5$, then I_G (defined in Equation (8)) is an unbiased estimator of $w(S,\bar{S})$. Furthermore, it gives a $(1+21\varepsilon, 1/9)$-approximation to $w(S,\bar{S})$ for such cut $(S,\bar{S})$.*

Proof: Since $\mathbf{E}[Y_u^v] = \frac{s}{d_u(G)}$ (by (7)), it follows that

$$\mathbf{E}[I_G] = \sum_{u\in S}\Big[\delta_u - \sum_{v\in S,\ (u,v)\in E} w(u,v)\Big] = w(S,\bar{S}).$$

We next analyze the variance of I_G. Recall that $|\partial(A,B)| = \sum_{e=(u,v)\in E}\mathbb{1}_{\{u\in A, v\in B\}}$ is the number of edges between sets A and B.

$$\begin{aligned}
&\mathbf{Var}\,[I_G]\\
=\ &\mathbf{Var}\left[\sum_{u\in S}\frac{d_u(G)}{s}\sum_{v\in S,\ (u,v)\in E} Y_u^v w(u,v)\right]\\
=\ &\sum_{u\in S}\frac{(d_u(G))^2}{s^2}\sum_{v\in S,\ (u,v)\in E}\mathbf{Var}\,[Y_u^v]\,(w(u,v))^2\\
\le\ &\sum_{u\in S}\frac{(d_u(G))^2}{s^2}\sum_{v\in S,\ (u,v)\in E}\frac{s}{d_u(G)}(2\gamma)^2 && \text{(by (7) and } w(u,v)\in[\gamma,2\gamma))\\
=\ &4\varepsilon\gamma^2\sum_{u\in S} d_u(G)d_u(G(S)) && (s=1/\varepsilon \text{ and } G(S) \text{ is the subgragh of } G \text{ induced by vertices in } S)\\
\le\ &4\varepsilon\gamma^2|S|\sum_{u\in S} d_u(G) && (d_u(G(S)) \le |S|)\\
\le\ &4\varepsilon\gamma^2|S|\cdot\Big[\big|\partial(S,\bar{S})\big| + 2|S|^2\Big] && \Big(\big|\partial(S,\bar{S})\big| = \sum_{u\in S} d_u(G) - 2\,|\partial(S,S)| \text{ and } |\partial(S,S)| \le |S|^2\Big)\\
\le\ &4\varepsilon^2\gamma^2(\big|\partial(S,\bar{S})\big|)^2\Big[1 + 2\varepsilon^2\,\big|\partial(S,\bar{S})\big|\Big] && (|S| \le \varepsilon\,\big|\partial(S,\bar{S})\big|)\\
\le\ &4\varepsilon^2(w(S,\bar{S}))^2\Big[1 + 2\gamma\,\big|\partial(S,\bar{S})\big|\Big] && (\gamma\,\big|\partial(S,\bar{S})\big| \le w(S,\bar{S}) \text{ and } \gamma \ge \varepsilon^2)\\
\le\ &44\varepsilon^2(w(S,\bar{S}))^2. && (\gamma\,\big|\partial(S,\bar{S})\big| \le w(S,\bar{S}) \le 5) \qquad (9)
\end{aligned}$$

Now by a Chebyshev's inequality, with probability at least 8/9, we have

$$\big|I_G - w(S,\bar{S}))\big| \le 3\sqrt{44\varepsilon^2(w(S,\bar{S}))^2} \le 21\varepsilon\, w(S,\bar{S}).$$

□

We summarize our result for S1-graph as below.

Theorem 4.3 *There is a sketching algorithm which given an S1-graph, outputs a $(1+\varepsilon, 1/9)$-cut-sketch of size $\tilde{O}(n/\varepsilon)$ for any cut $(S, \bar{S})$ such that $S \subseteq V$ and $w(S, \bar{S}) \le 5$.*

Proof: The correctness immediately follows from Lemma 4.2. We only need to show the size of the sketch. Note that in Algorithm 1 we only store all weighted degrees of vertices, and sample $1/\varepsilon$ edges for each vertex. Thus the size of sketch can be bounded by $\tilde{O}(n/\varepsilon)$. □

4.1.2 Graphs with Polynomial Weights

We now show how to sketch general graphs with polynomial integer weights, say, in $[1, n^5]$, by extending Algorithm 1. In Section 4.1.3 we will extend the construction to graphs with general weights.

Sketching. Our sketching algorithm consists of two components. The first component is a standard 1.2-cut sparsifier (recall that a $(1+\varepsilon)$-cut sparsifier is a sparse graph H on the same vertex-set as G that approximates every cut in G within a factor of $1+\varepsilon$). We can use the construction of Benczúr and Karger [BK96], or subsequent constructions [SS11, BSS14, FHHP11, KP12] (some of which produce a spectral sparsifier, which is only stronger); any of these methods will produce a graph H with $\tilde{O}(n)$ edges.

The second component is the main ingredient of the sketch, which preprocesses a given graph into a bunch of S1-graphs. Let $\tilde{C} = \{1.4^i \mid 0 \le i \le \log_{1.4} n^5\}$ be the set of size $O(\log n)$ such that each cut value in G is 1.4-approximated by some value $c \in \tilde{C}$. For each value $c \in \tilde{C}$ we construct a data structure D_c as follows.

1. By scaling all the edge weights, let us assume $c = 1$. We discard all edges e whose (scaled) weight $w(e) > 5$. Note that those edges are too heavy to be included into any cut whose weight is at most 5.
2. Apply the *importance sampling*: We sample each (remaining) edge $e \in E$ independently with probability $p_e = \min\{w(e)/\varepsilon^2, 1\}$, and assign each sampled edge e with a new weight $\tilde{w}(e) = w(e)/p_e$. Notice that $\tilde{w}(e) \in [\varepsilon^2, 5]$. (It may be convenient to consider the non-sampled edges as having weight $\tilde{w}(e) = 0$.) Let $\tilde{E}$ be the set of sampled edges. We further partition $\tilde{E}$ into $l = O(\log \frac{1}{\varepsilon})$ classes according to the (new) edge weights, namely, $\tilde{E} = L_1 \cup \cdots \cup L_l$ where $L_i = \{e \in \tilde{E} : \tilde{w}(e) \in (5 \cdot 2^{-i}, 5 \cdot 2^{-i+1}]\}$.
3. For each class L_i, we recursively partition the graph (V, L_i) as follows: For each connected component $P = (V_P, E_P)$ in (V, L_i) which contains a subset $S' \subset V_P$ of size $|S'| \le |V_P|/2$ such that $\left|\partial(S', \bar{S}')\right| / |S'| < 1/\varepsilon$ (where $\bar{S}' = V_P \setminus S'$), we replace P with its two vertex-induced subgraphs $P(S')$ and $P(\bar{S}')$ in (V, L_i), and store all edges in the cut $(S', \bar{S}')$ to Q_i. Once the recursive partitioning process is finished, denoting the resulting components by $\mathcal{P}_i$, we store in the sketch all the edges in Q_i connecting different connected components of $\mathcal{P}_i$.

This preprocessing step is described in Algorithm 2. Note that each component $P \in \mathcal{P}_i$ is an S1-graph.

Estimation. Given a query subset $S \subset V$, we first use the graph sparsifier H to compute $\tilde{c}$, a 1.2-approximation to the desired cut value $w(S, \bar{S})$, and then use the data structure D_c to answer the query, where $c \in \tilde{C}$ is an approximation to $\tilde{c}/(1.4)^2$, such that $c \in [\tilde{c}/(1.4)^3, \tilde{c}/1.4]$. This implies

```
Algorithm 2: Cut-Preprocessing($G, c, \varepsilon$)
  Input: A graph $G = (V, E, w)$ such that for any $e \in E$, $w(e) \in [1, n^5]$; a parameter $c > 0$; a
         quality control parameter $\varepsilon \in [1/n, 1/30]$
  Output: A set $\mathcal{P}$ of edge disjoint components of $G$ such that for each $P \in \mathcal{P}$, $\Gamma_P > 1/\varepsilon$;
          and a graph $Q$ induced by the rest of the edges in $G$.
1  $\mathcal{P} \leftarrow \emptyset$, $Q \leftarrow \emptyset$;
2  $\forall e \in E, w(e) \leftarrow w(e)/c$;
3  Discard any edge $e \in E$ with $w(e) > 5$;
4  Sample each edge $e \in E$ independently with probability $p_e \leftarrow \min\{w(e)/\varepsilon^2, 1\}$, and rescale
   its weight to $\tilde{w}(e) \leftarrow w(e)/p_e$ if the edge is sampled. Let $\tilde{E}$ denote the set of sampled edges;
5  Partition $\tilde{E}$ to $L_1 \cup \cdots \cup L_l$ with $l = O(\log \frac{1}{\varepsilon})$, where $L_i = \{e \in \tilde{E} : \ \tilde{w}(e) \in (5 \cdot 2^{-i}, 5 \cdot 2^{-i+1}]\}$;
6  foreach $L_i$ do
7      $\mathcal{P}_i \leftarrow \{(V, L_i)\}$, $Q_i \leftarrow \emptyset$;
8      while $\exists$ a connected component $P = (V_P, E_P) \in \mathcal{P}_i$ such that the expansion constant
       $\Gamma_P < 1/\varepsilon$ do
9          Find an arbitrary cut $(S', \bar{S}')$ (where $|S'| \le |V_P|/2$) in $P$ such that
           $\left|\partial(S', \bar{S}')\right| / |S'| < 1/\varepsilon$;
10         Replace $P$ with its two subgraphs $P(S')$ and $P(\bar{S}')$ in $\mathcal{P}_i$;
11         Add all edges in the cut $(S', \bar{S}')$ into $Q_i$;
12     $\mathcal{P} = \mathcal{P} \cup \mathcal{P}_i$, $Q = Q \cup Q_i$;
13 return ($\mathcal{P}$, $Q$);
```

that $w(S, \bar{S}) \in [c, 4c]$. Thus, by rescaling c to 1, we only need to estimate the (resclaled) cut weight $w(S, \bar{S})$, which is between 1 and 4.

The contribution to the cut $(S, \bar{S})$ from edges in each class L_i consists of two parts: (i) the total weight of cut edges between S and $\bar{S}$, i.e., $\sum_{e \in Q_i \cap \partial(S, \bar{S})} \tilde{w}(e)$; and (ii) the weight of cut edges inside each component $P \in \mathcal{P}_i$, which can be estimated using the sketches $\mathrm{sk}(P)$ for each $P \in \mathcal{P}_i$. We construct the estimator as follows.

$$\hat{w}(S, \bar{S}) = \sum_{i \in [l]} \Big(\sum_{P \in \mathcal{P}_i} I_P + \sum_{e \in Q_i \cap \partial(S, \bar{S})} \tilde{w}(e) \Big), \tag{10}$$

where I_P, defined in Equation (8), is the estimator of the cut weight in the component $P \in \mathcal{P}_i$.

In the rest of this section we show $\hat{w}(S, \bar{S})$ is a $1 + O(\varepsilon)$-approximation of the (scaled) cut weight $w(S, \bar{S})$ with constant probability. The following lemma guarantees that the importance sampling step preserves the cut weight.

Lemma 4.4 *Let $\tilde{G} = (V, \tilde{E}, \tilde{w})$ be the graph after the important sampling step (Line 4 of Algorithm 2). Then for any $S \subset V$ such that $w(S, \bar{S}) \in [c, 4c]$, with probability at least $8/9$, $|\tilde{w}(S, \bar{S}) - w(S, \bar{S})| \le 3\varepsilon w(S, \bar{S})$, where $w(S, \bar{S})$ and $\tilde{w}(S, \bar{S})$ are the cut weight of $(S, \bar{S})$ in G and $\tilde{G}$, respectively.*

Proof: Since $w(S, \bar{S}) \in [c, 4c]$, after rescaling (Line 2) c to 1, the cut weight $w(S, \bar{S})$ becomes a value in $[1, 4]$. Also note that the discarded edges (Line 3) will not affect the final cut estimation since they can not be part of the cut $(S, \bar{S})$, given that their weights are more than 5.

We now analyze the effect of the importance sampling step (Line 4) on the weight of the cut. It is easy to see that $\mathbf{E}[\tilde{w}(S,\bar{S})] = w(S,\bar{S})$, since every edge e that was not discarded satisfies $\mathbf{E}[\tilde{w}(e)] = w(e)$ (and more generally, it is a Horvitz-Thompson estimator), and its variance is

$$\mathbf{Var}\left[\tilde{w}(S,\bar{S})\right] = \sum_{e\in\partial(S,\bar{S})} \mathbf{Var}\left[\tilde{w}_e\right] = \sum_{e\in\partial(S,\bar{S})} w(e)^2/p_e - w(e)^2 \le \sum_{e\in\partial(S,\bar{S})} \varepsilon^2 w(e) = \varepsilon^2 w(S,\bar{S}),$$

where the inequality is verified separately for $p_e = 1$ and for $p_e = w(e)/\varepsilon^2$. Thus, by a Chebyshev's inequality, with probability at least $8/9$, we have $\left|\tilde{w}(S,\bar{S}) - w(S,\bar{S})\right| \le 3\varepsilon\sqrt{w(S,\bar{S})} \le 3\varepsilon \cdot w(S,\bar{S})$. □

Then, it suffices to show that the constructed estimator $\hat{w}(S,\bar{S})$ can approximate $\tilde{w}(S,\bar{S})$ well.

Lemma 4.5 *For any $S \subset V$ such that $w(S,\bar{S}) \in [1,4]$, with probability at least $8/9$, $|\hat{w}(S,\bar{S}) - \tilde{w}(S,\bar{S})| \le 21\varepsilon\tilde{w}(S,\bar{S})$.*

Proof: First, it is clear that our estimator $\hat{w}(S,\bar{S})$ captures exactly the contribution of cut edges in all Q_i's ($i \in [l]$), since the sketch stores all edges in each Q_i. We thus only need to show the contribution of cut edges in each $P \in \mathcal{P}_i (i \in [l])$, denoted as $\tilde{w}_i(S\cap V_P, \bar{S}\cap V_P)$, can be approximated well by the estimator I_P (Equation (8)).

Let $\gamma_i = 5\cdot 2^{-i} \ge \varepsilon^2$ (since $\tilde{w}(e) \in [\varepsilon^2, 5]$), and then all edges $e \in L_i$ have weight $\tilde{w}(e) \in [\gamma_i, 2\gamma_i]$. Also note that by the stopping condition of the recursive partition (Line 8 of Algorithm 2), each returned $P \in \mathcal{P}_i$ satisfies $\Gamma_P \ge 1/\varepsilon$. Therefore each $P \in \mathcal{P}_i$ is an S1-graph.

Now observe that

$$\tilde{w}_i(S\cap V_P, \bar{S}\cap V_P) \le \tilde{w}(S,\bar{S}) \le (1+3\varepsilon)\, w(S,\bar{S}) \le 5,$$

together with the fact that P is an S1-graph, we know, according to Lemma 4.2, that I_P defined by Equation (8) is an unbiased estimator of $\tilde{w}_i(S\cap V_P, \bar{S}\cap V_P)$ with the variance $\mathbf{Var}\left[I_P\right] \le 44\varepsilon^2(\tilde{w}_i(S\cap V_P, \bar{S}\cap V_P))^2$. It follows that $\hat{w}(S,\bar{S})$ is an unbiased estimator of $\tilde{w}(S,\bar{S})$, since

$$\begin{aligned}\mathbf{E}[\hat{w}(S,\bar{S})] &= \sum_{i\in[l]}\Big(\sum_{P\in\mathcal{P}_i}\mathbf{E}[I_P] + \sum_{e\in Q_i\cap\partial(S,\bar{S})}\tilde{w}(e)\Big) = \sum_{i\in[l]}\Big(\sum_{P\in\mathcal{P}_i}\tilde{w}_i(S\cap V_P,\bar{S}\cap V_P) + \sum_{e\in Q_i\cap\partial(S,\bar{S})}\tilde{w}(e)\Big)\\ &= \tilde{w}(S,\bar{S}). \end{aligned} \tag{11}$$

And the variance

$$\begin{aligned}\mathbf{Var}\left[\hat{w}(S,\bar{S})\right] &= \sum_{i\in[l]}\sum_{P\in\mathcal{P}_i}\mathbf{Var}\left[I_P\right] \le 44\varepsilon^2\sum_{i\in[l]}\sum_{P\in\mathcal{P}_i}(\tilde{w}_i(S\cap V_P,\bar{S}\cap V_P))^2\\ &\le 44\varepsilon^2\Big(\sum_{i\in[l]}\sum_{P\in\mathcal{P}_i}\tilde{w}_i(S\cap V_P,\bar{S}\cap V_P)\Big)^2 = 44\varepsilon^2(\tilde{w}(S,\bar{S}))^2. \end{aligned} \tag{12}$$

Thus by a Chebyshev's inequality, with probability at least $8/9$, we have that $\left|\hat{w}(S,\bar{S}) - \tilde{w}(S,\bar{S}))\right| \le 3\sqrt{44\varepsilon^2(\tilde{w}(S,\bar{S}))^2} \le 21\varepsilon\,\tilde{w}(S,\bar{S})$. □

Algorithm 3: Cut-Basic(G, ε)

```
   Input: $G = (V, E, w)$ with all weights in $[1, n^5]$; a quality control parameter $\varepsilon \in [1/n, 1/30]$
   Output: A $(1 + 27\varepsilon, 2/9)$-cut-sketch $\mathrm{sk}(G)$ of $G$
1  $\mathrm{sk}(G) \leftarrow \emptyset$;
2  Build a 1.2-cut sparsifier $H$ of $G$, and add $H$ into $\mathrm{sk}(G)$;
3  Let $\tilde{C} = \{1.4^i \mid 0 \le i \le \log_{1.4} n^5\}$;
4  foreach $c \in \tilde{C}$ do
5      $D_c \leftarrow \emptyset$;
6      $\{\mathcal{P}, Q\} \leftarrow$ Cut-Preprocessing$(G, c)$, and add $Q$ into $D_c$;
7      foreach $P \in \mathcal{P}$ do
8          $\mathrm{sk}(P) \leftarrow$ Cut-S1$(P, \varepsilon)$, and add $\mathrm{sk}(P)$ in $D_c$;
9      Add $D_c$ into $\mathrm{sk}(G)$;
10 return $sk(G)$;
```

After the preprocessing (Algorithm 2), we only need to run Algorithm 1 on each resulting S1-graph and on special queries (i.e., $w(S, \bar{S}) \le 5$). The overall algorithm for general graph with polynomial weights is described in Algorithm 3.

The following theorem summarizes the results of this section.

Theorem 4.6 *Given a weighted graph $G = (V, E, w)$ on n vertices, where the non-zero weights are in the range $[1, W]$ with $W = n^5$, and $1/n \le \varepsilon \le 1/30$, there exists a cut sketch of size $\tilde{O}(n/\varepsilon)$ bits. Specifically, for every query $S \subset V$, the sketch produces a $1 + O(\varepsilon)$ approximation to $w(S, \bar{S})$, with probability at least $7/9$.*

Proof: Given a query subset $S \subset V$, we use the data structure D_c with $c \in [\tilde{c}/(1.4)^3, \tilde{c}/1.4]$, where $\tilde{c}$ is a 1.2-approximation to the cut weight $w(S, \bar{S})$. Thus we have $w(S, \bar{S}) \in [c, 4c]$. Since we rescale c to 1, we here show the constructed estimator is $1 + O(\varepsilon)$ approximation of the (sclaled) cut weight $w(S, \bar{S})$.

Then, by Lemma 4.4 and 4.5, we have that with probability at least $7/9$, the estimator $\hat{w}(S, \bar{S})$ (using the data structure D_c) is a $1 + O(\varepsilon)$ approximation of $\tilde{w}(S, \bar{S})$, which in turn is a $1 + O(\varepsilon)$ approximation of $w(S, \bar{S})$. More precisely,

$$\left|\hat{w}(S, \bar{S}) - w(S, \bar{S})\right| \le 3\varepsilon\, w(S, \bar{S}) + 21\varepsilon(1 + 3\varepsilon)\, w(S, \bar{S}) \le 27\varepsilon\, w(S, \bar{S}).$$

We next bound the sketch size. The sparsifier H has $\tilde{O}(n)$ edges. By construction, we have $O(\log n)$ possible cut values $\tilde{C}$ and for each one, we have $l = O(\log \frac{1}{\varepsilon}) \le O(\log n)$ edge weight classes. For each weight class L_i, the sketch stores:

- $O(\frac{1}{\varepsilon} n \log n)$ edges in Q_i: each step in the recursive partition contributes $\left|\partial(S', \bar{S}')\right| / |S'| < 1/\varepsilon$ edges per vertex in S', and each vertex appears in the smaller subset S' at most $\log n$ times.
- At most n/ε sampled edges: for each non-isolated vertex we sample $s = 1/\varepsilon$ incident edges.

Summing up, we have $\tilde{O}(n/\varepsilon)$ edges, each requiring $O(\log n)$ bits. Therefore the size of our cut sketch is $\tilde{O}(n/\varepsilon)$ bits.

□

```
Algorithm 4: MST($G$)
  Input: A graph $G = (V, E, w)$
  Output: A maximum-weight spanning tree $T$
1 $E_T \leftarrow \emptyset$, $T \leftarrow (V, E_T)$;
2 $\pi \leftarrow$ an order of edges $e \in E$ in decreasing weight (break ties arbitrarily if any);
3 foreach $e \in E$ in the ordering $\pi$ do
4     Add $e$ into $E_T$ if this will not introduce a cycle in $T$, where edges in $E_T$ are sorted in the
      order of insertion (which is also their ordering according to $\pi$);
5 return $T$;
```

4.1.3 Graphs with General Edge Weights

We now build on the results of Section 4.1.2 for polynomial weights to show the upper bound for general edge weights. That is, assume that there is a sketching algorithm, which we shall call the "basic sketch", for the case where all edge weights are in a polynomial range, say for concreteness $[1, n^5]$ (which by scaling is equivalent to the range $[b, n^5 b]$ for any $b > 0$), which uses space $\tilde{O}(n/\varepsilon)$. We may assume the success probability of this sketch is at least $1 - 1/n^8$, e.g., by using the standard "$O(\log n)$ repetitions and then taking the median", thereby increasing the sketch size by at most $O(\log n)$ factor. As before, we may assume $\varepsilon > 1/n$, as otherwise the theorem is trivial.

Sketching. The sketch has two components: (i) the first component is essentially a maximum-weight spanning tree T computed using Kruskal's algorithm; see Algorithm 4; and (ii) the second component is a set of cut sketches of the graphs reduced according to those tree edges in T, which are constructed via the sketching algorithm introduced in Section 4.1.2; see Algorithm 5, where G_j, G'_j are defined.

Estimation. Given a query subset $S \subset V$, find the smallest $j \in [n-1]$ such that e_j is in the cut $\partial(S, \bar{S})$; such j exists because $\{e_1, \ldots, e_{n-1}\}$ forms a spanning tree (we assume the graph is connected, since otherwise we can sketch each connected component separately). We further show in Lemma 4.7 below that e_j is the heaviest edge in this cut, hence $w_G(S, \bar{S})/w(e_j) \in [1, n^2]$. Now find the largest $k \le j$ for which we sketched and stored in $\mathrm{sk}(G)$; by construction $w(e_k)/w(e_j) \in [1, 2)$. Lemma 4.8 below proves that the cut values in G and in G_k are almost the same.

Next, compute the connected components of the graph $(V, \{e_1, \ldots, e_k\})$, and observe they must be exactly the same as the connected components of G_k. Obviously, the value of the cut $(S, \bar{S})$ in G_k is just the sum, over all connected components P in G_k, of the contribution to the cut from edges inside that component, namely $w_{G_k}(S \cap V_P, \bar{S} \cap V_P)$ with V_P as the vertex set of P. Recall that G'_k has essentially the same cuts as G_k and we can thus estimate each such term $w_{G_k}(S \cap V_P, \bar{S} \cap V_P)$ using the sketch we prepared for G'_k (more precisely, using the sketch of the respective component V' of G'_k, unless $|V'| = 1$ in which case that term is trivially 0). To this end, we need to find out which vertices of G_k were merged together to form G'_k, which can be done using $e_1, \ldots, e_{n-1}$ as follows. Find the largest k^* such that $w(e_{k^*}) \ge n^2 \cdot w(e_k)$, and compute the connected components of the graph $(V, \{e_1, \ldots, e_{k^*}\})$. Lemma 4.9 below proves that these connected components (or more precisely the partition of V they induce) are exactly the subsets of vertices that are merged in G_k to create G'_k. Now that knowing the vertex correspondence between G_k and G'_k, we estimate the cut value $w_{G_k}(S \cap V', \bar{S} \cap V')$ by simply using the estimate for the corresponding cut value in G'_k,

Algorithm 5: Cut-Sketch(G, ε)

```
  Input: A graph $G = (V, E, w)$ with general weights; a quality control parameter
         $\varepsilon \in [1/n, 1/30]$
  Output: A $(1+\varepsilon, \delta)$-cut-sketch sk$(G)$ of $G$
1 sk$(G) \leftarrow \emptyset$;
2 $T \leftarrow$ MST$(G)$, denoted as $T = \{e_1, ..., e_{n-1}\}$, and add $T$ into sk$(G)$;
3 for $j = 1$ to $n-1$ do
4     if there is no earlier iteration $k < j$ with $w(e_k)/w(e_j) < 2$ that is sketched and stored
      then
5         Remove all edges $e \in E$ of weight $w(e) < w(e_j)/n^3$;
6         Change all edges $e \in E$ of weight $w(e) \geq n^2 \cdot w(e_j)$ to have infinite weight, and
          denote the resulting graph as $G_j$;
7         Contract all edges of infinite weight in $G_j$ (keeping parallel edges and removing
          self-loops), and denote the resulting graph as $G'_j$;
8     foreach connected component $P$ of $G'_j$ of size at least 2 do
9         sk$(P) \leftarrow$ Cut-Basic$(P, \varepsilon)$, and add sk$(P)$ into sk$(G)$;
          /* (Note that $\forall e$ in $P$, $w(e) \in [n^{-3} \cdot w(e_j), n^2 \cdot w(e_j)]$.)   */
10 return sk$(G)$;
```

where the latter is obtained using the basic sketch prepared for G'_k. Thus, the estimator is

$$\hat{w}(S, \bar{S}) = \sum_{P \in G'_k} \hat{w}(S \cap V_P, \bar{S} \cap V_P), \tag{13}$$

where $\hat{w}(S \cap V_P, \bar{S} \cap V_P)$ is defined in Equation (10) and can be computed via using the sketch sk(P).

To show the performance of the estimator $\hat{w}(S, \bar{S})$, we first present three lemmas.

Lemma 4.7 *Fix $S \subset V$ and let $e' \in E$ be the first edge, according to the ordering π, that in the cut $\partial(S, \bar{S})$. Then this e' is the first edge in the sequence $e_1, \ldots, e_{n-1}$ that in the cut $\partial(S, \bar{S})$.*

Proof: Let $e' \in E$ be the first edge, according to the ordering π, that in the cut $\partial(S, \bar{S})$. Clearly, e' is the heaviest edge in this cut. Now observe that in the construction of T (i.e., $e_1, \ldots, e_{n-1}$), when e' is considered, T has no edge between S and $\bar{S}$, hence the endpoints of e' lie in different connected components, and e' must be added to T. □

Lemma 4.8 *Consider a query $S \subset V$ and let $k \in [n-1]$ be the value computed in the estimation algorithm. Then the ratio between the value of $w(S, \bar{S})$ in the graph G_k and that in the graph G is in the range $[1 - \frac{1}{n}, 1] \subset [1 - \varepsilon, 1]$, formally*

$$1 - \tfrac{1}{n} \leq \frac{w_{G_k}(S, \bar{S})}{w_G(S, \bar{S})} \leq 1.$$

Proof: The edges in G_k are obtained from the edges of G, by either (1) removing edges e whose weight is $w(e) < w(e_k)/n^3$; or (2) changing edges e with $w(e) \geq n^2 \cdot w(e_k)$ to have infinite weight.

The first case can only decrease any cut value, while the second case can only increase any cut value.

Recall that the estimation process finds j such that $w_G(S,\bar{S})/w(e_j) \in [1,n^2]$, and then finds $k \le j$, which we said always satisfies $w(e_k)/w(e_j) \in [1,2)$. Thus, $w_G(S,\bar{S})/w(e_k) \in (\frac{1}{2},n^2]$. So one direction of the desired inequality follows by observing that edges in G that fall into case (1) have the total weight at most

$$\binom{n}{2} w(e_k)/n^3 \le \frac{2}{n} w(e_k) \le \frac{1}{n} w_G(S,\bar{S}).$$

The other direction follows by observing that edges e that fall into case (2) have (in G) weight $w(e) > n^2 \cdot w(e_k) \ge w_G(S,\bar{S})$, and therefore do not belong to the cut $(S,\bar{S})$. □

Lemma 4.9 *Fix $w^* > 0$, let $E^* = \{e \in E : w(E) \ge w^*\}$, and find the largest $i^* \in [n-1]$ such that $w(e_{i^*}) > w^*$. Then the graphs (V,E^*) and $(V,\{e_1,\ldots,e_{i^*}\})$ have exactly the same connected components (in terms of the partition they induce of V).*

Proof: It is easy to see that executing our construction of T above on the set E^* gives the exact same result as executing it for E but stopping once we reach edges of weight smaller than w^*. The latter results with the edges $e_1,\ldots,e_{i^*}$, while the former is clearly an execution of Kruskal's algorithm, i.e. computes a maximum weight forest in E^*. □

The next two lemmas show the performance of Algorithm 5.

Lemma 4.10 ***(Accuracy Guarantee)*** *With high probability, $\hat{w}(S,\bar{S})$ defined in Equation (13) is $1+O(\varepsilon)$ approximation of $w_G(S,\bar{S})$.*

Proof: Lemma 4.7 and 4.8 show that $(1-\varepsilon)w_G(S,\bar{S}) \le w_{G_k}(S,\bar{S}) \le w_G(S,\bar{S})$. Lemma 4.9 show that $w_{G_k}(S,\bar{S})$ is the same as the cut weight of $(S,\bar{S})$ in G'_k. Since we build the sketch of each connected component of size at least 2 in G'_k, we can obtain a $1+O(\varepsilon)$ approximation, with high probability (say $1-1/n^8$), of the contribution to cut weight of each component in G'_k, i.e., the estimator $\hat{w}(S \cap V_P, \bar{S} \cap V_P)$ defined in Equation (10). Since the number of component is $O(n)$ (because they correspond to disjoint subsets of V), applying the union bound over the events of an error in any of the basic estimates used along the way, we know that the constructed estimator $\hat{w}(S,\bar{S})$ defined in Equation (13) is $1+O(\varepsilon)$ approximation of $w_G(S,\bar{S})$, with high probability. (The union bound is applicable because these basic sketch are queried in a non-adaptive manner, or alternatively, because we make at most one query to every basic sketch that is constructed independently of the others.) □

Lemma 4.11 ***(Size Analysis)*** *The total size of the sketch is at most $\tilde{O}(n/\varepsilon \cdot \log\log W)$, where we assume all non-zero edge weights are in the range $[1,W]$.*

Proof: The first component of the sketch is just a list of $n-1$ edges with their edge weights, hence its size is $O(n\log(\log W/\varepsilon))$ (we can store a $1+\varepsilon/2$ approximation to each weight using space $\log(\log W/\varepsilon)$).

The second component of the sketch has $n-1$ parts, one for each G'_j where $j \in [n-1]$. For some of these j values, we compute and store the basic sketch for every connected components of G'_j that is of size at least 2. Denoting by n_j the number of vertices in G'_j, and by m_j the number of connected components in G'_j, the storage requirement for each G'_j is at most $\tilde{O}(\frac{n_j - m_j}{\varepsilon})$, because each connected component of size $s \geq 2$ requires storage $\tilde{O}(\frac{s}{\varepsilon}) \leq \tilde{O}(\frac{s-1}{\varepsilon})$, and these sizes (the different s values) add up to at most n.

Denote the values of j for which we do store a basic sketch for G'_j by $j_1 < j_2 < \cdots < j_p$, where by construction $w(e_{j_i})/w(e_{j_{i+1}}) \geq 2$. Summing over these values of j, the second component's storage requirement is at most

$$\tilde{O}(\tfrac{1}{\varepsilon} \sum_{i \in [p]} (n_{j_i} - m_{j_i})). \tag{14}$$

To ease notation, let $M = 5 \log_2 n$, and consider the graphs G'_{j_i} and $G'_{j_{i+M}}$ for some $i \in [p-M-1]$. Observe that every edge in G'_{j_i} has weight at least $w(e_{j_i})/n^3 \geq 2^M \cdot w(e_{j_{i+M}})/n^3 = n^2 \cdot w(e_{j_{i+M}})$ (because edges of smaller weight are removed); thus, in $G_{j_{i+M}}$, these same edges have infinite weight, and then to create the reduced form $G_{j_{i+M}}$, these edges are contracted. It follows from this observation that every connected component in G'_{j_i} becomes in $G_{j_{i+M}}$ a single vertex, hence $n_{j_{i+M}} \leq m_{j_i}$ (we do not obtain equality since additional contractions may occur). Using this last inequality, for every $i^* \in [M]$, we can bound the following by a telescopic sum

$$\sum_{i=i^*, i^*+M, i^*+2M, \ldots} (n_{j_i} - m_{j_i}) \leq \sum_{i=i^*, i^*+M, i^*+2M, \ldots} (n_{j_i} - n_{j_{i+M}}) \leq n_{j_{i^*}} \leq n,$$

and therefore

$$\sum_{i \in [p]} (n_{j_i} - m_{j_i})) \leq \sum_{i^* \in [M]} \sum_{i=i^*, i^*+M, i^*+2M, \ldots} (n_{j_i} - m_{j_i}) \leq M \cdot n.$$

Plugging this last inequality into (14), we obtain that the second component's storage requirement is at most $M \cdot \tilde{O}(n/\varepsilon)$, which is still bounded by $\tilde{O}(n/\varepsilon)$. □

Based on Lemma 4.10 and 4.11, we conclude the following theorem.

Theorem 4.12 *Fix an integer n and $\varepsilon \in (1/n, 1/30)$. Then every n-vertex graph $G = (V, E, w)$ with edge weights in the range $[1, W]$ admits a cut sketch of size $\tilde{O}(n\varepsilon^{-1} \cdot \log \log W)$ bits with the "for each" guarantee. Specifically, for every query $S \subset V$ (equivalently, $x \in \{0,1\}^n$), the sketch can produce with high probability a $1 + O(\varepsilon)$ approximation to $w(S, \bar{S})$.*

4.1.4 A Tight Lower Bound

The following theorem shows that our sketch from Theorem 4.12 achieves optimal space up to a poly-logarithmic factor, even for unweighted graphs.

Theorem 4.13 *Fix an integer n and $\varepsilon \in [2/n, 1/2]$. Suppose $sk(\cdot)$ is a sketching algorithm that outputs at most $s = s(n, \varepsilon)$ bits, and* **est** *is an estimation algorithm, such that together for every n-vertex graph G,*

$$\forall S \subset V, \qquad \mathbf{Pr}\Big[\mathbf{est}(S, sk(G)) \in (1 \pm \varepsilon) \cdot |\partial(S, \bar{S})|\Big] \geq 9/10,$$

where $\partial(S, \bar{S}) = \{(u, v) \in E \mid u \in S, v \in \bar{S}\}$. Then $s \geq \Omega(n/\varepsilon)$.

Proof: We will show how to encode a bit-string of length $l = n/(8\varepsilon)$ into a graph, so that, given its sketch $\mathrm{sk}(G)$, one can reconstruct any bit of the string with constant probability. Standard information-theoretical argument would then imply that $s \geq \Omega(l) = \Omega(n/\varepsilon)$.

Given a string $x \in \{0,1\}^l$, we embed it into a bipartite graph G on with $n/2$ vertices on each side, and vertex degrees bounded by $D = 1/(4\varepsilon)$ as follows. Partition the vertices on each side into disjoint blocks of D, and let the i-th block on the left side and on the right side form a (bipartite) graph which we call G_i, for $i = 1, \ldots, n/(2D)$. Then partition the string x in $n/(2D)$ blocks, each block is of length D^2 and describes the adjacency matrix of some bipartite G_i.

We now show that evaluating a bit from the string x corresponds to testing the existence of some edge (u,v) from some graph G_i, which we can do using the $1+\varepsilon$ approximating sketch only. Formally, let $\delta(S)$ be the cut value of the set S, i.e., $|\partial(S,\bar{S})|$, and observe that

$$\delta(\{u\}) + \delta(\{v\}) - \delta(\{u,v\}) = \begin{cases} 2 & \text{if } (u,v) \text{ is an edge in } G; \\ 0 & \text{otherwise.} \end{cases}$$

Since the considered values of $\delta(\cdot)$ are bounded by D, the sketch estimates each such value with additive error at most $\varepsilon D = 1/4$, which is enough to distinguish between the two cases. Furthermore, since we query the sketch only 3 times, the probability of correct reconstruction of the bit is at least 7/10. The lower bound follows. □

4.2 Laplacian Matrices with Spectral Queries

In this section, we construct sketches for a Laplacian matrix with spectral queries. We first design sketches of size $\tilde{O}(n/\varepsilon^{5/3})$, in Section 4.2.1, and then improve it to size $\tilde{O}(n/\varepsilon^{8/5})$ in Section 4.2.2. In each of these, we will start with an algorithm for a class of special graphs, and then extend it to general graphs.

4.2.1 A Basic Sketching Algorithm

In this section, we described a sketch of size $\tilde{O}(n/\varepsilon^{\frac{5}{3}})$ for a Laplacian matrix with spectral queries. Let $\alpha = c_\alpha \varepsilon^{-\frac{5}{3}}$ be a parameter we will use in this section, where $c_\alpha > 0$ is a large enough constant. We will start with an algorithm for a class of special graphs, and then extend it to general graphs.

4.2.1.1 Special Graphs

In this section we consider a class of special graphs, defined as follows.

Definition 4.14 (S2-graph) *We say an undirected weighted graph $G = (V, E, w)$ is an* S2-graph *(reads "simple type-2 graph") if it satisfies the followings.*

1. *All weights $\{w(e) \mid e \in E\}$ are within a factor of 2, i.e. for any $e \in E$, $w(e) \in [\gamma, 2\gamma)$ for some $\gamma > 0$.*
2. *The Cheeger's constant $h_G > \alpha\varepsilon^2 = c_\alpha \varepsilon^{\frac{1}{3}}$.*

Let $\mathcal{S}(G) = \{v \in V \mid \delta_v \leq \gamma\alpha\}$, $\mathcal{L}(G) = \{v \in V \mid \delta_v > \gamma\alpha\}$. For $u \in \mathcal{L}(G)$, let $\delta_u^{\mathcal{L}}(G) = \sum_{v \in \mathcal{L}(G)} w(u,v)$. We will omit "$(G)$" when there is no confusion. The algorithm for sketching

Algorithm 6: Spectral-S2(G, ε)

```
   Input: An S2-Graph $G = (V, E, w)$; a quality control parameter $\varepsilon$
   Output: a $(1+\varepsilon, 0.001)$-spectral-sketch sk$(G)$ of $G$
 1 sk$(G) \leftarrow \emptyset$;
 2 Add $\{\delta_u \mid u \in V\}$ to sk$(G)$;
 3 for $u \in \mathcal{S}$ do
 4     Add all of $u$'s adjacent edges to sk$(G)$;
 5 for $u \in \mathcal{L}$ do
 6     Add $\delta_u^{\mathcal{L}}$ to sk$(G)$;
 7     $E_u \leftarrow \{(u,v) \mid v \in \mathcal{L}\}$;
 8     Sample (with replacement) $\alpha$ edges from $E_u$, where each time the probability of
       sampling $e = (u,v) \in E_u$ is $p_e = w(e)/\delta_u^{\mathcal{L}}$;
 9     Add the sampled $\alpha$ edges to sk$(G)$;
10 return $sk(G)$;
```

S2-graph is described in Algorithm 6. When we say “add an edge to the sketch” we always mean “add the edge together with its weight”.

Let Y_u^v be the random variable denoting the number of times edge (u, v) is sampled at Line 8 in Algorithm 6. It is easy to see that

$$\mathbf{E}[Y_u^v] = \frac{\alpha w(u,v)}{\delta_u^{\mathcal{L}}} \quad \text{and} \quad \mathbf{Var}\left[Y_u^v\right] = \alpha \left(1 - \frac{w(u,v)}{\delta_u^{\mathcal{L}}}\right) \frac{w(u,v)}{\delta_u^{\mathcal{L}}} \le \alpha \frac{w(u,v)}{\delta_u^{\mathcal{L}}}. \tag{15}$$

Given a vector $x \in \mathbb{R}^n$, we use the following expression as an estimator of $x^T L x$:

$$I_G = \sum_{u \in V} \delta_u x_u^2 - \sum_{u \in \mathcal{S}} \sum_{v \in V} x_u x_v w(u,v) - \sum_{u \in \mathcal{L}} \sum_{v \in \mathcal{S}} x_u x_v w(u,v) - \sum_{u \in \mathcal{L}} \frac{\delta_u^{\mathcal{L}}}{\alpha} \sum_{v \in \mathcal{L}} x_u x_v Y_u^v. \tag{16}$$

Lemma 4.15 *Let $G = (V, E, w)$ be an S2-Graph and $L = L(G)$ be the (unnormalized) Laplacian of G, then I_G (defined in Equation (16)) is an unbiased estimator of $x^T L x$. Furthermore, it gives a $(1+\varepsilon, 0.001)$-approximation to $x^T L x$.*

Proof: Since $\mathbf{E}[Y_u^v] = \frac{\alpha w(u,v)}{\delta_u^{\mathcal{L}}}$ (by (15)), it is straightforward to show that

$$\mathbf{E}[I_G] = \sum_{u \in V} \delta_u x_u^2 - \sum_{u \in \mathcal{S}} \sum_{v \in V} x_u x_v w(u,v) - \sum_{u \in \mathcal{L}} \sum_{v \in V} x_u x_v w(u,v) = \sum_{(u,v) \in E} (x_u - x_v)^2 w(u,v) = x^T L x.$$

Now let us compute the variance of I_G. Note that if $\mathbf{Var}\left[I_G\right] = O\left(\varepsilon^2 (x^T L x)^2\right)$, then by taking constant c_α in $\alpha = c_\alpha \varepsilon^{-\frac{5}{3}}$ large enough, a Chebyshev's inequality immediately yields the lemma.

The variance of I_G

$$\begin{aligned}
\mathbf{Var}\left[I_G\right] &= \mathbf{Var}\left[\sum_{u\in\mathcal{L}}\frac{\delta_u^{\mathcal{L}}}{\alpha}\sum_{v\in\mathcal{L}}x_u x_v Y_u^v\right] \\
&= \sum_{u\in\mathcal{L}}\frac{(\delta_u^{\mathcal{L}})^2}{\alpha^2}\sum_{v\in\mathcal{L}}x_u^2 x_v^2\mathbf{Var}\left[Y_u^v\right] \\
&\le \sum_{u\in\mathcal{L}}\frac{(\delta_u^{\mathcal{L}})^2}{\alpha^2}x_u^2\sum_{v\in\mathcal{L}}x_v^2\frac{\alpha w(u,v)}{\delta_u^{\mathcal{L}}} \qquad \text{(by (15))} \\
&= \frac{1}{\alpha}\sum_{u\in\mathcal{L}}\delta_u^{\mathcal{L}}x_u^2\sum_{v\in\mathcal{L}}x_v^2 w(u,v) \\
&\le \frac{1}{\alpha}\sum_{u\in\mathcal{L}}\delta_u^{\mathcal{L}}x_u^2\sum_{v\in\mathcal{L}}x_v^2\frac{2\delta_v}{\alpha} \qquad \left(w(u,v)\le 2\gamma\le\frac{2\delta_v}{\alpha}\text{ by def. of S2-graph and def. of }\mathcal{L}\right) \\
&\le \frac{2}{\alpha^2}\sum_{u\in V}\delta_u x_u^2\sum_{v\in V}\delta_v x_v^2 \qquad (\delta_u^{\mathcal{L}}\le\delta_u\text{ by definitions}) \\
&= \frac{2}{\alpha^2}\left\|D^{1/2}x\right\|_2^4, \qquad (17)
\end{aligned}$$

where $D = \mathrm{diag}(\delta_1, \delta_2, \ldots, \delta_n)$. The normalized Laplacian of G can be written as $\tilde{L} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$. Define $\hat{x} = D^{1/2}x$, we have

$$\|\hat{x}\|_2^2 = \hat{x}^T\hat{x} \le \frac{1}{\lambda_1(\tilde{L})}\hat{x}^T\tilde{L}\hat{x} \overset{by\ (1)}{\le} \frac{2}{h_G^2}(x^TLx) \overset{\text{property of S2-graph}}{<} \frac{2}{\alpha^2\varepsilon^4}(x^TLx),$$

which together with (17) gives $\mathbf{Var}\left[I_G\right] < \frac{8}{\alpha^6\varepsilon^8}(x^TLx)^2 = O\left(\varepsilon^2(x^TLx)^2\right)$. □

We summarize our result for the S2-Graph G in the following theorem.

Theorem 4.16 *There is a sketching algorithm which given an S2-graph, outputs a $(1+\varepsilon, 0.001)$-spectral-sketch of size $\tilde{O}(n/\varepsilon^{\frac{5}{3}})$.*

4.2.1.2 General Graphs

Now let us extend our result to general positively weighted simple graphs $G = (V, E, w)$. We now require $w_{\max}/w_{\min} = \mathrm{poly}(n)$.

The following lemma will be used in the analysis of our sketching algorithms.

Lemma 4.17 *Given an undirected positively weighted graph $G = (V, E, w)$ with $w_{\max}/w_{\min} =$ poly(n), there is an algorithm that takes G as the input, and output a graph $\tilde{G} = (V, \tilde{E}, \tilde{w})$ such that*

1. *$\tilde{G}$ is a $(1+\varepsilon, 0.001)$-spectral-sketch of G of size $\tilde{O}(n/\varepsilon^2)$ bits.*

2. *$\tilde{w}_{\max}/\tilde{w}_{\min}$ is bounded by poly(n).*

Algorithm 7: Spectral-Preprocessing(G, h)

```
  Input: A graph $G = (V, E, w)$ such that for any $e \in E$, $w(e) \in [\gamma, 2\gamma)$; a parameter $h > 0$
  Output: A set $\mathcal{P}$ of edge disjoint components of $G$ such that for each $P \in \mathcal{P}$, $h_P > h$; and
          a graph $Q$ induced by the rest of the edges in $G$.
1 $\mathcal{P} \leftarrow \{G\}$, $Q \leftarrow \emptyset$;
2 while $\exists\ P \in \mathcal{P}$ such that Cheeger's constant $h_P \leq h$ do
3     Find an arbitrary cut $(S, \overline{S})$ in $P$, such that $\Phi(S) \leq h$;
4     Replace $P$ with its two subgraphs $P(S)$ and $P(\overline{S})$ in $\mathcal{P}$;
5     Add all edges in the cut $(S, \overline{S})$ into $Q$;
6 return $(\mathcal{P}, Q)$;
```

Proof: We first run the spectral sparsification algorithm by Batson, Spielman and Srivastava [BSS14], which produces $(1+\varepsilon)$-spectral sparsifier H of G with size of $\tilde{O}(n/\varepsilon^2)$ bits with probability 0.999.

Assume (by rescaling) that the edge weights in G are between 1 and n^C for a constant $C > 0$. Since H is also a cut sparsifier, all weights in H must be at most $2n^C$, as otherwise H would not preserve a specific cut up to a factor of 2. Let $\tilde{G}$ be formed from H by removing all edge weights smaller than $1/n^D$ for a sufficiently large constant $D > 0$. Let x be a unit vector orthogonal to 1^n, and assume the underlying graph with Laplacian $L(\tilde{G})$ is connected (otherwise we could have first split it into connected components). Then $x^T L(\tilde{G})x \geq x^T L(H)x - O(n^2/n^D)$. Since $x^T L(H)x \geq (1-\varepsilon)x^T L(G)x \geq (1-\varepsilon)\lambda_1(L(G))$, and $\lambda_1(L(G)) \geq 1/n^2$ by [GY03], it follows that $x^T L(\tilde{G})x \geq (1-\varepsilon)x^T L(H)x$, assuming $D > 0$ is a sufficiently large constant. Since also $x^T L(\tilde{G})x \leq x^T L(H)x$, it follows that $\tilde{G}$ is a $(1+O(\varepsilon))$-spectral sparsifier of G with edge weights that are between $1/n^D$ and $2n^C$. $\square$

The following observation is due to the linearity of Laplacian.

Observation 4.18 *Given any simple graph $G = (V, E, w)$, let L be its Laplacian. Let $E_1, E_2, \ldots, E_k$ be a disjoint partition of E, and let $G_i = (V, E_i, w)$. Let L_i be the Laplacian of G_i. We have $x^T Lx = \sum_{i=1}^k x^T L_i x$ for any $x \in \mathbb{R}^n$.*

Our high-level idea is to reduce general graphs to S2-graphs. Based on Observation 4.18, we can first partition the edge set E into $E_1, \ldots, E_k$ ($k = \Theta(\log n)$) such that for any $e \in E_i$ we have $w(e) \in [2^{i-1}w_{\min}, 2^i w_{\min})$, and then sketch each subgraph G_i separately. Finally, at the time of a query, we simply add all estimators I_{G_i} together. Thus it suffices to focus on a graph with all weight $w(e) \in [\gamma, 2\gamma)$ for some $\gamma > 0$.

We next partition each subgraph G_i further so that each component P satisfies $h_P \geq c_\alpha \varepsilon^{\frac{1}{3}}$. Once this property is established, we can use Algorithm 6 to sketch each P separately. We describe this preprocessing step in Algorithm 7.

The Q returned by Algorithm 7 is a set of cut edges we will literally keep. The following lemma bounds the size of Q. The proof is folklore, and we include it for completeness.

Lemma 4.19 *For any positively weighted graph $G = (V, E, w)$ such that for any $e \in E$, $w(e) \in [\gamma, 2\gamma)$ for some $\gamma > 0$, the number of edges of Q returned by Algorithm 7 Spectral-Preprocessing(G, h) is bounded by $O(hm \log m)$.*

Algorithm 8: Spectral-Basic(G, ε)

```
    Input: $G = (V, E, w)$ with all weights in $[w_{\min}, w_{\max}]$; a quality control parameter $\varepsilon$
    Output: A $(1+\varepsilon, 0.001)$-spectral-sketch $\mathrm{sk}(G)$ of $G$
1   Edge-disjointly partition $G$ into $\mathcal{H} = \{H_1, \dots, H_k\}$ s.t. all edges in $H_i$ have weights in
    $[2^{i-1} w_{\min}, 2^i w_{\min})$;
2   $\mathrm{sk}(G) \leftarrow \emptyset$;
3   foreach $H \in \mathcal{H}$ do
4       $(\mathcal{P}, Q) \leftarrow$ Spectral-Preprocessing$(H, \alpha\varepsilon^2)$;
5       Add $Q$ into $\mathrm{sk}(G)$;
6       for $P \in \mathcal{P}$ do
7           Add Spectral-S2$(P, \varepsilon)$ into $\mathrm{sk}(G)$;
8   return $sk(G)$;
```

Proof: In Algorithm 7, we recursively split the graph $G = (V, E, w)$ into connected components until for every component $P = (V_P, E_P)$, its Cheeger's constant $h_P = \inf_{S \subset V_P} \Phi_C(S) > h$. Consider a single splitting step: We find a cut $(S, \bar{S})$ with $\mathrm{vol}_P(S) \le \mathrm{vol}_P(\bar{S})$ in P such that $\Phi_P(S) = \frac{w_P(S,\bar{S})}{\mathrm{vol}_P(S)} \le h$. We can think in this step each edge in $\mathrm{vol}_P(S)$ contributes at most h edges to Q on average, while edges in $\mathrm{vol}_P(\bar{S})$ contribute nothing to Q. We call S the *Smaller-Subset*.

By the definition of volume, we have $\mathrm{vol}_P(V_P) = \mathrm{vol}_P(S \cup \overline{S}) = \mathrm{vol}_P(S) + \mathrm{vol}_P(\overline{S}) \ge 2\mathrm{vol}_P(S)$, and $\mathrm{vol}_G(V) = 2m$. Thus in the whole recursion process, each edge will appear at most $O(\log m)$ times in Smaller-Subsets, hence will contribute at most $O(h \log m)$ edges to Q. Therefore the number of edges of Q is bounded by $O(hm \log m)$ words. $\square$

Now we show the main algorithm for general graphs and analyze its performance. The algorithm is described in Algorithm 8.

The following lemma summarize the functionality of Algorithm 8.

Lemma 4.20 *Given a graph $G = (V, E, w)$, let sk$(G) \leftarrow$ Spectral-Basic(G, ε), then for any given $x \in \mathbb{R}^n$, sk(G) can be used to construct an unbiased estimator I_G which gives a* $(1+\varepsilon, 0.001)$-*approximation to $x^T L(G) x$. The sketch sk(G) uses $\tilde{O}(\varepsilon^{\frac{1}{3}} m + n/\varepsilon^{\frac{5}{3}})$ bits.*

Proof: In Algorithm 8, G is partitioned into a set of edge disjoint components $\mathcal{P} = \{P_1, \dots, P_t\}$, and we build a sketch $\mathrm{sk}(P_i)$ for each $P_i \in \mathcal{P}$ from which we can construct an unbiased estimator I_{P_i} for $x^T L(P_i) x$ with variance bounded by $O(\varepsilon^2 (x^T L(P_i) x)^2)$ according to Lemma 4.15. Moreover, we have stored all edges between these components; let Q be the induced subgraph of these edges. Our estimator to $x^T L(G) x$ is

$$I_G = \sum_{i=1}^{t} I_{P_i} + x^T L(Q) x, \tag{18}$$

where I_{P_i} defined in Equation (16) is an unbiased estimator of $x^T L(P_i) x$. By the linearity of

Laplacian (Observation 4.18), I_G is an unbiased estimator of $x^T L(G)x$. Now consider its variance:

$$\begin{aligned}
\mathbf{Var}\left[I_G\right] &= \mathbf{Var}\left[\sum_{1\le i\le t} I_{P_i} + x^T L(Q)x\right] \\
&\le O(\varepsilon^2) \sum_{1\le i\le t} \left(x^T L(P_i)x\right)^2 \qquad \text{(due to the independence of } P_i\text{'s)} \\
&\le O(\varepsilon^2) \left(\sum_{1\le i\le t} x^T L(P_i)x + x^T L(Q)x\right)^2 \qquad (L(P_i) \text{ and } L(Q) \text{ are positive semidefinite}) \\
&= O(\varepsilon^2)\left(x^T L(G)x\right)^2.
\end{aligned}$$

The correctness follows from a Chebyshev's inequality.

Now we bound the size of the sketch $\mathrm{sk}(G)$. Consider a particular H at Line 3 of Algorithm 8. For each $P = (V_P, E_P) \in \mathcal{P}$, the size of $\mathrm{sk}(P)$ by running *Spectral-S2*(P, ε) at Line 7 is bounded by $\tilde{O}(|V_P|/\varepsilon^{\frac{5}{3}})$ bits (Theorem 4.16); and for the remaining subgraph $Q = (V_Q, E_Q)$, $\mathrm{sk}(Q)$ is bounded by $\tilde{O}(\varepsilon^{\frac{1}{3}}|E_Q|)$ bits (Lemma 4.19). Thus

$$\begin{aligned}
\mathrm{size}(\mathrm{sk}(P_i)) &= \mathrm{size}(\mathrm{sk}(Q)) + \sum_{P\in\mathcal{P}} \mathrm{size}(\mathrm{sk}(P)) \\
&\le \tilde{O}\left(\varepsilon^{\frac{1}{3}}|E_Q| + \sum_{P\in\mathcal{P}} |V_P|/\varepsilon^{\frac{5}{3}}\right) \\
&\le \tilde{O}\left(\varepsilon^{\frac{1}{3}}m + n/\varepsilon^{\frac{5}{3}}\right) \text{ bits.} \qquad (\{P\in\mathcal{P}\} \text{ are vertex-disjoint})
\end{aligned}$$

Since there are $k = \Theta(\log n)$ of H_i's in $\mathcal{H}$, the size of $\mathrm{sk}(G)$ is bounded by $\tilde{O}\left(n/\varepsilon^{\frac{5}{3}} + \varepsilon^{\frac{1}{3}}m\right)\cdot \log n = \tilde{O}\left(n/\varepsilon^{\frac{5}{3}} + \varepsilon^{\frac{1}{3}}m\right)$ bits. □

We conclude this section with the following theorem.

Theorem 4.21 *There is a sketching algorithm which given an undirected positively weighted graph $G = (V, E, w)$ with $w_{\max}/w_{\min} = poly(n)$, outputs a $(1+\varepsilon, 0.01)$-spectral-sketch of size $\tilde{O}(n/\varepsilon^{\frac{5}{3}})$.*

Proof: The algorithm is as follows: we first run the spectral sparsification algorithm in [BSS14], obtaining a graph $\tilde{G} = (V, \tilde{E}, \tilde{w})$. By Lemma 4.17 we have $|\tilde{E}| = \tilde{O}(n/\varepsilon^2)$ and $\tilde{w}_{\max}/\tilde{w}_{\min} = \mathrm{poly}(n)$. We then run Algorithm 8, getting a $(1+\varepsilon, 0.01)$-spectral-sketch of size $\tilde{O}\left(n/\varepsilon^{\frac{5}{3}} + \varepsilon^{\frac{1}{3}}|\tilde{E}|\right) = \tilde{O}(n/\varepsilon^{\frac{5}{3}})$. □

4.2.2 An Improved Sketching Algorithm

In this section, we further reduce the space complexity of the sketch to $\tilde{O}(n/\varepsilon^{\frac{8}{5}})$. At a high level, such an improvement is achieved by partitioning the graph into more subgroups (compared with a hierarchical partition on weights, and $\mathcal{S}(G)$ and $\mathcal{L}(G)$ for each weight class in the basic approach), in each of which vertices have similar unweighted degrees *and* weighted degrees. An estimator based on a set of sampled edges from such groups will have smaller variance. This finer partition, however, will introduce a number of technical subtleties, as we will describe below.

We set the constant $\beta = c_\beta \varepsilon^{-\frac{8}{5}}$ throughout this section where c_β is a constant.

4.2.2.1 Special Graphs

We first consider a class of simple graphs.

Definition 4.22 (S3-graph) *We say an undirected weighted graph $G = (V, E, w)$ is an* S3-graph *(reads "simple type-3 graph") if we can assign directions to its edges in a certain way, getting a directed graph $\vec{G} = (V, \vec{E}, w)$ satisfying the following.*

1. *All weights $\{w(e) \mid e \in \vec{E}\}$ are within a factor of 2, i.e., for any $e \in \vec{E}$, $w(e) \in [\gamma, 2\gamma)$ for some constant $\gamma > 0$.*
2. *For each $u \in V$, $d_u^{\mathtt{out}}(\vec{G}) \in [2^\kappa \beta, 2^{\kappa+1}\beta)$, where $2^\kappa \beta \le 1/\varepsilon^2$.*

We call $\vec{G}$ the buddy *of G. Note that $\vec{G}$ is not necessarily unique, and we just need to consider an arbitrary but fixed one. In this section we will assume that we can obtain such a buddy directed graph $\vec{G}$ "for free", and will not specify the concrete algorithm. Later when we deal with general graphs we will discuss how to find such a direction scheme.*

We still make use of Algorithm 7 to partition the graph G into components such that the Cheeger's constant of each component is larger than $h = \beta\varepsilon^2$. One issue here is that, after storing and removing those cut edges (denoted by Q in Algorithm 7), the second property of S3-graph may not hold, since the out-degree of some vertices in G's buddy graph $\vec{G}$ will be reduced. Fortunately, the following lemma shows that the number of vertices whose degree will be reduced more than half is small, and we can thus afford to store all their out-going edges. The out-degree of the remaining vertices is within a factor of 4, thus we can still effectively bound the variance of our estimator.

Lemma 4.23 *If we run Algorithm 7 on an S3-graph $G = (V, E, w)$ with $h = 2^{-\kappa}$, then*

1. *At most $\tilde{O}(\beta n)$ cut edges (i.e., Q) will be removed from G.*
2. *There are at most $\tilde{O}(2^{1-\kappa} n)$ vertices in G's buddy graph $\vec{G}$ which will reduce their out-degrees by more than a half after the removal of Q.*

Proof: Since $m = O(2^\kappa \beta n)$, $h = 2^{-\kappa}$ and $2^\kappa \beta = \tilde{O}(\frac{1}{\varepsilon^2})$, Lemma 4.19 directly gives the first part. For the second part, note that for each vertex u we have $\delta_u^{\mathtt{out}}(\vec{G}) \ge 2^\kappa \beta$, thus we need to remove at least $2^{\kappa-1}\beta$ edges to reduce $\delta_u^{\mathtt{out}}(\vec{G})$ to $2^{\kappa-1}\beta$. Therefore the number of such vertices is at most $\tilde{O}(\beta n / 2^{\kappa-1}\beta) = \tilde{O}(2^{1-\kappa} n)$. □

For each component $P = (V_P, E_P)$ after running Algorithm 7, let $\vec{P} = (V_P, \vec{E}_P)$ be its buddy directed graph. Slightly abusing the notation, define $\mathcal{S}(\vec{P}) = \{(u, v) \in \vec{E}_P \mid d_u^{\mathtt{out}}(\vec{P}) < 2^{\kappa-1}\beta\}$, and $\mathcal{L}(\vec{P}) = \{(u, v) \in \vec{E}_P \mid d_u^{\mathtt{out}}(\vec{P}) \ge 2^{\kappa-1}\beta\}$. We will again omit "$(\vec{P})$" or "$(P)$" when there is no confusion.

The sketch for an S3-graph G is constructed using Algorithm 9.

It is easy to see that the size of $\mathrm{sk}(G)$ is bounded $\tilde{O}(2^{1-\kappa}|V_P|) \cdot 2^{\kappa-1}\beta + \tilde{O}(\beta n) + \tilde{O}(\beta n) = \tilde{O}(\beta n)$, where the first term in LHS is due to the definition of $\mathcal{S}(\vec{P})$ and the second property of Lemma 4.23.

Algorithm 9: Spectral-S3(G, ε)

```
   Input: An S3-graph $G$ and a parameter $\varepsilon$
   Output: A $(1+\varepsilon, 0.01)$-spectral-sketch of $G$
1  $\{\mathcal{P}, Q\} \leftarrow$ Spectral-Preprocessing$(G, \beta\varepsilon^2)$;
2  foreach $P = (V_P, E_P, w) \in \mathcal{P}$ do
3      Let $\vec{P} = (V_P, \vec{E}_P, w)$ be its buddy directed graph;
4      foreach $u \in V_P$ do
5          Add $\delta_u^{\mathtt{in}}(\vec{P})$ and $\delta_u(P)$ to sk$(G)$;
6      Add $\mathcal{S}(\vec{P})$ to sk$(G)$;
7      foreach $u \in V_P$ do
8          Sample $\beta = c_\beta \varepsilon^{-\frac{8}{5}}$ edges with replacement from $\{(v,u) \in \mathcal{L}\}$, where the probability
           that $(v,u)$ is sampled is $w(v,u)/\delta_u^{\mathtt{in}}(\vec{P})$. Add sampled edges to sk$(G)$;
9      Add $Q$ to sk$(G)$;
10 return sk$(G)$
```

Let Y_u^v be the random variable denoting the number of times (directed) edge (v,u) is sampled when we process the vertex u. Clearly, $\mathbf{E}[Y_u^v] = \frac{\beta w(v,u)}{\delta_u^{\mathtt{in}}(\vec{P})}$ and $\mathbf{Var}[Y_u^v] \le \frac{\beta w(v,u)}{\delta_u^{\mathtt{in}}(\vec{P})}$. For a given $x \in \mathbb{R}^{|V_P|}$, we construct the following estimator for each component P using sk(G).

$$I_P = \sum_{u \in V_P} x_u^2 \delta_u(P) - 2 \sum_{(u,v)\in\mathcal{S}} x_u x_v w(u,v) - 2 \sum_{u\in V_P} \frac{\delta_u^{\mathtt{in}}(\vec{P})}{\beta} \sum_{(v,u)\in\mathcal{L}} x_u x_v Y_u^v \tag{19}$$

Similar to the analysis in Section 4.2.1.1, it is easy to show that I_P is an unbiased estimator of $x^T L(P) x$ by noticing

$$\begin{aligned} x^T L(P) x &= \sum_{(u,v)\in\vec{E}_P} (x_u - x_v)^2 w(u,v) \\ &= \sum_{u\in V_P} x_u^2 \delta_u(P) - 2\sum_{(u,v)\in\mathcal{S}} x_u x_v w(u,v) - 2 \sum_{(u,v)\in\mathcal{L}} x_u x_v w(u,v). \end{aligned}$$

We next bound the variance

$$\begin{aligned}
\mathbf{Var}\left[I_P\right] &= \mathbf{Var}\left[2\sum_{u\in V_P}\frac{\delta_u^{\mathtt{in}}(\vec{P})}{\beta}\sum_{(v,u)\in\mathcal{L}} x_u x_v Y_u^v\right] \\
&= 4\sum_{u\in V_P}\frac{\left(\delta_u^{\mathtt{in}}(\vec{P})\right)^2}{\beta^2}\sum_{(v,u)\in\mathcal{L}} x_u^2 x_v^2 \mathbf{Var}\left[Y_u^v\right] \\
&\le 4\sum_{u\in V_P}\frac{\left(\delta_u^{\mathtt{in}}(\vec{P})\right)^2}{\beta^2}\sum_{(v,u)\in\mathcal{L}} x_u^2 x_v^2 \frac{\beta w(v,u)}{\delta_u^{\mathtt{in}}(\vec{P})} & \left(\mathbf{Var}\left[Y_u^v\right]\le\frac{\beta w(v,u)}{\delta_u^{\mathtt{in}}(\vec{P})}\right) \\
&= 4\sum_{u\in V_P} x_u^2\frac{\delta_u^{\mathtt{in}}(\vec{P})}{\beta}\sum_{(v,u)\in\mathcal{L}} x_v^2 w(v,u) \\
&\le 4\sum_{u\in V_P} x_u^2\frac{\delta_u^{\mathtt{in}}(\vec{P})}{\beta}\sum_{(v,u)\in\mathcal{L}} x_v^2\cdot 2\gamma & (w(v,u)\in[\gamma,2\gamma)) \\
&\le \frac{4}{\beta}\sum_{u\in V_P} x_u^2\delta_u^{\mathtt{in}}(\vec{P})\sum_{(v,u)\in\mathcal{L}} x_v^2\frac{2\delta_v^{\mathtt{out}}(\vec{P})}{2^{\kappa-1}\beta} & \left(d_v^{\mathtt{out}}(\vec{P})\ge 2^{\kappa-1}\beta \text{ and } \delta_v^{\mathtt{out}}(\vec{P})\ge\gamma\cdot d_v^{\mathtt{out}}(\vec{P})\right) \\
&\le \frac{16}{2^\kappa\beta^2}\sum_{u\in V_P}\delta_u(P)x_u^2\sum_{v\in V_P}\delta_v(P)x_v^2 & \left(\delta_u^{\mathtt{in}}(\vec{P})\le\delta_u(P) \text{ and } \delta_v^{\mathtt{out}}(\vec{P})\le\delta_v(P)\right) \\
&= \frac{16}{2^\kappa\beta^2}\|\hat{x}\|_2^4,
\end{aligned}$$

where $\|\hat{x}\|_2^4 = \|D^{1/2}x\|_2^4 = \sum_{u\in V_P}\delta_u(P)x_u^2\sum_{v\in V_P}\delta_v(P)x_v^2$.

Similar to before we have

$$\|\hat{x}\|_2^2 = \hat{x}^T\hat{x} \le \frac{1}{\lambda_1(\tilde{L})}\hat{x}^T\tilde{L}\hat{x} \overset{by\ (1)}{\le} \frac{2}{h_G^2}(x^TLx) \overset{h_G>2^{-\kappa}\text{ by Algorithm 7}}{<} 2\cdot 2^{2\kappa}\cdot(x^TLx),$$

Recall that in an S3-graph, $2^\kappa \le 1/(\beta\varepsilon^2)$, hence

$$\frac{\mathbf{Var}\left[I_P\right]}{\left(x^TL(P)x\right)^2} = O\left(\frac{2^{3\kappa}}{\beta^2}\right) = O\left(1/(\beta^5\varepsilon^6)\right) = O(\varepsilon^2).$$

Setting constant c_β large enough in $\beta = c_\beta\varepsilon^2$, by a Chebyshev's inequality, I_P is a $(1+\varepsilon, 0.01)$-approximation to $x^TL(P)x$.

Theorem 4.24 *There is a sketching algorithm which given an S3-graph, outputs a $(1+\varepsilon, 0.01)$-spectral-sketch of size $\tilde{O}(n/\varepsilon^{\frac{8}{5}})$.*

4.2.2.2 General Graphs

To deal with general graphs $G = (V, E, w)$ for which the only requirement is $w_{\max}/w_{\min} \le \mathrm{poly}(n)$, we try to "partition" it to polylog n subgraphs, each of which is an S3-graph. We note that the partition we used here is *not* a simple vertex-partition or edge-partition, as will be evident shortly.

Algorithm 10: Assign-Direction(G, t)

```
  Input: A graph $G = (V, E)$ and a parameter $t$
  Output: $\vec{G} = (V, \vec{E})$, a directed graph by assigning each edge in $G$ a direction
1 Arbitrarily assign a direction to each edge in $E$, getting $\vec{E}$;
2 while $\exists (u,v) \in \vec{E}$ s.t. $d_u^{\texttt{out}}(\vec{G}) \geq t$ and $d_v^{\texttt{out}}(\vec{G}) < t-1$ do
3     Change the direction of $(u,v)$;
4 return $\vec{G} = (V, \vec{E})$;
```

Our first step is to assign each edge in E a direction so that in the later partition step we can partition G to S3-graphs and simultaneously get their buddy directed graphs. The algorithm is described in Algorithm 10.

Lemma 4.25 *Given $G = (V, E)$ and $s > 1$ as input, Assign-Direction(G, s) (Algorithm 10) will finally stop and return $\vec{G} = (V, \vec{E})$ with the property that for each $(u,v) \in \vec{E}$, $d_u^{\texttt{out}}(\vec{G}) < s$ or $d_v^{\texttt{out}}(\vec{G}) \geq s-1$.*

Proof: The second part is trivial according to Algorithm 10. Now we show that Assign-Direction(G, s) will finally stop. Let $S = \{(u,v) \in \vec{E} \mid d_u^{\texttt{out}}(\vec{G}) \geq s$ and $d_v^{\texttt{out}}(\vec{G}) < s-1\}$, and $\Delta(S) = \sum_{(u,v) \in S} (d_u^{\texttt{out}}(\vec{G}) - d_v^{\texttt{out}}(\vec{G}))$. The algorithm stops if and only if $\Delta(S) = 0$. It is easy to see that $\Delta(S)$ is finite for arbitrary $\vec{G}$, thus the algorithm will stop if we can show that $\Delta(S)$ will decrease by *at least* 2 each time we execute Line 3. To this end, we only need to show that each execution of Line 3 will not add any new edge to S.

Consider executing Line 3 on edge (u,v). For $(u,w) \notin S$, since $(u,v) \in S$, we have $d_w^{\texttt{out}}(\vec{G}) \geq s-1$. Clearly, after executing Line 3, (u,w) will not be added to S because $d_w^{\texttt{out}}(\vec{G}) \geq s-1$ still holds. For $(w,v) \notin S$, since $(u,v) \in S$, we have $\delta_w^{\texttt{out}}(\vec{G}) < s$. After executing Line 3, $d_w^{\texttt{out}}(\vec{G}) < s$ still holds, hence (w,v) will not be added into S. $\square$

For a set of directed edges $\vec{E}$, let $d_u^{\texttt{out}}(\vec{E}) \leftarrow |\{v \mid (u,v) \in \vec{E}\}|$. Our partition step is described in Algorithm 11. We first run the spectral sparsification algorithm [BSS14], and then assign directions to each edge. Next, we partition the edges based on their weights, and then partition the directed graph based on the unweighted out-degree of each vertex. Finally, we recursively perform all the above steps on a subgraph induced by a set of edges which have large weights. Notice that the purpose of introducing directions on edges is to assist the edge partition.

For the analysis, we first show that after each recursion in Algorithm 11, the number of vertices of the graph induced by the remaining edges will decrease by at least a constant fraction. In this way we can bound the number of recursion steps by $O(\log n)$.

Lemma 4.26 *Given a graph $G = (V, E)$ with $m \leq sn$ $(s > 1)$, let $\vec{G} \leftarrow$ Assign-Direction$(G, 2s)$. If we remove all (u,v) with $d_u^{\texttt{out}}(\vec{E}) < 2s$ from $\vec{E}$ and get a subset $\vec{E}_r \subset \vec{E}$, then we have $|V_r(\vec{E}_r)| \leq n/(2 - 1/s)$*

Before proving this lemma, note that in Algorithm 11, $m \leq sn$ is guaranteed by Line 3, and "remove all (u,v) with $d_u^{\texttt{out}}(\vec{E}) < 2s$" is done by Line 12.

Algorithm 11: Partition(G)

```
    Input: A graph $G = (V, E, w)$ and a parameter $\varepsilon$
    Output: A set of graph components $\mathcal{P}$
 1  if $n < 3$ then
 2      return $\{G\}$;
 3  Run [BSS14] on $G$ with parameter $\varepsilon$, get a spectral sparsifier $G' = (V, E', w')$ with
    $|E'| = \eta \cdot \frac{n}{\varepsilon^2}$ where $\eta = \tilde{O}(1)$;
 4  $\tilde{\varepsilon} = \varepsilon/\sqrt{\eta}$, $s \leftarrow 1/\tilde{\varepsilon}^2$;
 5  $\vec{G'} = (V, \vec{E'}, w') \leftarrow$ Assign-Direction$(G', 2s)$ ;
 6  Partition $\vec{E'}$ into $\vec{E}'_j$'s such that for each $\vec{E}'_j$, all $e \in \vec{E}'_j$ have $w(e) \in [2^j, 2^{j+1})$;
 7  $\mathcal{P} \leftarrow \emptyset$;
 8  foreach $\vec{E}'_j$ do
        /* Recall $\beta = c_\beta \varepsilon^{-\frac{8}{5}}$ for a large enough constant $c_\beta$ */
 9      Let $\vec{E}_{-\infty} \leftarrow \left\{(u, v) \in \vec{E}'_j \mid d_u^{\texttt{out}}(\vec{E}'_j) < \beta\right\}$;
10      Let $\vec{E}_i \leftarrow \left\{(u, v) \in \vec{E}'_j \mid d_u^{\texttt{out}}(\vec{E}'_j) \in [2^i\beta, 2^{i+1}\beta)\right\}$ for all $i \geq 0$ s.t. $2^i\beta \leq s = 1/\tilde{\varepsilon}^2$;
11      Add $G(\vec{E}_{-\infty})$ and $G(\vec{E}_i)$ for all $0 \leq i \leq \log(1/(\tilde{\varepsilon}^2\beta))$ into $\mathcal{P}$;
12      Remove $\vec{E}_{-\infty}, \vec{E}_i$ from $\vec{E}'$;
    /* Recursively apply on the remaining edges $E'$ (remove directions on edges)
       */
13  return $\mathcal{P} \cup$ Partition $(G(E'))$;
```

Proof: By Lemma 4.25, for each $(u, v) \in \vec{E}$, we have $d_u^{\texttt{out}}(\vec{E}) < 2s$ or $d_v^{\texttt{out}}(\vec{E}) \geq 2s - 1$. If we remove all (u, v) with $d_u^{\texttt{out}}(\vec{E}) < 2s$, then for each $(u, v) \in \vec{E}_r$, we have $d_u^{\texttt{out}}(\vec{E}_r) \geq 2s$ and $d_v^{\texttt{out}}(\vec{E}_r) \geq 2s - 1$. Consequently, $|V_r(\vec{E}_r)|(2s - 1) \leq m \leq sn$. Therefore we have $|V_r(\vec{E}_r)| \leq n/(2 - 1/s)$. □

The following lemma summarizes the properties of $\mathcal{P}$ returned by Algorithm 11.

Lemma 4.27 *Given $G = (V, E, w)$ with $w_{\max}/w_{\min} = poly(n)$, let $\mathcal{P} \leftarrow Partition(G)$ be a set of graphs after the partition, then (1) $|\mathcal{P}| = poly(\log n)$; and (2) for each $\vec{P} = (V_P, \vec{E}_P, w_P) \in \mathcal{P}$, if $|V_P| > 2$ then for any $e \in \vec{E}_P$, $w(e) \in [\gamma, 2\gamma)$ for some $\gamma > 0$, and one of the following properties holds:*

Property 1: *For each $u \in V_P$, $d_u^{\texttt{out}}(\vec{P}) < \beta$.*

Property 2: *There exists i $(0 \leq i \leq \log(\eta/(\beta\varepsilon^2)))$, for each $u \in V_P$, $d_u^{\texttt{out}}(\vec{P}) \in [2^i\beta, 2^{i+1}\beta)$.*

Proof: We only need to bound the size of $\mathcal{P}$. The rest directly follows from the algorithm.

First, Line 6 and Line 10 will partition $\vec{E}'$ to $O(\log^2 n)$ (assuming $n > 1/\varepsilon$) sets. Second, we bound the number of recursion steps. Note that if we directly remove $E_s = \{(u, v) \in \vec{E}' \mid d_u^{\texttt{out}}(\vec{E}') < 2s\}$ from $\vec{E}'$, then by Lemma 4.26 we know that there are at most $O(\log n)$ recursion steps. The subtlety is that we first partition $\vec{E}'$ into $\vec{E}'_j$'s and then remove all $(u, v) \in \vec{E}'_j$ with $d_u^{\texttt{out}}(\vec{E}'_j) < 2s$. However, since $d_u^{\texttt{out}}(\vec{E}'_j) \leq d_u^{\texttt{out}}(\vec{E}')$, every edge in E_s will still be removed by at Line 12. Therefore $|\mathcal{P}| = O(\log^3 n)$. □

Algorithm 12: Spectral-Improved(G, ε)

```
  Input: $G = (V, E, w)$; a quality control parameter $\varepsilon$
  Output: A $(1 + \varepsilon, 0.01)$-spectral-sketch sk$(G)$ of $G$
1 Let $\mathcal{H} \leftarrow$ Partition(G);
2 sk$(G) \leftarrow \emptyset$;
3 foreach $H \in \mathcal{H}$ do
4     if $H$ satisfies Property 1 in Lemma 4.27 then
5         Add the whole $H$ to sk$(G)$;
6     else if $H$ satisfies Property 2 in Lemma 4.27 then
7         Add Spectral-S3$(H, \varepsilon)$ into sk$(G)$ ;
8 return $sk(G)$;
```

We summarize our main result.

Theorem 4.28 *Given an undirected positively weighted $G = (V, E, w)$ with $w_{\max}/w_{\min} \le poly(n)$, there is a sketching algorithm that outputs a $(1 + \varepsilon, 0.01)$-spectral-sketch of size $\tilde{O}(n/\varepsilon^{8/5})$.*

Proof: Our final algorithm is described in Algorithm 12. For each component $H \in \mathcal{P}$, we store the whole H if Property 1 in Lemma 4.27 holds (let $\mathcal{P}_1$ be this set), and we apply Theorem 4.24 on H (set $\delta = 1/\text{poly}\log n$) if Property 2 in Lemma 4.27 holds (let $\mathcal{P}_2$ be this set), thus the space usage is bounded by $\tilde{O}(n/\tilde{\varepsilon}^{\frac{8}{5}}) = \tilde{O}(n/\varepsilon^{\frac{8}{5}})$. Since by $|\mathcal{P}_1 \cup \mathcal{P}_2| = |\mathcal{P}| = O(\log^3 n)$ by Lemma 4.27, we can bound the total space by $\tilde{O}(n/\varepsilon^{\frac{8}{5}})$.

Our final estimator, given $x \in \mathbb{R}^n$, is

$$I_G = \sum_{H \in \mathcal{P}_2} I_H + \sum_{H \in \mathcal{P}_1} x^T L(H) x,$$

where I_H is defined in Equation (19). Similar to the proof of Lemma 4.20, we can bound $\mathbf{Var}[I_G]$ by $O(\varepsilon^2 \cdot (x^T L(G) x)^2)$. By a Chebyshev's inequality, I_G is a $(1+\varepsilon, 0.01)$-approximation of $x^T L(G) x$. $\square$

5 Positive-Semidefinite Matrices

In this section we consider sketching positive-semidefinite matrices, in the "for all" and "for each" models, respectively.

5.1 "For All" Model

We first show that in the "for all" case (for general PSD matrices A), there is no better sketching algorithm (up to a logarithmic factor) than simply storing the whole matrix.

Theorem 5.1 *For a general PSD matrix A and relative approximation $\varepsilon > 0$ that is a sufficiently small constant, every sketch sk(A) that satisfies the "for all" guarantee (with constant probability of success), even if all of entries of A are promised to be in the range $\{-1, -1 + 1/n^C, -1 + 2/n^C, \ldots, 1 - 1/n^C, 1\}$ for a sufficiently large constant $C > 0$, must use $\Omega(n^2)$ bits of space.*

Proof: Consider a net of $n \times n$ projection (thus PSD) matrices P onto $n/2$-dimensional subspaces U of $\mathbb{R}^n$. It is known (see Corollary 5.1 of [KT13], which uses a result of [AEK06]) that there exists a family $\mathcal{F}$ of $r = 2^{\Omega(n^2)}$ distinct matrices $P_1, \ldots, P_r$ so that for all $i \neq j$, $\|P_i - P_j\|_2 \geq \frac{1}{2}$. By rounding each of the entries of each matrix P_i to the nearest additive multiple of $1/n^C$, obtaining a symmetric matrix Q_i with entries in $\{-1, -1 + 1/n^C, \ldots, 1\}$, we obtain a family $\mathcal{F}'$ of $r = 2^{\Omega(n^2)}$ distinct matrices Q_i such that $\|Q_i - Q_j\|_2 \geq \frac{1}{4}$ for all $i \neq j$. This implies there is a unit vector x^* for which $\|Q_i x^* - Q_j x^*\|_2 \geq \frac{1}{4}$, or equivalently,

$$\|Q_i x^*\|_2^2 + \|Q_j x^*\|_2^2 - 2\langle (x^*)^T Q_i^T, Q_j x^* \rangle \geq \frac{1}{16}. \tag{20}$$

In the rest of the proof, for simplicity, we often abuse the notation by directly using a matrix M to denote the column space of M. Let J be the subspace of $\mathbb{R}^n$ which is the intersection of the spaces spanned by the columns of Q_i and Q_j, let K_i be the subspace of Q_i orthogonal to J, and let K_j be the subspace of Q_j orthogonal to J. We identify J, K_i, and K_j, with their corresponding projection matrices. To maximize the lefthand side of (20), we can assume the unit vector x^* is in the span of the union of columns of J, K_i, and K_j. We can therefore write $x^* = Jx^* + K_i x^* + K_j x^*$, and note that the three summand vectors are orthogonal to each other. Expanding (20), the lefthand side is equal to

$$2\|Jx^*\|_2^2 + \|K_i x^*\|_2^2 + \|K_j x^*\|_2^2 - 2\|Jx^*\|_2^2 = \|K_i x^*\|_2^2 + \|K_j x^*\|_2^2.$$

Hence, by (20), it must be that either $\|K_i x^*\|_2^2 \geq \frac{1}{32}$ or $\|K_j x^*\|_2^2 \geq \frac{1}{32}$. This implies the vector $z = K_i x^*$ satisfies $\|Q_i z\|_2^2 \geq \frac{1}{32}$, but $\|Q_j z\|_2^2 = 0$.

Therefore, if there were a sketch sk(A) which had the "for all" guarantee for any matrix $A \in \mathcal{F}$, one could query sk(A) on the vector z given above for each pair $Q_i, Q_j \in \mathcal{F}$, thereby recovering the matrix $A \in \mathcal{F}$. Hence, sk(A) is an encoding of an arbitrary element $A \in \mathcal{F}$ which implies that the size of sk(A) is $\Omega(\log |\mathcal{F}|) = \Omega(n^2)$ bits, completing the proof. $\square$

5.2 "For Each" Model

In the "for each" case for PSD matrices A, we can use the Johnson-Lindenstrauss lemma to obtain a sketch sk(A) of size $O(n/\varepsilon^2 \cdot \log(1/\delta) \log n)$. One instantiation of this lemma works by choosing a random $r \times n$ matrix S with i.i.d. entries in $\{-1/\sqrt{r}, +1/\sqrt{r}\}$ for $r = \Theta(\varepsilon^{-2} \log 1/\delta)$. Then for any $n \times n$ matrix B and fixed x, we have $\mathbf{Pr}[\|SBx\|_2^2 \in (1 \pm \varepsilon)\|Bx\|_2^2] \geq 1 - \delta$. The sketch simply computes a matrix B where B satisfies $B^T B = A$ (such a decomposition exists since A is PSD), hence $x^T A x = x^T B^T B x = \|Bx\|_2^2$. Therefore, the sketch sk(A) can be SB, which is an $r \times n$ matrix of size $\tilde{O}(n/\varepsilon^2)$ bits, and the estimate it produces would be $\|SBx\|^2$.

For general PSD matrices A, this $\tilde{O}(n/\varepsilon^2)$ upper bound turns out to be optimal up to logarithmic factors.

Theorem 5.2 *For a general PSD matrix A and relative approximation $\varepsilon \in (1/\sqrt{n}, 1)$, every sketch sk$(A)$ that satisfies the "for each" guarantee (with constant probability) must use $\Omega(n/\varepsilon^2)$ bits of space.*

The proof is similar (and inspired from) a result in [GWWZ15] for approximating the number of non-zero entries of Ax. Before giving the proof, we need some tools. Let $\Delta(a, b)$ be the Hamming distance between two bitstrings a and b.

Lemma 5.3 (modified from [JKS08]) *Let x be a random bitstring of length $\gamma = 1/\varepsilon^2$, and let i be a random index in $[\gamma]$. Choose γ public random bitstrings $r^1, \ldots, r^\gamma$, each of length γ. Create γ-length bitstrings a, b as follows:*

- *For each $j \in [\gamma]$, $a_j = majority\{r^j_k \mid$ indices k for which $x_k = 1\}$.*
- *For each $j \in [\gamma]$, $b_j = r^j_i$.*

There is a procedure which, with probability $1/2 + \delta$ for a constant $\delta > 0$, can determine the value of x_i from any $c\sqrt{\Delta(a, b)}$-additive approximation to $\Delta(a, b)$, provided $c > 0$ is a sufficiently small constant.

We introduce the Indexing problem. In Indexing, we have two randomized parties Alice and Bob. Alice has $x \in \{0, 1\}^n$, and Bob has an index $i \in [n]$. The communication is one-way from Alice to Bob, and the goal is for Bob to compute x_i.

Lemma 5.4 (see, e.g., [KN97]) *To solve the Indexing problem with success probability $1/2 + \delta$ for any constant $\delta > 0$, Alice needs to send Bob $\Omega(n)$ bits even with shared randomness.*

Proof: (for Theorem 5.2) Let $\gamma = 1/\varepsilon^2$. The proof is by a reduction from the Indexing problem, where Alice has a random bitstring z of length $(n - \gamma) \cdot \gamma$, and Bob has an index $\ell \in [(n - \gamma) \cdot \gamma]$.

Partition z into $n - \gamma$ contiguous substrings $z^1, z^2, \ldots, z^{n-\gamma}$. Alice constructs a 0/1 matrix B as follows: she uses shared randomness to sample γ random bitstrings $r^1, \ldots, r^\gamma$, each of length γ. For the leftmost $\gamma \times \gamma$ submatrix of B, in the i-th column for each $i \in [\gamma]$, Alice uses $r^1, \ldots, r^\gamma$ and the value i to create the γ-length bitstring b according to Lemma 5.3 and assigns it to this column. Next, in the remaining $n - \gamma$ columns of B, in the j-th column for each $j \in \{\gamma + 1, \ldots, n\}$, Alice uses $z^{j-\gamma}$ and $r^1, \ldots, r^\gamma$ to create the γ-length bitstring a according to Lemma 5.3 and assigns it to this column.

Our PSD matrix is set to be $A = B^T B$.

Alice then sends Bob the sketch sk(A), together with $\{\|B^i\|_2^2 \ (i \in [n])\}$ and $\{nnz(B^i) \ (i \in [n])\}$ where $nnz(x)$ is the number of non-zero coordinates of x. Note that both $\{\|B^i\|_2^2 \ (i \in [n])\}$ and $\{nnz(B^i) \ (i \in [n])\}$ can be conveyed using $O(n \log(1/\varepsilon)) = o(n/\varepsilon^2)$ bits of communication, which is negligible.

Bob creates a vector x by putting a 1 in the i-th and j-th coordinates, where i, j $(i \in [\gamma], j \in \{\gamma + 1, \ldots, n\})$ satisfies $\ell = i + (j - \gamma - 1) \cdot \gamma$. Then Bx is simply the sum of the i-th and j-th columns of B, denoted by $B^i + B^j$. Note that B^i, B^j correspond to a pair of (a, b) created from $r^1, \ldots, r^\gamma$ and $z^{j-\gamma}$ (and $(z^{j-\gamma})_i = z_\ell$). Now, from a $(1 + c\varepsilon)$-approximation to $x^T Ax = \|B^i + B^j\|_2^2$ for a sufficiently small constant c, and exact values of $\|B^i\|_2^2$ and $\|B^j\|_2^2$, Bob can approximate $2\langle B^i, B^j\rangle = \|B^i + B^j\|_2^2 - \|B^i\|_2^2 - \|B^j\|_2^2$ up to an additive error $c\varepsilon \cdot \|B^i + B^j\|_2^2 \leq c'/\varepsilon$ for a sufficiently small constant c'. Then, using a (c'/ε)-additive approximation of $2\langle B^i, B^j\rangle$, and exact values of $nnz(B^i)$ and $nnz(B^j)$, Bob can approximate $\Delta(B^i, B^j) = nnz(B^i) + nnz(B^j) - 2\langle B^i, B^j\rangle$ up to a $c'/\varepsilon = c''\sqrt{\Delta(B^i, B^j)}$ additive approximation for a sufficiently small constant c'', and consequently compute z_ℓ correctly with probability $1/2 + \delta$ for a constant $\delta > 0$ (by Lemma 5.3).

Therefore, any algorithm that produces a $(1 + c\varepsilon)$-approximation of $x^T Ax = \|B^i + B^j\|_2^2$ with probability $1 - \delta'$ for some sufficiently small constant $\delta' < \delta$ can be used to solve the Indexing problem of size $(n - \gamma)\gamma = \Omega(n/\varepsilon^2)$ with probability $1 - \delta' - (1/2 - \delta) > 1/2 + \delta''$ for a constant $\delta'' > 0$. The theorem follows by the reduction and Lemma 5.4. $\square$

References

[AEK06] P.-A. Absil, A. Edelman, and P. Koev. On the largest principal angle between random subspaces. *Linear Algebra and its applications*, 414(1):288–294, 2006. doi:10.1016/j.laa.2005.10.004.

[AGM12a] K. J. Ahn, S. Guha, and A. McGregor. Analyzing graph structure via linear measurements. In *23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 459–467, 2012.

[AGM12b] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 5–14, 2012.

[AHK05] S. Arora, E. Hazan, and S. Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *46th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '05, pages 339–348. IEEE Computer Society, 2005. doi:10.1109/SFCS.2005.35.

[Alo97] N. Alon. On the edge-expansion of graphs. *Comb. Probab. Comput.*, 6(2):145–152, June 1997. doi:10.1017/S096354839700299X.

[ARV09] S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2):1–37, 2009. doi:10.1145/1502793.1502794.

[BBDS13] J. Blocki, A. Blum, A. Datta, and O. Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Innovations in Theoretical Computer Science, ITCS 2013*, pages 87–96, 2013.

[BGPW13] M. Braverman, A. Garg, D. Pankratov, and O. Weinstein. Information lower bounds via self-reducibility. In *Computer Science Theory and Applications*, volume 7913 of *Lecture Notes in Computer Science*, pages 183–194. Springer, 2013. doi:10.1007/978-3-642-38536-0_16.

[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. doi:10.1145/237814.237827.

[BSS14] J. D. Batson, D. A. Spielman, and N. Srivastava. Twice-ramanujan sparsifiers. *SIAM Review*, 56(2):315–334, 2014. doi:10.1137/130949117.

[CW09] K. L. Clarkson and D. P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pages 205–214, 2009.

[FHHP11] W. S. Fung, R. Hariharan, N. J. Harvey, and D. Panigrahi. A general framework for graph sparsification. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 71–80. ACM, 2011. doi:10.1145/1993636.1993647.

[GRU12] A. Gupta, A. Roth, and J. Ullman. Iterative constructions and private data release. In *9th International Conference on Theory of Cryptography*, TCC'12, pages 339–356. Springer-Verlag, 2012. doi:10.1007/978-3-642-28914-9_19.

[GWWZ15] D. V. Gucht, R. Williams, D. P. Woodruff, and Q. Zhang. On the communication complexity of distributed set-joins. In *PODS, to appear*, 2015.

[GY03] J. L. Gross and J. Yellen. *Handbook of Graph Theory.* CRC Press, Abingdon, 2003.

[HW96] M. R. Henzinger and D. P. Williamson. On the number of small cuts in a graph. *Inf. Process. Lett.*, 59(1):41–44, 1996.

[JKS08] T. S. Jayram, R. Kumar, and D. Sivakumar. The one-way communication complexity of hamming distance. *Theory of Computing*, 4(1):129–135, 2008. doi:10.4086/toc.2008.v004a006.

[JT12] P. Jain and A. Thakurta. Mirror descent based database privacy. In *15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012*, pages 579–590, 2012.

[Kar00] D. R. Karger. Minimum cuts in near-linear time. *J. ACM*, 47(1):46–76, 2000. `doi:10.1145/331605.331608`.

[KLM$^+$14] M. Kapralov, Y. T. Lee, C. Musco, C. Musco, and A. Sidford. Single pass spectral sparsification in dynamic streams. In *55th Annual Symposium on Foundations of Computer Science*, FOCS '14, pages 561–570. IEEE Computer Society, 2014. `arXiv:1407.1289`, `doi:10.1109/FOCS.2014.66`.

[KN97] E. Kushilevitz and N. Nisan. *Communication complexity.* Cambridge: Cambridge Univ, 1997.

[KNPR15] H. Klauck, D. Nanongkai, G. Pandurangan, and P. Robinson. Distributed computation of large-scale graph problems. In *26th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 391–410. SIAM, 2015. `arXiv:1311.6209`, `doi:10.1137/1.9781611973730.28`.

[KP12] M. Kapralov and R. Panigrahy. Spectral sparsification via random spanners. In *3rd Innovations in Theoretical Computer Science Conference*, pages 393–398. ACM, 2012. `doi:10.1145/2090236.2090267`.

[KT13] M. Kapralov and K. Talwar. On differentially private low rank approximation. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013*, pages 1395–1414, 2013.

[Mad10] A. Madry. Fast approximation algorithms for cut-based problems in undirected graphs. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 245–254. IEEE, 2010.

[McG14] A. McGregor. Graph stream algorithms: a survey. *SIGMOD Record*, 43(1):9–20, 2014.

[Nil91] A. Nilli. On the second eigenvalue of a graph. *Discrete Math*, 91:207–210, 1991. `doi:10.1016/0012-365X(91)90112-F`.

[She09] J. Sherman. Breaking the multicommodity flow barrier for $O(\sqrt{\log n})$-approximations to sparsest cut. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 363–372, 2009.

[SS11] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, December 2011. `doi:10.1137/080734029`.

[ST04] D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 81–90. ACM, 2004. `doi:10.1145/1007352.1007372`.

[ST11] D. A. Spielman and S.-H. Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011. `doi:10.1137/08074489X`.

[Upa13] J. Upadhyay. Random projections, graph sparsification, and differential privacy. In *19th International Conference on Advances in Cryptology*, ASIACRYPT 2013, pages 276–295. Springer-Verlag, 2013. `doi:10.1007/978-3-642-42033-7_15`.

[Upa14] J. Upadhyay. Circulant matrices and differential privacy. *CoRR*, abs/1410.2470, 2014. `arXiv:1410.2470`.

[WZ13] D. P. Woodruff and Q. Zhang. When distributed computation is communication expensive. In *Distributed Computing - 27th International Symposium, DISC 2013*, pages 16–30, 2013. `doi:10.1007/978-3-642-41527-2_2`.

A General Matrices, "For Each" Model

Theorem A.1 *Any sketch sk(A) of a general $n \times n$ matrix A that satisfies the "for each" guarantee with probability* 0.9*, even when all entries of A are promised to be in the set* $\{0, 1\}$*, must use* $\Omega(n^2)$ *bits of space.*

Proof: Let A be a symmetric matrix with zero on the diagonal, and a random bit in every other entry. Set the query vector $x = (e_i + e_j)$. Then using $\frac{1}{2}x^T Ax$ we can recover the entry $A_{i,j}$, with probability 0.9. Think the sketching problem as a communication problem where Alice holds the matrix A; she sends a message (the sketch) M to Bob such that Bob can recover each entries of A with probability 0.9 (except for the diagonal entries, which are fixed to be 0, Bob can recover exactly). Then,

$$\begin{aligned} H(A \mid M) &= \sum_{i,j \in [n]} H(A_{i,j} \mid M) \quad (A_{i,j} \text{ are independent}) \\ &\leq (H_2(0.9) + 0.1) \cdot n^2 \quad \text{(Fano's inequality)} \\ &< 0.6n^2. \end{aligned}$$

Thus $H(M) \geq H(A) - H(A \mid M) = \Omega(n^2)$. □

B Reduction from SDD Matrices to Laplacian Matrices

In this section we show that the quadratic form of an SDD matrix, $x^T Ax$, can be reduced to the quadratic form of a Laplacian, therefore our upper bounds for Laplacian matrices in Section 4.1 and Section 4.2 can be extended to SDD matrices.

An SDD matrix A has the property that $A_{i,i} \geq \sum_{j \neq i} |A_{i,j}|$ for all i. In the case when $A_{i,i} = \sum_{i \neq j} |A_{i,j}|$ for all i, we can write A as $A_p + A_n + D$ where D is the diagonal of A, A_n is the matrix consisting of only the negative off-diagonal entries of A, and A_p is the matrix consisting of only the positive off-diagonal entries of A. It is straightforward to verify that

$$\begin{pmatrix} x^T & -x^T \end{pmatrix} \begin{pmatrix} D + A_n & -A_p \\ -A_p & D + A_n \end{pmatrix} \begin{pmatrix} x \\ -x \end{pmatrix} = 2x^T Ax.$$

The matrix $\begin{pmatrix} D + A_n & -A_p \\ -A_p & D + A_n \end{pmatrix}$ is clearly a Laplacian matrix.

For the general case when $A_{i,i} \geq \sum_{i \neq j} |A_{i,j}|$. We can remove some "weights" from the diagonal entries of A, so that A can be written as $A = D + B$ where D is a diagonal matrix and B satisfies the requirement $B_{i,i} = \sum_{i \neq j} |B_{i,j}|$ for all i. We then have $x^T Ax = x^T Dx + x^T Bx$. The matrix D can be stored explicitly, and $x^T Bx$ can be reduced to the quadratic form of a Laplacian matrix as discussed above.

Theorem B.1 *Given an $n \times n$ SDD matrix A, let* $w_{\max} = \max_{i,j} |A_{i,j}|$ *and* $w_{\min} = \min_{i,j \text{ with } A_{i,j} \neq 0} |A_{i,j}|$*, and assume* $w_{\max}/w_{\min} = poly(n)$ *We can then construct a sketch of A that gives a* $(1 + \varepsilon, 0.99)$*-approximation to $x^T Ax$ for any fixed* $x \in \mathbb{R}^n$*. The size of this sketch is* $\tilde{O}(n/\varepsilon^{8/5})$ *bits.*