On Sketching Quadratic Forms*

Alexandr Andoni†
Columbia University
New York, NY, USA
andoni@cs.columbia.edu

Jiecao Chen‡
Indiana University
Bloomington, IN, USA
jiecchen@indiana.edu

Robert Krauthgamer§
Weizmann Institute of Science
Rehovot, Israel
robert.krauthgamer@weizmann.ac.il

Bo Qin¶
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
bqin@cse.ust.hk

David P. Woodruff‖
IBM Almaden Research
San Jose, CA, USA
dpwoodru@us.ibm.com

Qin Zhang‡
Indiana University
Bloomington, IN, USA
qzhangcs@indiana.edu

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*.

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.

Categories and Subject Descriptors

F.2.0 [**Theory of Computation**]: ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY—*General*

Keywords

Quadratic Forms, Sketching, Graph Sparsification, Lower Bound

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 $x^T Ax$ for every query vector $x \in \mathbb{R}^n$. Typically, we aim at $(1+\varepsilon)$-approximation, i.e., the estimate is in the range $(1 \pm \varepsilon)x^T Ax$, 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

*A full version of this paper is available at arXiv:1511.06099.

†Work done in part while the author was at Microsoft Research Silicon Valley.

‡Work supported in part by NSF CCF-1525024, and IU's Office of the Vice Provost for Research through the Faculty Research Support Program.

§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.

¶Work of this author partially supported by Hong Kong RGC GRF grant 16208415.

‖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.

ITCS'16, January 14 - 16, 2016, Cambridge, MA, USA

ISBN 978-1-4503-4057-1/16/01…$15.00

DOI: http://dx.doi.org/10.1145/2840728.2840753

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 $x^T Ax$ 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 $x^T Ax$ 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 $\|By - c\|_2^2 = \|Mx\|_2^2 = x^T M^T Mx$, and thus regression queries can be modeled by a quadratic form over the PSD matrix $A = M^T M$. 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 2.1); 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 2.1). 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 2.2). This 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 $n\varepsilon^{-1}\operatorname{polylog}(n)$ bits of space (see Section 2.3.1). 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 $n^{3/2}\operatorname{polylog}(n)$ 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 2.8).

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 $\{1, 2, \ldots, n\}$ gives the average squared distortion of a line embedding of G. We design in Section 2.3.2 a sketch for spectral queries that uses $n\varepsilon^{-1.6}\operatorname{polylog}(n)$ 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 illus-

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. 2.1	$\tilde{O}(n\epsilon^{-2})$ Sec. 2.1	$\Omega(n\varepsilon^{-2})$ Sec. 2.1
Laplacian, SDD	$\tilde{O}(n\varepsilon^{-2})$ [BSS14]	$\Omega(n\varepsilon^{-2})$ [BSS14]	$\tilde{O}(n\varepsilon^{-1.6})$ Sec. 2.3.2	$\Omega(n\varepsilon^{-1})$ Sec. 2.3.1
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. 2.2	$\tilde{O}(n\varepsilon^{-1})$ Sec. 2.3.1	$\Omega(n\varepsilon^{-1})$ Sec. 2.3.1
edge-count:	$O(n\varepsilon^{-2})$ [BSS14]	$\Omega(n\varepsilon^{-2})$ Sec. 2.2		

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

trative 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

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

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, $\Phi_G(S) \le \varepsilon^{1/3}$. 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 $x^T Lx = \sum_{(u,v)\in E}(x_u - x_v)^2 w(u,v)$ as

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

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. 2.3.1 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. MAIN THEOREMS

Due to space constraints, this extended abstract lists only the main results of our paper. We refer the reader to the full version of the paper for other results and the proof details.

2.1 Positive-Semidefinite Matrices

For PSD matrices, we show the following two lower bounds, for the "for all" and "for each" models, respectively, which resolve the sketching complexities for PSD matrices up to a logarithmic factor. Indeed, the first lower bound matches the trivial upper bound of storing the whole matrix, and the second one matches a straightforward application of the Johnson-Lindenstrauss dimension reduction lemma.

THEOREM 2.1. *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), must use $\Omega(n^2)$ bits of space. This is true 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 $C > 0$,*

THEOREM 2.2. *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.*

2.2 Symmetric Diagonally Dominant Matrices, "For All" Model

Our first main result is an $\Omega(n/\varepsilon^2)$ space lower space for sketching SDD matrices under the "for all" guarantee. In fact, we prove the lower bound $\Omega(n/\varepsilon^2)$ for the special

case where A is a Laplacian matrix and for cut queries $x \in \{0,1\}^n$.

We additionally show in Appendix B that the quadratic form of an SDD matrix can be reduced 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 a logarithmic factor.

THEOREM 2.3. *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]

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

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

If the sketch must take the form of a graph H (i.e., be a cut sparsifier), then a straightforward application of Theorem 2.3 implies that H must have $\Omega(n/(\varepsilon^2 \log n))$ edges. The following theorem improves this lower bound by a logarithmic factor, and obtains a bound that is tight up to constant factors.

THEOREM 2.4. *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.*

Proof outline for Theorem 2.3.

The proof uses the following communication lower bound for a version of the Gap-Hamming-Distance problem. Fix $c = 10^{-3}$.

THEOREM 2.5. *Consider a distributional communication problem, where 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]

- *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.

[1]The probability is over the randomness of the two algorithms; more precisely, 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.

We then prove Theorem 2.3 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 $T \subseteq R(G_j)$. Let $N(v)$ be the neighbor set of the vertex v. We have the following lemma.

LEMMA 2.6. *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*

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

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

Finally, Bob can decide whether $\Delta(s_i, t)$ is $\geq \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$ or $\leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$ using L_{high}, thereby solving the (variation of the) Gap-Hamming problem, which implies that $m \geq \Omega(n/\varepsilon^2)$, and proves Theorem 2.3.

2.3 Symmetric Diagonally Dominant Matrices, "For Each" Model

Our second and third main results design sketches of SDD matrices in the "for each" model, namely, sketch-size $\tilde{O}(n/\varepsilon)$ for cut queries $x \in \{0,1\}^n$ and sketch-size $\tilde{O}(n/\varepsilon^{1/6})$ for spectral queries $x \in \mathbb{R}^n$. Again, due to the reduction shown in Appendix B, sketching of SDD matrices and of Laplacian matrices are equivalent, hence, we only need to prove these bounds for Laplacian matrices.

2.3.1 Laplacian Matrices with Cut Queries

Given a Laplacian matrix L, let $G = G(L) = (V, E, w)$ be the corresponding graph, and let $n = |V|$. For cut queries, we obtain the following upper bound.

THEOREM 2.7. *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 $w(S, \bar{S})$.*

We can also show a matching lower bound (up to logarithmic factors).

THEOREM 2.8. *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,*

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

Then $s \geq \Omega(n/\varepsilon)$.

We next outline the proof of Theorem 2.7. The starting point of our algorithm design is to consider the a special graph (S1-graph) under a special set of queries (the weighted cut value $w(S,\bar{S}) \leq 5$). We then show how to massage a general graph with polynomial weights to a set of S1-graph's (using the importance sampling and a hierarchical partition on the edge weights, and by repeatedly finding sparse cuts), and how to handle general cut queries. Finally we remove the polynomial weight constraints using a finer grade of "pruning and partition" step making use of a minimum-weight spanning tree of the graph. Due to the space constraints, we only present here the definition of the S1-graph and the algorithm for sketching S1-graph for cut queries $w(S,\bar{S}) \leq 5$.

DEFINITION 2.9 (S1-GRAPH). *We say an undirected weighted graph $G = (V, E, w)$ is an* S1-graph *if it satisfies the following.*

1. *All edge weights are within a factor of 2, i.e. $\forall e \in E$, $w(e) \in [\gamma, 2\gamma)$ for some $\gamma > 0$.*
2. *The expansion constant of G is $\Gamma_G \geq \frac{1}{\varepsilon}$, where $\Gamma_G := \min_{|S| \leq n/2} \frac{|E(S,\bar{S})|}{|S|}$, where $E(S,\bar{S}) = \{(u,v) \in E \mid u \in S, v \in \bar{S}\}$.*

The sketching algorithm for S1-graph and the special cut queries $w(S,\bar{S}) \leq 5$ is fairly simple: we first add all weighted degrees of vertices to the sketch, and then for each vertex we sample uniformly with replacement a set of $1/\varepsilon$ adjacent edges and store them in the sketch. As mentioned in the techniques overview (Section 1.2), using a "difference-based" estimator together with the fact that there is no sparse cut and the assumption that $w(S,\bar{S}) \leq 5$ (and consequently the cut is of size at most $O(1/\varepsilon^2)$ because the weight of each edge of an S1-graph is at least ε^2), we can tightly bound the variance of the estimator of a cut $w(S,\bar{S})$ in an S1-graph, which allows a small set of edge samples and thus the small space usage.

2.3.2 Laplacian Matrices with Spectral Queries

THEOREM 2.10. *Given a Laplacian matrix L with polynomially-bounded entries, and $\varepsilon \in (0,1)$, there exists a spectral sketch of size $\tilde{O}(n/\varepsilon^{8/5})$ bits such that for every query $x \in \mathbb{R}^n$, the sketch can produce with high probability a $(1+\varepsilon)$ to $x^T L x$.*

We next outline the proof of Theorem 2.10. Let G be the corresponding graph of the Laplacian matrix L. To prove Theorem 2.10 we again start from a simple type of graphs we call S2-graph, whose definition is given below. Note that an S2-graph again has almost uniform edge weights, and a small Cheeger's constant which is similar in spirit to the expansion constant, but has more to do with the spectral properties of the graph. We then finish the proof via a long sequence of generalizations and optimizations, as partly discussed in the technique overview (Section 1.2).

DEFINITION 2.11 (S2-GRAPH). *We say an undirected weighted graph $G = (V, E, w)$ is an* S2-graph *if it satisfies the following.*

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

The sketching algorithm for S2-graph is slightly more complicated than that for S1-graph in the cut query case: We first add all weighted degrees of vertices to the sketch. Next, we partition the vertices to two sets $\mathcal{S}$ and $\mathcal{L}$, where $\mathcal{S}$ contains all vertices of weighted degrees less than $\gamma c_\alpha \varepsilon^{-5/3}$, and $\mathcal{L}$ contains the rest of the vertices. We then store all adjacent edges of vertices in $\mathcal{S}$ in the sketch, and then delete these vertices and edges from G, obtaining a graph $G' = (V', E', w')$. For each vertex in G', we sample with replacement $c_\alpha \varepsilon^{-5/3}$ of its adjacent edges with probability proportional to the edge weights, and store them in the sketch. This algorithm, however, only gives a sketch of size $\tilde{O}(n/\varepsilon^{5/3})$ bits (even for this special type of graphs). To improve it to $\tilde{O}(n/\varepsilon^{1.6})$ we need a finer process of edge partition.

3. REFERENCES

[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.

[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.

[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.

[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.

[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.

[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.

APPENDIX

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 2.3.1 and Section 2.3.2 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.*