HHS Public Access

Author manuscript

Proc ACM Manag Data. Author manuscript; available in PMC 2024 November 20.

Published in final edited form as:

Proc ACM Manag Data. 2024 May ; 2(2): . doi:10.1145/3651148.

Tight Lower Bounds for Directed Cut Sparsification and Distributed Min-Cut

Yu Cheng[*], **Max Li**[†], **Honghao Lin**[‡], **Zi-Yi Tai**[§], **David P. Woodruff**[¶], **Jason Zhang**[‖]

[*] Brown University.

[†] Carnegie Mellon University.

[‡] Carnegie Mellon University.

[§] Carnegie Mellon University.

[¶] Carnegie Mellon University.

[‖] Carnegie Mellon University.

Abstract

In this paper, we consider two fundamental cut approximation problems on large graphs. We prove new lower bounds for both problems that are optimal up to logarithmic factors.

The first problem is to approximate cuts in balanced directed graphs. In this problem, the goal is to build a data structure that $(1 \pm \varepsilon)$-approximates cut values in graphs with n vertices. For arbitrary directed graphs, such a data structure requires $\Omega(n^2)$ bits even for constant ε. To circumvent this, recent works study β-balanced graphs, meaning that for every directed cut, the total weight of edges in one direction is at most β times that in the other direction. We consider two models: the *for-each* model, where the goal is to approximate each cut with constant probability, and the *for-all* model, where all cuts must be preserved simultaneously. We improve the previous $\Omega(n\sqrt{\beta/\varepsilon})$ lower bound to $\widetilde{\Omega}(n\sqrt{\beta}/\varepsilon)$ in the for-each model, and we improve the previous $\Omega(n\beta/\varepsilon)$ lower bound to $\Omega(n\beta/\varepsilon^2)$ in the for-all model. [1] This resolves the main open questions of (Cen et al., ICALP, 2021).

The second problem is to approximate the global minimum cut in a local query model, where we can only access the graph via degree, edge, and adjacency queries. We improve the previous $\Omega(\frac{m}{k})$ query complexity lower bound to $\Omega(\min\{m, \frac{m}{\varepsilon^2 k}\})$ for this problem, where m is the number of edges, k is the size of the minimum cut, and we seek a $(1+\varepsilon)$-approximation. In addition, we show that existing upper bounds with slight modifications match our lower bound up to logarithmic factors.

[1] In this paper, we use $\widetilde{O}(\cdot)$ and $\widetilde{\Omega}(\cdot)$ to hide logarithmic factors in its parameters.

yu_cheng@brown.edu .

1 Introduction

The notion of cut sparsifiers has been extremely influential. It was introduced by Benczúr and Karger [BK96] and it is the following: Given a graph $G = (V, E, w)$ with $n = |V|$ vertices, $m = |E|$ edges, edge weights $w_e \geq 0$, and a desired error parameter $\varepsilon > 0$, a $(1 \pm \varepsilon)$ *cut sparsifier* of G is a subgraph H on the same vertex set V with (possibly) different edge weights, such that H approximates the value of every cut in G within a factor of $(1 \pm \varepsilon)$. Benczúr and Karger [BK96] showed that every undirected graph has a $(1 \pm \varepsilon)$ cut sparsifier with only $O(n \log n/\varepsilon^2)$ edges. This was later extended to the stronger notion of spectral sparsifiers [ST11] and the number of edges was improved to $O(n/\varepsilon^2)$ [BSS12]; see also related work with different bounds for both cut and spectral sparsifiers [FHHP19, KP12, ST04, SS11, LS17, CKST19].

In the database community, a key result is the work of [AGM12], which shows how to construct a sparsifer using $\widetilde{O}(n/\varepsilon^2)$ linear measurements to $(1 + \varepsilon)$-approximate all cut values. Sketching massive graphs arises in various applications where there are entities and relationships between them, such as webpages and hyperlinks, people and friendships, and IP addresses and data flows. As large graph databases are often distributed or stored on external memory, sketching algorithms are useful for reducing communication and memory usage in distributed and streaming models. We refer the readers to [McG14] for a survey of graph stream algorithms in the database community.

For very small values of ε, the $1/\varepsilon^2$ dependence in known cut sparsifiers may be prohibitive. Motivated by this, the work of [ACK+16] relaxed the cut sparsification problem to outputting a data structure D, such that for any fixed cut $S \subset V$, the value $D(S)$ is within a $(1 \pm \varepsilon)$ factor of the cut value of S in G with probability at least $2/3$. Notice the order of quantifiers — the data structure only needs to preserve the value of any fixed cut (chosen independently of its randomness) with high constant probability. This is referred to as the *for-each* model, and the data structure is called a *for-each cut sketch*. Surprisingly, [ACK+16] showed that every undirected graph has a $(1 \pm \varepsilon)$ for-each cut sketch of size $\widetilde{O}(n/\varepsilon)$ bits, reducing the dependence on ε to linear. They also showed an $\Omega(n/\varepsilon)$ bits lower bound in the for-each model. The improved dependence on ε is indeed coming from relaxing the original sparsification problem to the for-each model: [ACK+16] proved an $\Omega(n/\varepsilon^2)$ bit lower bound on any data structure that preserves all cuts simultaneously, which is referred to as the *for-all* model. This lower bound in the for-all model was strengthened to $\Omega(n \log n/\varepsilon^2)$ bits in [CKST19].

While the above results provide a fairly complete picture for undirected graphs, a natural question is whether similar improvements are possible for directed graphs. This is the main question posed by [CCPS21]. For directed graphs, even in the for-each model, there is an $\Omega(n^2)$ lower bound without any assumptions on the graph. Motivated by this, [EMPS16, IT18, CCPS21] introduced the notion of β*-balanced* directed graphs, meaning that for every directed cut $(S, V \setminus S)$, the total weight of edges from S to $V \setminus S$ is at most β times that from $V \setminus S$ to S. The notion of β-balanced graphs turned out to be very useful for

directed graphs, as [IT18, CCPS21] showed an $\widetilde{O}(n\sqrt{\beta}/\varepsilon)$ upper bound in the for-each model, and an $\widetilde{O}\left(n\beta/\varepsilon^2\right)$ upper bound in the for-all model, thus giving non-trivial bounds for both problems for small values of β. The work of [CCPS21] also proved lower bounds: they showed an $\Omega(n\sqrt{\beta/\varepsilon})$ lower bound in the for-each model, and an $\Omega(n\beta/\varepsilon)$ lower bound in the for-all model. While their lower bounds are tight for constant ε, there is a quadratic gap for both models in terms of the dependence on ε. The main open question of [CCPS21] is to determine the optimal dependence on ε, which we resolve in this work.

Recent work further explored *spectral sketches*, faster computation of sketches, and sparsification of Eulerian graphs (β-balanced graphs with $\beta = 1$) [ACK+16, JS18, CKK+18, CGP+23, SW19]. In this paper, we focus on the space complexity of cut sketches for general values of β.

As observed in [ACK+16], one of the main ways to use for-each cut sketches is to solve the distributed minimum cut problem. This is the problem of computing a $(1+\varepsilon)$-approximate global minimum cut of a graph whose edges are distributed across multiple servers. One can ask each server to compute a (1 ± 0.2) for-all cut sketch and a $(1 \pm \varepsilon)$ for-each cut sketch. This allows one to find all $O(1)$-approximate minimum cuts, and because there are at most $n^{O(C)}$ cuts with value within a factor of C of the minimum cut, one can query all these $\text{poly}(n)$ cuts using the more accurate for-each cut sketches, resulting in an optimal linear in $1/\varepsilon$ dependence in the communication.

Motivated by this connection to distributed minimum cut estimation, we also consider the problem of directly approximating the minimum cut in a *local query* model, which was introduced in [RSW18] and studied for minimum cut in [ER18, BGMP21]. The model is defined as follows.

Let $G(V, E)$ be an unweighted and undirected graph, where the vertex set V is known but the edge set E is unknown. In the *local query* model, we have access to an oracle that can answer the following three types of local queries:

1. Degree query: Given $u \in V$, the oracle returns the degree of u.
2. Edge query: Given $u \in V$ and index i, the oracle returns the i-th neighbor of u, or $\perp$ if the edge does not exist.
3. Adjacency query: Given $u, v \in V$, the oracle returns whether $(u, v) \in E$.

In the Min-Cut problem, our goal is to estimate the global minimum cut up to a $(1 \pm \varepsilon)$-factor using these local queries. The complexity of the problem is measured by the number of queries, and we want to use as few queries as possible. For this problem we focus on undirected graphs.

Previous work [ER18] showed an $\Omega\left(\frac{m}{k}\right)$ query complexity lower bound, where k is the size of the minimum cut. The main open question is what the dependence on ε should be. There is also an $\widetilde{O}(\frac{m}{k\ \text{poly}(\varepsilon)})$ upper bound in [BGMP21], and a natural question is to close this gap.

1.1 Our Results

We resolve the main open questions mentioned above.

Cut Sketch for Balanced (Directed) Graphs.—We study the space complexity of $(1 \pm \varepsilon)$ cut sketches for n-node β-balanced (directed) graphs. Previous work [IT18, CCPS21] gave an $\widetilde{O}\left(n\beta/\varepsilon^2\right)$ upper bound in the for-all model and an $\widetilde{O}(n\sqrt{\beta}/\varepsilon)$ upper bound in the for-each model, along with an $\Omega(n\beta/\varepsilon)$ lower bound and an $\Omega(n\sqrt{\beta/\varepsilon})$ lower bound, respectively.

We close these gaps and resolve the dependence on ε, improving the lower bounds to match the upper bounds for all parameters n, β, and ε (up to logarithmic factors). Formally, we have:

Theorem 1.1 (For-Each Cut Sketch for Balanced Graphs).—*Let* $\beta \geq 1$ *and* $0 < \varepsilon < 1$. *Assume* $\sqrt{\beta}/\varepsilon \leq n/2$. *Any* $(1 \pm \varepsilon)$ *for-each cut sketching algorithm for* β*-balanced* n*-node graphs must output* $\widetilde{\Omega}(n\sqrt{\beta}/\varepsilon)$ *bits.*

Theorem 1.2 (For-All Cut Sketch for Balanced Graphs).—*Let* $\beta \geq 1$ *and* $0 < \varepsilon < 1$. *Assume* $\beta/\varepsilon^2 \leq n/2$. *Any* $(1 \pm \varepsilon)$ *for-all cut sketching algorithm for* β*-balanced* n*-node graphs must output* $\Omega\left(n\beta/\varepsilon^2\right)$ *bits.*

Query Complexity of Min-Cut in the Local Query Model.—We study the problem of $(1 \pm \varepsilon)$-approximating the (undirected) global minimum cut in a local query model, where we can only access the graph via degree, edge, and adjacency queries.

We close the gap on the ε dependence in the query complexity of this problem by proving a tight $\Omega(\min\{m, \frac{m}{\varepsilon^2 k}\})$ lower bound, where m is the number of edges and k is the size of the minimum cut. This improves the previous $\Omega\left(\frac{m}{k}\right)$ lower bound in [ER18]. Formally, we have:

Theorem 1.3 (Approximating Min-Cut using Local Queries).—*Any algorithm that estimates the size of the global minimum cut of a graph* G *up to a* $(1 \pm \varepsilon)$ *factor requires* $\Omega(\min\{m, \frac{m}{\varepsilon^2 k}\})$ *queries in expectation in the local query model, where* m *is the number of edges in* G *and* k *is the size of the minimum cut.*

We also show that with a slight modification, the $\widetilde{O}(\frac{m}{k \ \mathrm{poly}(\varepsilon)})$ query complexity upper bound in [BGMP21] can be improved to $\widetilde{O}\left(\frac{m}{\varepsilon^2 k}\right)$, which implies that our lower bound is tight (up to logarithmic factors).

1.2 Our Techniques

A common technique we use for the different problems is communication complexity games that involve the approximation parameter ε. For example, suppose Alice has a bit string s of length $\left(1/\varepsilon^2\right)$, and she can encode s into a graph G such that, if she sends Bob a $(1 \pm \varepsilon)$

(for-each or for-all) cut sketch to Bob, then Bob can recover a specific bit of s with high constant probability. By communication complexity lower bounds, we know Alice must send $\Omega(1/\varepsilon^2)$ bits to Bob, which gives a lower bound on the size of the cut sketch.

For-Each Cut Sketch Lower Bound.—Let $k = \sqrt{\beta}/\varepsilon$. At a high level, we partition the n nodes into $n/(2k)$ sub-graphs, where each sub-graph is a k-by-k bipartite graph with two parts L and R. We then divide L and R into $\sqrt{\beta}$ disjoint clusters $|L_1| = |L_2| = \ldots = |L_{\sqrt{\beta}}| = 1/\varepsilon$ and $|R_1| = |R_2| = \ldots = |R_{\sqrt{\beta}}| = 1/\varepsilon$. For every cluster pair L_i and R_j, there are a total of $1/\varepsilon^2$ edges. Intuitively, we wish to encode a bit string $s \in \{-1, 1\}^{1/\varepsilon^2}$ into forward edges (left to right) each with weight $\Theta(1)$, and add backward edges (right to left) each with weight $1/\beta$ so that the graph β-balanced. If we could approximately decode this string from a for-each cut sketch, then we would get an $\Omega((n/k) \cdot (\sqrt{\beta})^2 \cdot (1/\varepsilon)^2) = \Omega(n\sqrt{\beta}/\varepsilon)$ lower bound.

However, if we use a simple encoding method [ACK+16, CCPS21] where each bit s_i is encoded into one edge (u, v) (e.g., with weight 1 or 2) and query the edges leaving $S = \{u\} \cup (R \setminus \{v\})$, then the $(k-1)^2 = \Omega(\beta/\varepsilon^2)$ backward edges with weight $1/\beta$ will cause the cut value to be $\Omega(1/\varepsilon^2)$. The $(1 \pm \varepsilon)$ cut sketch will have additive error $\Omega(1/\varepsilon) \gg \Theta(1)$, which will obscure $s_i = \{-1, 1\}$. To address this, we instead encode $1/\varepsilon^2$ bits of information across $1/\varepsilon^2$ edges simultaneously. When we want to decode a specific bit s_i, we query the (directed) cut values between two *carefully designed* subsets $A \in L_i$ and $B \in R_j$. The key idea of our construction is that, although each edge in $A \times B$ is used to encode many bits of s, the encoding of different bits of s is *never too correlated*: while encoding other bits does affect the total weight from A to B, this effect is similar to adding noise which only varies the total weight from A to B by a small amount.

For-All Cut Sketch Lower Bound.—Let $k = \beta/\varepsilon^2$. At a high level, we partition the n nodes into $n/(2k)$ sub-graphs, where each sub-graph is a k-by-k bipartite graph with two parts L and R. Let $L = \{\ell_1, \ldots, \ell_k\}$. We partition R into β disjoint clusters $|R_1| = \ldots = |R_\beta| = 1/\varepsilon^2$. We use edges from ℓ_i to R_j to encode a bit string $s \in \{0, 1\}^{1/\varepsilon^2}$ by setting the weight of each forward edge to 1 or 2, and adding a backward edge of weight $1/\beta$ to balance the graph.

We can show that the following problem requires $\Omega(1/\varepsilon^2)$ bits of communication: Consider $\ell_i \in L$ and a random subset $T \subset R_j$ where $|T| = \frac{|R_j|}{2}$. Let $N(\ell_i)$ denote ℓ_i's neighbors v such that (ℓ_i, v) has weight 2, which is uniformly random if s is uniformly random. The problem is to decide whether $|N(\ell_i) \cap T| \geq \frac{1}{4\varepsilon^2} + \frac{c}{2\varepsilon}$ or $|N(\ell_i) \cap T| \leq \frac{1}{4\varepsilon^2} - \frac{c}{2\varepsilon}$ for a sufficiently small constant $c > 0$. Intuitively, the graph encodes a $(k\beta)$-fold version of this communication problem, which implies an $\Omega((n/k) \cdot k\beta \cdot (1/\varepsilon)^2) = \Omega(n\beta/\varepsilon^2)$ lower bound.

We need to show that Bob can distinguish between the two cases of $|N(\ell_i) \cap T|$ given a for-all cut sketch. However, there are some challenges. The difference between the two cases

is $\Theta(1/\varepsilon)$ while the natural cut to query $S = \{\ell_i\} \cup (R \setminus T)$ has value $\Omega(\beta/\varepsilon^4)$. The $(1 \pm \varepsilon)$ cut sketch will have additive error $\Omega(\beta/\varepsilon^3) \gg \Theta(1/\varepsilon)$, which is too much. To overcome this, note that we have not used the property that the for-all cut sketch preserves all cuts. We make use of the following crucial observation in [ACK+16]: In expectation, roughly half of the nodes $\ell_i \in L$ satisfy $|N(\ell_i) \cap T| \geq \frac{1}{4\varepsilon^2} + \frac{c}{2\varepsilon}$ because c is small. If Bob enumerates all subsets $Q \subset L$ of size $\frac{|L|}{2}$, he will eventually get lucky and find a set Q that contains almost all such nodes. Since there are roughly $\frac{|L|}{2} = \frac{\beta}{2\varepsilon^2}$ such nodes, the (c/ε) bias per node will contribute $\Omega(c\beta/\varepsilon^3)$ in total, which is enough to be detected even under an $O(\beta/\varepsilon^3)$ additive error.

Query Complexity of Min-Cut in the Local Query Model.—We prove our lower bound using communication complexity, but unlike previous work [ER18], we consider the following 2SUM problem [WZ14]: Given $2t$ length-L binary strings $(x^1, x^2, \ldots, x^t)$ and $(y^1, y^2, \ldots, y^t)$, we want to approximate the value of $\sum_{i \in [t]} \mathrm{DISJ}(x^i, y^i)$ up to a $\sqrt{t}$ additive error, with the promise that at least a constant fraction of the (x^i, y^i) satisfy $\mathrm{INT}(x^i, y^i) = \alpha$ while the remaining pairs satisfy $\mathrm{INT}(x^i, y^i) = 0$ or α. Here $\mathrm{INT}(x, y) = \sum_{i=1}^{L} x_i \wedge y_i$ is the number of indices where x and y are both 1, and $\mathrm{DISJ}(x, y)$ is the set-disjointness problem, i.e., $\mathrm{DISJ}(x, y) = 1$ if $\mathrm{INT}(x, y) = 0$ and $\mathrm{DISJ}(x, y) = 0$ otherwise. The parameters L, t, and α will be chosen later.

We construct our graph $G_{x,y}$ based on the vectors x^i and y^i in a way inspired by [ER18]. We then give a careful analysis of the size of the minimum cut of $G_{x,y}$, and show that under certain conditions, the size of the minimum cut is exactly $2\sum_{i \in [t]} \mathrm{INT}(x^i, y^i)$. Consequently, a $(1 \pm \varepsilon)$-approximation of the minimum cut yields an approximation of $\sum_{i \in [t]} \mathrm{DISJ}(x^i, y^i)$ up to a $\sqrt{\varepsilon}$ additive error, which implies the desired lower bound.

2 Preliminaries

Let $G = (V, E, w)$ be a weighted (directed) graph with n vertices and m edges, where each edge $e \in E$ has weight $w_e \geq 0$. We write $G = (V, E)$ if G is unweighted and leave out w. For two sets of nodes $S, T \subseteq V$, let $E(S, T) = \{(u, v) \in E : u \in S, v \in T\}$ denote the set of edges from S to T. Let $w(S, T) = \sum_{e \in E(S,T)} w_e$ denote the total weight of edges from S to T. For a node $u \in V$ and a set of nodes $S \subseteq V$, we write $w(u, S)$ for $w(\{u\}, S)$.

We write $[n]$ for $\{1, \ldots, n\}$. We use $\mathbf{1}$ to denote the all-ones vector. For a vector v, we write $\|v\|_2$ and $\|v\|_\infty$ for the ℓ_2 and ℓ_∞ norm of x respectively. For two vectors $u, v \in \mathbb{R}^n$, let $u \otimes v \in \mathbb{R}^{n^2}$ be the tensor product of u and v. Given a matrix A, we use A_i to denote the i-th row of A.

Directed Cut Sketches.

We start with the definitions of β-balanced graphs, for-all and for-each cut sketches [BK96, ST11, ACK+16, CCPS21].

We say a directed graph is balanced if all cuts have similar values in both directions.

Definition 2.1 (β-Balanced Graphs).

A strongly connected directed graph $G = (V, E, w)$ is β-balanced if, for all $\emptyset \subset S \subset V$, it holds that $w(S, V \setminus S) \leq \beta \cdot w(V \setminus S, S)$.

We say $\mathrm{sk}(G)$ is a for-all cut sketch if the value of all cuts can be approximately recovered from it. Note that $\mathrm{sk}(G)$ is not necessarily a graph and can be an arbitrary data structure.

Definition 2.2 (For-All Cut Sketch).

Let $0 < \varepsilon < 1$. We say $\mathcal{A}$ is a $(1 \pm \varepsilon)$ for-all cut sketching algorithm if there exists a recovering algorithm f such that, given a directed graph $G = (V, E, w)$ as input, $\mathcal{A}$ can output a sketch $\mathrm{sk}(G)$ such that, with probability at least $2/3$, for all $\emptyset \subset S \subset V$:

$$(1-\varepsilon) \cdot w(S, V \setminus S) \leq f(S, \mathrm{sk}(G)) \leq (1+\varepsilon) \cdot w(S, V \setminus S).$$

Another notion of cut approximation is that of a "for-each" cut sketch, which requires that the value of each individual cut is preserved with high constant probability, rather than approximating the values of all cuts simultaneously.

Definition 2.3 (For-Each Cut Sketch).

Let $0 < \varepsilon < 1$. We say $\mathcal{A}$ is a $(1 \pm \varepsilon)$ for-each cut sketching algorithm if there exists a recovering algorithm f such that, given a directed graph $G = (V, E, w)$ as input, $\mathcal{A}$ can output a sketch $\mathrm{sk}(G)$ such that, for each $\emptyset \subset S \subset V$, with probability at least $2/3$,

$$(1-\varepsilon) \cdot w(S, V \setminus S) \leq f(S, \mathrm{sk}(G)) \leq (1+\varepsilon) \cdot w(S, V \setminus S).$$

In Definitions 2.2 and 2.3, the sketching algorithm $\mathcal{A}$ and the recovering algorithm f can be randomized, and the probability is over the randomness in $\mathcal{A}$ and f.

3 For-Each Cut Sketch

In this section, we prove an $\Omega(n\sqrt{\beta}/\varepsilon)$ lower bound on the output size of $(1 \pm \varepsilon)$ for-each cut sketching algorithms (Definition 2.3).

Theorem 1.1 (For-Each Cut Sketch for Balanced Graphs).

Let $\beta \geq 1$ and $0 < \varepsilon < 1$. Assume $\sqrt{\beta}/\varepsilon \lesssim n/2$. Any $(1 \pm \varepsilon)$ for-each cut sketching algorithm for β-balanced n-node graphs must output $\widetilde{\Omega}(n\sqrt{\beta}/\varepsilon)$ bits.

Our result uses the following communication complexity lower bound for a variant of the Index problem, where Alice and Bob's inputs are random.

Lemma 3.1 ([KNR01]).

Suppose Alice has a uniformly random string $s \in \{-1, 1\}^n$ *and Bob has a uniformly random index* $i \in [n]$. *If Alice sends a single (possibly randomized) message to Bob, and Bob can recover* s_i *with probability at least* 2/3 *(over the randomness in the input and their protocol), then Alice must send* $\Omega(n)$ *bits to Bob.*

Our lower-bound construction relies on the following technical lemma.

Lemma 3.2.

For any integer $k \geq 1$, *there exists a matrix* $M \in \{-1, 1\}^{\left(2^k-1\right)^2 \times 2^{2k}}$ *such that*:

1. $\langle M_t, \mathbf{1}\rangle = 0$ *for all* $t \in [\left(2^k-1\right)^2]$.
2. $\langle M_t, M_{t'}\rangle = 0$ *for all* $1 \leq t < t' \leq \left(2^k-1\right)^2$.
3. *For all* $t \in [\left(2^k-1\right)^2]$, *the* t*-th row of* M *can be written as* $M_t = u \otimes v$ *where* $u, v \in \{-1, 1\}^{2^k}$ *and* $\langle u, \mathbf{1}\rangle = \langle v, \mathbf{1}\rangle = 0$.

Proof.—Our construction is based on the Hadamard matrix $H = H_{2^k} \in \{-1, 1\}^{2^k \times 2^k}$. Recall that the first row of H is the all-ones vector and that $\langle H_i, H_j\rangle = 0$ for all $i \neq j$. For every $2 \leq i, j \leq 2^k$, we add $H_i \otimes H_j \in \{-1, 1\}^{2^{2k}}$ as a row of M, so M has $\left(2^k-1\right)^2$ rows.

Condition (3) holds because $\langle H_i, \mathbf{1}\rangle = \langle H_j, \mathbf{1}\rangle = 0$ for all $i, j \geq 2$. For Conditions (1) and (2), note that for any vectors u, v, w, and z, we have $\langle u \otimes v, w \otimes z\rangle = \langle u, w\rangle\langle v, z\rangle$. Using this fact, Condition (1) holds because $\langle M_t, \mathbf{1}\rangle = \langle H_i \otimes H_j, \mathbf{1} \otimes \mathbf{1}\rangle = \langle H_i, \mathbf{1}\rangle\langle H_j, \mathbf{1}\rangle = 0$, and Condition (2) holds because $(i, j) \neq (i', j')$ and thus $\langle M_t, M_{t'}\rangle = \langle H_i \otimes H_j, H_{i'} \otimes H_{j'}\rangle = \langle H_i, H_{i'}\rangle\langle H_j, H_{j'}\rangle = 0$. □

We first prove a lower bound for the special case $n = \Theta(\sqrt{\beta}/\varepsilon)$. Our proof for this special case introduces important building blocks for proving the general case $n = \Omega(\sqrt{\beta}/\varepsilon)$.

Lemma 3.3.

Suppose $n = \Theta(\sqrt{\beta}/\varepsilon)$. *Any* $(1 \pm \varepsilon)$ *for-each cut sketching algorithm for* β*-balanced* n*-node graphs must output* $\widetilde{\Omega}(n\sqrt{\beta}/\varepsilon) = \widetilde{\Omega}\left(\beta/\varepsilon^2\right)$ *bits.*

At a high level, we reduce the Index problem (Lemma 3.1) to the for-each cut sketching problem. Given Alice's string s, we construct a graph G to encode s, such that Bob can recover any single bit in s by querying $O(1)$ cut values of G. Our lower bound (Lemma 3.3) then follows from the communication complexity lower bound of the Index problem (Lemma 3.1), because Alice can run a for-each cut sketching algorithm and send the cut sketch to Bob, and Bob can successfully recover the $O(1)$ cut values with high constant probability.

Proof of Lemma 3.3.—We reduce from the Index problem. Let $s \in \{-1, 1\}^{\beta(\frac{1}{\varepsilon}-1)^2}$ denote Alice's random string.

Construction of G.

We construct a directed complete bipartite graph G to encode s. Let L and R denote the left and right nodes of G, where $|L| = |R| = \sqrt{\beta}/\varepsilon$. We partition L into $\sqrt{\beta}$ disjoint blocks $L_1, \ldots, L_{\sqrt{\beta}}$ of equal size, and similarly partition R into $R_1, \ldots, R_{\sqrt{\beta}}$. We divide s into β disjoint strings $s_{i,j} \in \{-1, 1\}^{(\frac{1}{\varepsilon}-1)^2}$ of the same length. We will encode $s_{i,j}$ using the edges from L_i to R_j. Note that the encoding of each $s_{i,j}$ is independent since $E(L_i, R_j) \cap E(L_{i'}, R_{j'}) = \emptyset$ for $(i, j) \neq (i', j')$.

We fix i and j and focus on the encoding of $s_{i,j}$. Note that $|L_i| = |R_j| = 1/\varepsilon$. We refer to the edges from L_i to R_j as *forward* edges and the edges from R_j to L_i as *backward* edges. Let $w \in \mathbb{R}^{1/\varepsilon^2}$ denote the weights of the forward edges, which we will choose soon. Every backward edge has weight $1/\beta$.

Let $z = s_{i,j} \in \{-1, 1\}^{(\frac{1}{\varepsilon}-1)^2}$. Assume w.l.o.g. that $1/\varepsilon = 2^k$ for some integer k. Consider the vector $x = \sum_{t=1}^{(\frac{1}{\varepsilon}-1)^2} z_t M_t \in \mathbb{R}^{1/\varepsilon^2}$ where M is the matrix in Lemma 3.2 with $2^k = 1/\varepsilon$. Because $z_t \in \{-1, 1\}$ is uniformly random, each coordinate of x is a sum of $O(1/\varepsilon^2)$ i.i.d. random variables of value ±1. By the Chernoff bound and the union bound, we know that with probability at least 99/100, $\|x\|_\infty \leq c_1 \ln(1/\varepsilon)/\varepsilon$ for some constant $c_1 > 0$. If this happens, we set $w = \varepsilon x + 2c_1 \ln(1/\varepsilon)1$, so that each entry of w is between $c_1 \ln(1/\varepsilon)$ and $3c_1 \ln(1/\varepsilon)$. Otherwise, we set $w = 2c_1 \ln(1/\varepsilon)1$ to indicate that the encoding failed.

We first verify that G is $O(\beta \log(1/\varepsilon))$-balanced. This is because every edge has a reverse edge with similar weight: For every $u \in L$ and $v \in R$, the edge (u, v) has weight $\Theta(\log(1/\varepsilon))$, while the edge (v, u) has weight $1/\beta$.

We will show that given a $(1 \pm \frac{c_2\varepsilon}{\ln(1/\varepsilon)})$ cut sketch for some constant $c_2 > 0$, Bob can recover a specific bit of z using 4 cut queries. By Lemma 3.1, this implies an $\Omega(\beta/\varepsilon^2) = \widetilde{\Omega}(\beta'/\varepsilon'^2)$ lower bound for cut sketching algorithms for $\beta' = O(\beta \log(1/\varepsilon))$ and $\varepsilon' = c_2\varepsilon/\ln(1/\varepsilon)$.

Recovering a bit in s from a for-each cut sketch of G.

Suppose Bob wants to recover a specific bit of s, which belongs to the substring $z = s_{i,j}$ and has an index t in z. We assume that z is successfully encoded by the subgraph between L_i and R_j.

For simplicity, we index the nodes in L_i as $1, \ldots, (1/\varepsilon)$ and similarly for R_j. We index the forward edges (u, v) in alphabetical order, first by $u \in L_i$ and then by $v \in R_j$. Under this

notation, $\langle w, \mathbf{1}_A \otimes \mathbf{1}_B \rangle$ gives the total weight $w(A, B)$ of forward edges from A to B, where $\mathbf{1}_A, \mathbf{1}_B \in \{0, 1\}^{1/\varepsilon}$ are the indicator vectors of $A \subset L_i$ and $B \subset R_j$.

The crucial observation is that, given a cut sketch of G, Bob can approximate $\langle w, M_t \rangle$ using 4 cut queries. By Lemma 3.2, $M_t = h_A \otimes h_B$ for some $h_A, h_B \in \{-1, 1\}^{1/\varepsilon}$. Let $A \subset L_i$ be the set of nodes $u \in L_i$ with $h_A(u) = 1$. Let $B \subset R_i$ be the set of nodes $v \in R_j$ with $h_B(v) = 1$. Let $\overline{A} = L_i \setminus A$ and $\overline{B} = R_j \setminus B$.

$$\begin{aligned}\langle w, M_t \rangle &= \langle w, h_A \otimes h_B \rangle = \langle w, (\mathbf{1}_A - \mathbf{1}_{\overline{A}}) \otimes (\mathbf{1}_B - \mathbf{1}_{\overline{B}}) \rangle \\ &= w(A, B) - w(\overline{A}, B) - w(A, \overline{B}) + w(\overline{A}, \overline{B}).\end{aligned}$$

To approximate the value of $w(A, B)$ (and similarly $w(\overline{A}, B)$, $w(A, \overline{B})$, $w(\overline{A}, \overline{B})$), Bob can query $w(S, V \setminus S)$ for $S = A \cup (R \setminus B)$. Consider the edges from S to $(V \setminus S)$: the forward edges are from A to B, each with weight $\Theta(\log(1/\varepsilon))$; and the backward edges are from $(R \setminus B)$ to $(L \setminus A)$, each with weight $1/\beta$. See Figure 1 as an example.

By Lemma 3.2, $\langle h_A, \mathbf{1} \rangle = \langle h_B, \mathbf{1} \rangle = 0$, so $|A| = |B| = \frac{|L_i|}{2} = \frac{|R_j|}{2} = \frac{1}{2\varepsilon}$. The total weight of the forward edges is $\Theta\left(\log(1/\varepsilon)/\varepsilon^2\right)$, and the total weight of the backward edges is $(\frac{\sqrt{\beta}}{\varepsilon} - \frac{1}{2\varepsilon})^2 \frac{1}{\beta} = \Theta\left(1/\varepsilon^2\right)$, so the cut value $w(S, V \setminus S)$ is $\Theta\left(\log(1/\varepsilon)/\varepsilon^2\right)$. Given a $(1 \pm \frac{c_2\varepsilon}{\ln(1/\varepsilon)})$ for-each cut sketch, Bob can obtain a $(1 \pm \frac{c_2\varepsilon}{\log(1/\varepsilon)})$ multiplicative approximation of $w(S, V \setminus S)$, which has $O(c_2/\varepsilon)$ additive error. After subtracting the total weight of backward edges, which is fixed, Bob has an estimate of $w(A, B)$ with $O(c_2/\varepsilon)$ additive error. Consequently, Bob can approximate $\langle w, M_t \rangle$ with $O(c_2/\varepsilon)$ additive error using 4 cut queries.

Now consider $\langle w, M_t \rangle$. By Lemma 3.2, $\langle M_t, \mathbf{1} \rangle = 0$ and the rows of M are orthogonal,

$$\langle w, M_t \rangle = \langle \varepsilon x, M_t \rangle = \varepsilon \left\langle \sum_{t'} z_{t'} M_{t'}, M_t \right\rangle = \varepsilon z_t \|M_t\|_2^2 = \frac{z_t}{\varepsilon}.$$

We can see that, for a sufficiently small universal constant c_2, Bob can distinguish whether $z_t = 1$ or $z_t = -1$ based on an $O(c_2/\varepsilon)$ additive approximation of $\langle w, M_t \rangle$.

Bob's success probability is at least 0.95, because the encoding of z fails with probability at most 0.01, and each of the 4 cut queries fails with probability at most 0.01.[2] □

We next consider the case with general values of n, β, and ε, and prove Theorem 1.1.

[2]The success probability of a cut query given a for-each cut sketch (Definition 2.3) can be boosted from 2/3 to 99/100, e.g., by running the sketching and recovering algorithms $O(1)$ times and taking the median. This increases the length of Alice's message by a constant factor, which does not affect our asymptotic lower bound.

Proof of Theorem 1.1.—Let $k = \sqrt{\beta}/\varepsilon$. We assume w.l.o.g. that k is an integer, n is a multiple of k, and $(1/\varepsilon)$ is a power of 2. Suppose Alice has a random string $s \in \{-1, 1\}^{\Omega(nk)}$. We will show that s can be encoded into a graph G such that

- **i.** G has n nodes and is $O(\beta \log(1/\varepsilon))$-balanced, and
- **ii.** Given a $(1 \pm \frac{c_2\varepsilon}{\ln(1/\varepsilon)})$ for-each cut sketch of G and an index q, where $c_2 > 0$ is a sufficiently small universal constant, Bob can recover s_q with probability at least 2/3.

Consequently, by Lemma 3.1, any for-each cut sketching algorithm must output $\Omega(nk) = \Omega(n\sqrt{\beta}/\varepsilon) = \widetilde{\Omega}(n\sqrt{\beta'}/\varepsilon')$ bits for $\beta' = O(\beta \log(1/\varepsilon))$ and $\varepsilon' = c_2\varepsilon/\ln(1/\varepsilon)$.

We first describe the construction of G. We partition the n nodes into $\ell = n/k \geq 2$ disjoint sets $V_1, \ldots, V_\ell$, each containing k nodes. Let s be Alice's random string with length $\beta\left(\frac{1}{\varepsilon} - 1\right)^2(\ell - 1) = \Omega\left(k^2\ell\right) = \Omega(nk)$. We partition s into $(\ell - 1)$ strings $(s_i)_{i=1}^{\ell-1}$, with k^2 bits in each substring. We then follow the same procedure as in Lemma 3.3 to encode s_i into a complete bipartite graph between V_i and V_{i+1}. Notice that we have $|s_i| = \beta\left(\frac{1}{\varepsilon} - 1\right)^2$ and $|V_i| = |V_{i+1}| = \sqrt{\beta}/\varepsilon$, which is the same setting as in Lemma 3.3.

We can verify that G is $O(\beta \log(1/\varepsilon))$-balanced. This is because every edge e has a reverse edge whose weight is at most $O(\beta \log(1/\varepsilon))$ times the weight of e. For every $u \in V_i$ and $v \in V_{i+1}$, the edge (u, v) has weight $\Theta(\log(1/\varepsilon))$, while the edge (v, u) has weight $1/\beta$.

We next show that Bob can recover the q-th bit of s. Suppose Bob's index q belongs to the substring s_i which is encoded by the subgraph between V_i and V_{i+1}. Similar to the proof of Lemma 3.3, Bob only needs to approximate $w(A, B)$ for 4 pairs of (A, B) with $O(1/\varepsilon)$ additive error, where $\subset V_i$, $B \subset V_{i+1}$, and $|A| = |B| = \frac{1}{2\varepsilon}$. To achieve this, Bob can query the cut value $w(S, V \setminus S)$ for $S = A \cup (V_{i+1} \setminus B) \bigcup_{j=i+2}^{\ell} V_j$. The edges from S to $(V \setminus S)$ are:

- $\frac{1}{4\varepsilon^2}$ forward edges from A to B, each with weight $\Theta(\log(1/\varepsilon))$.
- $\left(\frac{\sqrt{\beta}}{\varepsilon} - \frac{1}{2\varepsilon}\right)^2$ backward edges from $(V_{i+1} \setminus B)$ to $(V_i \setminus A)$, each with weight $\frac{1}{\beta}$.
- $\frac{\sqrt{\beta}}{2\varepsilon^2}$ backward edges from A to V_{i-1} when $i \geq 2$, each with weight $\frac{1}{\beta}$.

The cut value $w(S, V \setminus S)$ is $\Theta(\log(1/\varepsilon)/\varepsilon^2)$. Consequently, given a $(1 \pm \frac{c_2\varepsilon}{\ln(1/\varepsilon)})$ for-each cut sketch, after subtracting the fixed weight of the backward edges, Bob can approximate $w(A, B)$ with $O(c_2/\varepsilon)$ additive error. Similar to the proof of Lemma 3.3, for sufficiently small constant $c_2 > 0$, repeating this process for 4 different pairs of (A, B) will allow Bob to recover $s_q \in \{-1, 1\}$. □

4 For-All Cut Sketch

In this section, we prove an $\Omega(n\beta/\varepsilon^2)$ lower bound on the output size of $(1 \pm \varepsilon)$ for-all cut sketching algorithms (Definition 2.2).

Theorem 1.2 (For-All Cut Sketch for Balanced Graphs).

Let $\beta \geq 1$ and $0 < \varepsilon < 1$. Assume $\beta/\varepsilon^2 \leq n/2$. Any $(1 \pm \varepsilon)$ for-all cut sketching algorithm for β-balanced n-node graphs must output $\Omega(n\beta/\varepsilon^2)$ bits.

Our proof is inspired by [ACK+16] and uses the following communication complexity lower bound for an n-fold version of the Gap-Hamming problem.

Lemma 4.1 ([ACK+16]).

Consider the following distributional communication problem: Alice has h strings $s_1, \ldots, s_h \in \{0, 1\}^{1/\varepsilon^2}$ of Hamming weight $\frac{1}{2\varepsilon^2}$. Bob has an index $i \in [h]$ and a string $t \in \{0, 1\}^{1/\varepsilon^2}$ of Hamming weight $\frac{1}{2\varepsilon^2}$, drawn as follows:

1. *i is chosen uniformly at random;*
2. *every $s_{i'}$ for $i' \neq i$ is chosen uniformly at random;*
3. *s_i and t are chosen uniformly at random, conditioned on their Hamming distance $\Delta(s_i, t)$ being, with equal probability, either $\geq \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$ or $\leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$ for some universal constant $c > 0$.*

Consider a (possibly randomized) one-way protocol, in which Alice sends Bob a message, and Bob then 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 must send $\Omega(h/\varepsilon^2)$ bits to Bob.

Before proving Theorem 1.2, we first consider the special case $n = \Theta(\beta/\varepsilon^2)$.

Lemma 4.2.

Suppose $n = \Theta(\beta/\varepsilon^2)$. Any $(1 \pm \varepsilon)$ for-all cut sketching algorithm for β-balanced n-node graphs must output $\Omega(n\beta/\varepsilon^2) = \Omega(\beta^2/\varepsilon^4)$ bits.

We reduce the distributional Gap-Hamming problem (Lemma 4.1) to the for-all cut sketching problem. Suppose Alice has h strings $s_1, s_2, \ldots, s_h \in \{0, 1\}^{1/\varepsilon^2}$ where $h = \beta^2/\varepsilon^2$, and Bob has an index $i \in [h]$ and a string $t \in \{0, 1\}^{1/\varepsilon^2}$. We construct a graph G to encode $s_1, s_2, \ldots, s_h$, such that given a for-all cut sketch of G, Bob can determine whether $\Delta(s_i, t) \geq \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$ or $\Delta(s_i, t) \leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$ with high constant probability. Our lower bound then follows from Lemma 4.1.

Construction of G.

We construct a directed complete bipartite graph G. Let L and R denote the left and right nodes of G, where $|L| = |R| = \beta/\varepsilon^2$. We partition R into β disjoint sets with $|R_1| = \ldots = |R_\beta| = 1/\varepsilon^2$.

Consider the distributional Gap-Hamming problem in Lemma 4.1 with $h = \beta^2/\varepsilon^2$. We re-index Alice's $\left(\beta^2/\varepsilon^2\right)$ strings as $s_{i,j}$, where $i \in \left[\beta/\varepsilon^2\right]$ and $j \in [\beta]$. Let $\ell_1, \ell_2, \ldots, \ell_{\beta/\varepsilon^2}$ be the nodes in L. We encode $s_{i,j} \in \{0, 1\}^{1/\varepsilon^2}$ using the edges from ℓ_i to R_j: For node ℓ_i and the v-th node in R_j, the *forward* edge (ℓ_i, v) has weight $s_{i,j}(v) + 1$, and the *backward* edge (v, ℓ_i) has weight $1/\beta$. Note that the encoding of each $s_{i,j}$ is independent since $E(\ell_i, R_j) \cap E(\ell_{i'}, R_{j'}) = \varnothing$ for $(i, j) \neq (i', j')$.

Determining $\Delta(s_{i,j}, t)$ from a for-all cut sketch of G.

Suppose Bob's input (after re-indexing) is $1 \le i \le \beta/\varepsilon^2$, $1 \le j \le \beta$, and $t \in \{0, 1\}^{1/\varepsilon^2}$. Bob wants to decide whether $\Delta(s_{i,j}, t) \ge \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$ or $\Delta(s_{i,j}, t) \le \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$.

Let $N(\ell_i)$ denote the set of nodes $v \in R_j$ where the forward edge (ℓ_i, v) has weight 2, which corresponds to the positions of 1 in $s_{i,j}$. Let T be the set of nodes $v \in R_j$ such that $t(v) = 1$.

$$\Delta(s_{i,j}, t) = |N(\ell_i) \setminus T| + |T \setminus N(\ell_i)| = |N(\ell_i)| + |T| - 2|N(\ell_i) \cap T| = \frac{1}{\varepsilon^2} - 2|N(\ell_i) \cap T|.$$

Hence, to determine whether $\Delta(s_{i,j}, t) \le \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$ or $\Delta(s_{i,j}, t) \ge \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$, Bob only needs to decide whether $|N(\ell_i) \cap T| \ge \frac{1}{4\varepsilon^2} + \frac{c}{2\varepsilon}$ or $|N(\ell_i) \cap T| \le \frac{1}{4\varepsilon^2} - \frac{c}{2\varepsilon}$.

Let $S = \{\ell_i\} \cup (R \setminus T)$. The cut $w(S, V \setminus S)$ consists of forward edges from ℓ_i to T and backward edges from $(R \setminus T)$ to $(L \setminus \{\ell_i\})$. Ideally, if Bob knows $w(S, V \setminus S)$, he can subtract the weight of backward edges to obtain $w(\ell_i, T) = \frac{1}{\varepsilon^2} + |N(\ell_i) \cap T|$ and recover $|N(\ell_i) \cap T|$. However, Bob can only get a $(1 \pm \varepsilon)$-approximation of $w(S, V \setminus S)$, which may have $\Theta\left(\beta/\varepsilon^3\right)$ additive error because $w(S, V \setminus S) = \Theta\left(\beta/\varepsilon^4\right)$. With this much error, Bob cannot distinguish between the two cases.

To overcome this issue, we follow the idea of [ACK+16]. Intuitively, when c is small, roughly half of $\ell_i \in L$ satisfy $|N(\ell_i) \cap T| \ge \frac{1}{4\varepsilon^2} + \frac{c}{2\varepsilon}$. By enumerating all subsets $Q \subset L$ of size $|Q| = \frac{|L|}{2}$, Bob can find a set Q such that most nodes $v \in Q$ satisfy $|N(\ell_i) \cap T| \ge \frac{1}{4\varepsilon^2} + \frac{c}{2\varepsilon}$. Since $|Q| = \frac{\beta}{2\varepsilon^2}$, the $\frac{c}{2\varepsilon}$ bias per node adds up to roughly $\frac{c\beta}{2\varepsilon^3}$, which can be detected even with $\Theta\left(\beta/\varepsilon^3\right)$ error.

To prove Lemma 4.2, we need the following two technical lemmas, which are essentially proved in [ACK+16].

Lemma 4.3 (Claim 3.5 in [ACK+16]).

Let $c > 0$ *and* $\frac{\sqrt{\beta}}{\varepsilon} \geq \frac{10}{c}$. *Consider the following sets:*

$$L_{\text{high}} = \{\ell_i \in L : |N(\ell_i) \cap T| \geq \frac{1}{4\varepsilon^2} + \frac{c}{2\varepsilon}\}, \text{ and}$$
$$L_{\text{low}} = \{\ell_i \in L : |N(\ell_i) \cap T| \leq \frac{1}{4\varepsilon^2} - \frac{c}{2\varepsilon}\}.$$

With probability at least 0.98, *we have* $\frac{1}{2} - 10c \leq \frac{|L_{\text{high}}|}{|L|} \leq \frac{1}{2}$ *and* $\frac{1}{2} - 10c \leq \frac{|L_{\text{low}}|}{|L|} \leq \frac{1}{2}$.

Lemma 4.4 (Lemma 3.4 in [ACK+16]).

Let $c_1 > 0$ *be a sufficiently small universal constant. Suppose one can approximate* $w(U, T)$ *with additive error* $c_1\beta/\varepsilon^3$ *for every* $U \subset L$ *with* $|U| = \frac{|L|}{2}$. *Let* $Q \subset L$ *be the subset with the highest (approximate) cut value. Then, with probability at least* 0.96, *we have* $\frac{|L_{\text{high}} \cap Q|}{|L_{\text{high}}|} \geq \frac{4}{5}$.

We are now ready to prove Lemma 4.2.

Proof of Lemma 4.2.—We reduce from the distributional Gap-Hamming problem (Lemma 4.1) with $h = \beta^2/\varepsilon^2$. We re-index Alice's h strings as $s_{i,j}$, where $i \in \left[\beta/\varepsilon^2\right]$ and $j \in [\beta]$.

We construct a directed bipartite graph G with two parts L and R, where $|L| = |R| = \beta/\varepsilon^2$. Let $L = \{\ell_1, \ell_2, \ldots, \ell_{\beta/\varepsilon^2}\}$. We partition R into β disjoint sets with $|R_1| = \ldots = |R_\beta| = 1/\varepsilon^2$. We encode $s_{i,j} \in \{0, 1\}^{1/\varepsilon^2}$ using the edges from ℓ_i to R_j: For node ℓ_i and the v-th node in R_j, the *forward* edge (ℓ_i, v) has weight $s_{i,j}(v) + 1$, and the *backward* edge (v, ℓ_i) has weight $1/\beta$.

Note that G is (2β)-balanced. We will show that given a $(1 \pm c_2\varepsilon)$ for-all cut sketch of G for some constant $c_2 > 0$, Bob can decide whether $\Delta(s_{i,j}, t) \geq \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$ or $\Delta(s_{i,j}, t) \leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$ with probability at least 2/3. Consequently, by Lemma 4.1, any for-all cut sketching algorithm must output $\Omega(h/\varepsilon^2) = \Omega(\beta^2/\varepsilon^4) = \Omega(\beta'^2/\varepsilon'^4)$ bits for $\beta' = 2\beta$ and $\varepsilon' = c_2\varepsilon$.

Bob enumerates every $U \subset L$ with $|U| = \frac{|L|}{2} = \frac{\beta}{2\varepsilon^2}$ and uses the cut sketch to approximate $w(U, T)$, where $T \subset R_j$ corresponds to the positions of 1 in Bob's string t and $|T| = \frac{1}{2\varepsilon^2}$. Let $S = U \cup (R \setminus T)$. The cut $(S, V \setminus S)$ has $\frac{\beta}{4\varepsilon^4}$ forward edges from U to T with weights 1 or 2, and $\left(\frac{\beta}{\varepsilon^2} - \frac{1}{2\varepsilon^2}\right)\left(\frac{\beta}{2\varepsilon^2}\right) = O\left(\frac{\beta^2}{\varepsilon^4}\right)$ backward edges from $(R \setminus T)$ to $(L \setminus U)$ with weight

$\frac{1}{\beta}$. The total weight of these edges is $O(\beta/\varepsilon^4)$. Therefore, given a $(1 \pm c_2\varepsilon)$ for-all cut sketch, Bob can subtract the fixed weight of the backward edges and approximate $w(U, T)$ with additive error $O(c_2\beta/\varepsilon^3)$. When c_2 is sufficiently small, this additive error is at most $c_1\beta/\varepsilon^3$. By Lemma 4.4, Bob can find $Q \subset L$ with $|Q| = \frac{|L|}{2}$ such that $\frac{|L_{\text{high}} \cap Q|}{|L_{\text{high}}|} \geq \frac{4}{5}$. Finally, if $\ell_i \in Q$, Bob decides $|N(\ell_i) \cap T| \geq \frac{1}{4\varepsilon^2} + \frac{c}{2\varepsilon}$ and $\Delta(s_{i,j}, t) \leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$; and if $\ell_i \notin Q$, Bob decides $\Delta(s_{i,j}, t) \geq \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$.

Suppose Bob's index is (i, j). Notice that Bob uses j to determine which R_j to look at, but does not use any information about i. Therefore, when $\Delta(s_{i,j}, t) \leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$ and $|N(\ell_i) \cap T| \geq \frac{1}{4\varepsilon^2} + \frac{c}{2\varepsilon}$,

$$\mathbf{Pr}[i \in Q] = \frac{|L_{\text{high}} \cap Q|}{|L_{\text{high}}|} \geq \frac{4}{5}.$$

Conversely, when $\Delta(s_{i,j}, t) \leq \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$ and $|N(\ell_i) \cap T| \leq \frac{1}{4\varepsilon^2} - \frac{c}{2\varepsilon}$, because $L_{\text{low}} \cap L_{\text{high}} = \emptyset$,

$$\mathbf{Pr}[i \notin Q] = \frac{|L_{\text{low}} \setminus Q|}{|L_{\text{low}}|} = \frac{|L_{\text{low}}| - |L_{\text{low}} \cap Q|}{|L_{\text{low}}|} \geq \frac{|L_{\text{low}}| - \frac{1}{5}|L_{\text{high}}|}{|L_{\text{low}}|} \geq \frac{3}{4}.$$

The last inequality holds because $|L_{\text{low}}| \geq 0.4|L|$ and $|L_{\text{high}}| \leq 0.5|L|$ by Lemma 4.3 when $c \leq 0.1$.

We analyze Bob's success probability. Lemma 4.3 fails with probability at most 0.02, Lemma 4.4 fails with probability at most 0.04, and the for-all cut sketch fails with probability at most 0.01[3] If they all succeed, Bob's probability of answering correctly is at least $\min\left(\frac{|L_{\text{high}} \cap Q|}{|L_{\text{high}}|}, \frac{|L_{\text{low}} \setminus Q|}{|L_{\text{low}}|}\right) \geq \frac{3}{4}$. Bob's overall fail probability is at most 0.02 + 0.04 + 0.01 + 0.25 < 1/3. □

We next consider the case with general values of n, β, and ε, and prove Theorem 1.2.

Proof of Theorem 1.2.—Let $k = \beta/\varepsilon^2$. We assume w.l.o.g. that k is an integer and n is a multiple of k. We reduce from the distributional Gap-Hamming problem in Lemma 4.1 with $h = \Omega(n\beta)$. We will show that Alice's strings can be encoded into a graph G such that

i. G has n nodes and is (2β)-balanced, and

[3]The probability that a for-all cut sketch (Definition 2.2) preserves all cuts simultaneously can be boosted from 2/3 to 99/100, e.g., by running the sketching and recovering algorithms $O(1)$ times and taking the median. This increases the length of Alice's message by a constant factor, which does not affect our asymptotic lower bound.

ii. After receiving a string t, an index $q \in [h]$, and a $(1 \pm c_2\varepsilon)$ for-all cut sketch of G for some universal constant $c_2 > 0$, Bob can distinguish whether $\Delta(s_q, t) \leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$ or $\Delta(s_q, t) \geq \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$ with probability at least 2/3.

Consequently, by Lemma 4.1, any for-all cut sketching algorithm must output $\Omega\left(h/\varepsilon^2\right) = \Omega\left(n\beta/\varepsilon^2\right) = \Omega\left(n\beta'/\varepsilon'^2\right)$ bits for $\beta' = 2\beta$ and $\varepsilon' = c_2\varepsilon$.

We first describe the construction of G. We partition the n nodes into $\ell = n/k \geq 2$ disjoint sets $V_1, V_2, \cdots, V_\ell$, each containing k nodes. Let $s_1, s_2, \cdots, s_h \in \{0, 1\}^{1/\varepsilon^2}$ be Alice's random strings where $h = (t-1)\left(\beta^2/\varepsilon^2\right) = \Omega\left((n/k)\left(\beta^2/\varepsilon^2\right)\right) = \Omega(n\beta)$. We partition the h strings into $(t-1)$ disjoint sets $S_1, S_2, \cdots, S_{t-1}$, each with $\left(\beta^2/\varepsilon^2\right)$ strings. We then follow the same procedure as in Lemma 4.2 to encode S_i into a complete bipartite graph between V_i and V_{i+1}. Notice that S_i has $\left(\beta^2/\varepsilon^2\right)$ strings and $|V_i| = |V_{i+1}| = k = \beta/\varepsilon^2$, which is the same setting as in Lemma 4.2.

We can verify that G is (2β)-balanced. This is because every edge e has a reverse edge whose weight is at most 2β times the weight of e. For every $u \in V_i$ and $v \in V_{i+1}$, the edge (u, v) has weight 1 or 2, while the edge (v, u) has weight $1/\beta$.

We next show how Bob can distinguish between the two cases. Suppose Bob's index q specifies a string encoded by the subgraph between V_i and V_{i+1}. Similar to the proof of Lemma 4.2, we only need to show that given a $(1 \pm c_2\varepsilon)$ for-all cut sketch, Bob can approximate $w(U, T)$ with additive error $O\left(\beta/\varepsilon^3\right)$ for every $U \subset V_i$ with $|U| = \frac{|V_i|}{2} = \frac{\beta}{2\varepsilon^2}$ and for some $T \subset V_{i+1}$ with $|T| = \frac{1}{2\varepsilon^2}$. To see this, consider $S = U \cup (V_{i+1} \setminus T) \bigcup_{j=i+2}^{t} V_j$. The edges from S to $(V \setminus S)$ are

- $\frac{\beta}{4\varepsilon^4}$ forward edges from U to T, each with weight 1 or 2.
- $\left(\frac{\beta}{\varepsilon^2} - \frac{1}{2\varepsilon^2}\right)\left(\frac{\beta}{2\varepsilon^2}\right)$ backward edges from $(V_{i+1} \setminus T)$ to $(V_i \setminus U)$, each with weight $\frac{1}{\beta}$.
- $\frac{\beta^2}{2\varepsilon^4}$ backward edges from U to V_{i-1} when $i \geq 2$, each with weight $\frac{1}{\beta}$.

The total weight of these edges is $w(S, V \setminus S) = O\left(\beta/\varepsilon^4\right)$. Consequently, given a $(1 \pm c_2\varepsilon)$ cut sketch, Bob can subtract the fixed weight of the backward edges and approximate $w(U, T)$ with $O\left(c_2\varepsilon\left(\beta/\varepsilon^4\right)\right) = O\left(c_2\beta/\varepsilon^3\right)$ additive error. Similar to the proof of Lemma 4.2, for sufficiently small constant $c_2 > 0$, this will allow Bob to distinguish between the two cases $\Delta(s_q, t) \leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$ or $\Delta(s_q, t) \geq \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$ with probability at least 2/3. □

5 Local Query Complexity of Min-Cut

In this section, we present an $\Omega(\min\{m, \frac{m}{\varepsilon^2 k}\})$ lower bound on the query complexity of approximating the global minimum cut of an undirected graph G to a $(1 \pm \varepsilon)$ factor in the local query model. Formally, we have the following theorem.

Theorem 1.3 (Approximating Min-Cut using Local Queries).

Any algorithm that estimates the size of the global minimum cut of a graph G up to a $(1 \pm \varepsilon)$ factor requires $\Omega(\min\{m, \frac{m}{\varepsilon^2 k}\})$ queries in expectation in the local query model, where m is the number of edges in G and k is the size of the minimum cut.

To achieve this, we define a variant of the 2-SUM communication problem in Section 5.1, show a graph construction in Section 5.2, and show that approximating 2-SUM can be reduced to the minimum cut problem using our graph construction in Section 5.3. In Section 5.4, we will show that our lower bound is tight up to logarithmic factors.

5.1 2-SUM Preliminaries

Building off of the work of [WZ14], we define the following variant of the $2-\text{SUM}(t, L, \alpha)$ problem.

Definition 5.1.—*For binary strings $x = (x_1, \ldots, x_L) \in \{0, 1\}^L$ and $y = (y_1, \ldots, y_L) \in \{0, 1\}^L$, let $\text{INT}(x, y) = \sum_{i=1}^{L} x_i \wedge y_i$ denote the number of indices where x and y are both 1. Let $\text{DISJ}(x, y)$ denote whether x and y are disjoint. That is, $\text{DISJ}(x, y) = 1$ if $\text{INT}(x, y) = 0$, and $\text{DISJ}(x, y) = 0$ if $\text{INT}(x, y) \geq 1$.*

Definition 5.2.—*Suppose Alice has binary strings $\left(X^1, \ldots, X^t\right)$ where each string $X^i \in \{0, 1\}^L$ has length L and likewise Bob hast strings $\left(Y^1, \ldots, Y^t\right)$ each of length L. $\text{INT}\left(X^i, Y^i\right)$ is guaranteed to be either 0 or $\alpha \geq 1$ for each pair of strings $\left(X^i, Y^i\right)$. Furthermore, at least $1/1000$ of the $\left(X^i, Y^i\right)$ pairs are guaranteed to satisfy $\text{INT}\left(X^i, Y^i\right) = \alpha$. In the $2-\text{SUM}(t, L, \alpha)$ problem, Alice and Bob want to approximate $\sum_{i \in [t]} \text{DISJ}\left(X^i, Y^i\right)$ up to additive error $\sqrt{t}$ with high constant probability.*

Lemma 5.3.—*To solve $2-\text{SUM}(t, L, 1)$ with high constant probability, the expected number of bits Alice and Bob need to communicate is $\Omega(tL)$.*

Proof.: [WZ14] proved an expected communication complexity of $\Omega(tL)$ for $2-\text{SUM}(t, L, 1)$ without the promise that at least a $1/1000$ fraction of the t string pairs intersect. Adding this promise does not change the communication complexity, because if $\left(X^1, \ldots, X^t\right)$ and $\left(Y^1, \ldots, Y^t\right)$ do not satisfy the promise, we can add a number of new X^i and Y^i to satisfy the promise and later subtract their contribution to approximate $\sum_{i \in [t]} \text{DISJ}\left(X^i, Y^i\right)$ with additive error $\Theta(\sqrt{t})$. □

Theorem 5.4.—*To solve* $2-\mathrm{SUM}(t, L, \alpha)$ *with high constant probability, the expected number of bits Alice and Bob need to communicate is* $\Omega(tL/\alpha)$.

Proof.: Consider an instance of $2-\mathrm{SUM}(t, L/\alpha, 1)$ with Alice's strings $\left(X^1, \ldots, X^t\right)$ and Bob's strings $\left(Y^1, \ldots, Y^t\right)$ each with length L/α. For each of Alice's strings X^i with length L/α, we produce $X^{i,\alpha}$ (with length L) by concatenating α copies of X^i, and likewise we produce $Y^{i,\alpha}$ for each of Bob's strings Y^i. The setup where Alice has strings $\left(X^{1,\alpha}, \ldots, X^{t,\alpha}\right)$ and Bob has strings $\left(Y^{1,\alpha}, \ldots, Y^{t,\alpha}\right)$ is an instance of $2-\mathrm{SUM}(t, L, \alpha)$. From Lemma 5.3, the communication complexity of $2-\mathrm{SUM}(t, L/\alpha, 1)$ is $\Omega(tL/\alpha)$. Thus, the communication complexity of $2-\mathrm{SUM}(t, L, \alpha)$ is $\Omega(tL/\alpha)$. □

5.2 Graph Construction

Inspired by the graph construction from [ER18], given two strings $x, y \in \{0,\ 1\}^N$, we construct a graph $G_{x,y}(V,\ E)$ such that V is partitioned into A, A', B and B', where $|A| = |A'| = |B| = |B'| = \sqrt{N} = \ell$. Note that since $\ell^2 = N$, we can index the bits in x by $x_{i,j}$, where $1 \le i, j \le \ell$. We construct the edges E according to the following rule:

$$\begin{cases} (a_i, b_j'), (b_i, a_j') \in E \text{ if } x_{i,j} = y_{i,j} = 1 \\ (a_i, a_j'), (b_i, b_j') \in E \text{ otherwise} \end{cases}$$

Figure 2 illustrates an example of the graph $G_{x,y}(V,\ E)$ when $x = 000000100$ and $y = 100010100$.

We will show that under certain assumptions about N and $\mathrm{INT}(x,\ y)$, the number of intersections in x, y is twice the size of the minimum cut in $G_{x,y}$.

Lemma 5.5.—*Given* $x,\ y \in \{0,\ 1\}^N$, *if* $\sqrt{N} \ge 3 \cdot \mathrm{INT}(x, y)$, *then* $\mathrm{MINCUT}(G_{x,y}) = 2 \cdot \mathrm{INT}(x,\ y)$.

Proof.: To prove this, we use some properties about γ-connectivity of a graph. A graph is γ-*connected* if at least γ edges must be removed from G to disconnect it. In other words, if a graph G is γ-connected, then $\mathrm{MINCUT}(G) \ge \gamma$. Equivalently, a graph G is γ-connected if for every $u, v \in V$, there are at least γ edge-disjoint paths between u and v. Therefore, given $\mathrm{INT}(x, y) = \gamma$, if we can show that $G_{x,y}$ is 2γ-connected and there exists one cut of size exactly 2γ, then we can show $\mathrm{MINCUT}(G_{x,y}) \ge 2\gamma$. By the construction of the graph, it is easy to see that $\mathrm{CUT}(A \cup A',\ B \cup B')$ has size 2γ, since each intersection of x, y produces two crossing edges in between. Therefore, all we need to show here is that if $\sqrt{N} \ge 3 \cdot \gamma$, then $G_{x,y}$ is 2γ-connected.

Similar to [ER18], we prove this by looking at each pair of $u, v \in V$. Our goal is to show that for every $u, v \in V$, there exist at least 2γ edge-disjoint paths from u to v.

Case 1.— $u, v \in A$ *(or symmetrically* $u, v \in A', B, B'$*). For each pair* $u, v \in A$*, we have that there are at least* $\ell - \gamma$ *distinct common neighbors in* A'*. This is because one intersection at* x_{ij} *and* y_{ij} *implies that the edge* (a_i, a_j') *is not contained in* E*, and would remove at most one common neighbor in* A'*. Since* $\ell = \sqrt{N} \geq 3\gamma$*, we have that there are at least* $\ell - \gamma \geq 2\gamma$ *distinct common neighbors in* A'*, which we denote by* $u_1^{A'}, u_2^{A'}, \ldots, u_{2\gamma}^{A'}$*. Therefore, each path* $u \to u_i^{A'} \to v$ *is edge-disjoint, and we have at least* 2γ *edge-disjoint paths from* u *to* v*, as shown in* Figure 3.

Case 2.— $u \in A$, $v \in A'$ *(or symmetrically* $u \in B$, $v \in B'$*). Since* $\ell - \gamma \geq 2\gamma$*, we have that* v *has at least* 2γ *distinct neighbors in* A*, which we denote by* $u_1^A, u_2^A, \ldots, u_{2\gamma}^A$*. From* Case 1*, we also have that each* u_i^A *has at least* 2γ *distinct common neighbors in* A'*. Therefore, we can choose* $v_1^{A'}, v_2^{A'}, \ldots, v_{2\gamma}^{A'}$ *such that each path* $u \to v_i^{A'} \to u_i^A \to v$ *is edge-disjoint, so we have at least* 2γ *edge-disjoint paths from* u *to* v*, as shown in* Figure 4*. Note that it may be the case where* $u_i^A = u$*. In this case, we can simply take the edge* (u, v) *to be one of the edge-disjoint paths.*

Case 3.— $u \in A$, $v \in B'$ *(or symmetrically* $u \in A'$, $v \in B$*). In this case, we show two sets of edge-disjoint paths, where each set has at least* γ *edge-disjoint paths from* u *to* v*, and the two sets of paths do not overlap. Overall, we have at least* 2γ *edge-disjoint paths.*

The first set of paths S_1 *uses the edges between* A' *and* B*. Let* $(w_1, x_1), (w_2, x_2), \ldots, (w_\gamma, x_\gamma) \in A' \times B$ *be the edges between* A' *and* B*. Each of these edges represents one intersection in* x *and* y*. Therefore, there are exactly* γ *of them. From* Case 2*, we have that for every* w_i*, there are* 2γ *edge-disjoint paths from* u *to* w_i*. Hence, for every* w_i*, we can choose a path from* u *to* w_i *and these* γ *paths are edge-disjoint.* Figure 5 *illustrates the paths* $u \to u_i \to u_i' \to w_i \to x_i$*. By symmetry, we can extend the paths from* x_i *to* v*. This gives us* γ *edge-disjoint paths from* u *to* v*.*

We now consider the second set of paths S_2*. Let*

$$(y_1, z_1), (y_2, z_2), \ldots, (y_\gamma, z_\gamma) \in A \times B'$$

be the distinct edges between A *and* B'*. Once again, it suffices to prove that there are* 2γ *edge-disjoint paths from* u *to* y_i*, since the paths between* v *to* z_i *would be symmetric. From* Case 1*, we have that for every* y_i*, there are at least* 2γ *common neighbors between* y_i *and* u*. Therefore, we can always find distinct* $u_1'', u_2'', \ldots, u_\gamma''$ *such that the paths* $u \to u_i'' \to y_i$ *are edge-disjoint, as shown in* Figure 6*. Once we extend the paths from* z_i *to* v*, we have* γ*-edge disjoint paths in the second set.*

Now we have two sets of paths S_1 *and* S_2*, where both sets have at least* γ *edge-disjoint paths. It remains to show that the paths in* S_1 *and* S_2 *can be edge-disjoint. Observe that the only possible edge overlaps between the paths from* u *to the* w_i *and paths from* u *to the* y_i *are* $u \to u_i''$ *and* $u \to u_i$*, since they are both neighbors of* u*. However, note that what we have shown is that for every* w_i *or* y_i*, there are at least* 2γ *edge-disjoint paths from* u *to* w_i *or* y_i*. Therefore, one can choose* 2γ *edge-disjoint paths from* u *to* w_i *and* y_i *such that* u_i' *and* u_i'' *do not overlap.*

And similarly one can choose 2γ *edge-disjoint paths from* v *to the* z_i *and the* x_i*. Overall, we have* 2γ *edge-disjoint paths from* u *to* v*.*

Case 4.—*$u \in A$, $v \in B$ (or symmetrically $u \in A'$, $v \in B'$). This case is similar to* Case 3, *where we have two edge-disjoint sets* S_1' *and* S_2'*. Consider the set of paths* S_1'*, where we use the edges*

$$(w_1, x_1), (w_2, x_2), \ldots, (w_\gamma, x_\gamma) \in A' \times B.$$

We can construct the paths from u *to* w_i *using the same way as for* S_1 *in* Case 3 (Figure 5). *For the paths from* x_i *to* v*, however, we construct them using the same way as in* S_2 *in* Case 3 (Figure 6). *By connecting these paths, we obtain at least* γ *edge-disjoint paths in* S_1'*. Similarly, we can also construct at least* γ *edge-disjoint paths in* S_2'*, where we use the edges*

$$(y_1, z_1), (y_2, z_2), \ldots, (y_\gamma, z_\gamma) \in A \times B'.$$

We follow the same way of choosing the paths in S_1' *and* S_2' *that are edge-disjoint.* □

5.3 Reducing 2-SUM to MINCUT

In this section, we use the graph constructions in Section 5.2 to reduce the $2-\mathrm{SUM}(t, L, \alpha)$ problem to MINCUT and derive a lower bound on the number of queries in the local query model.

Lemma 5.6.—*Given* $M, \lambda > 0$, *and* $0 < \varepsilon < 1$, *suppose that we have any algorithm* $\mathscr{A}$ *that can estimate the size of the minimum cut of a graph up to a* $(1 \pm \varepsilon)$ *multiplicative factor with* T *expected queries in the local query model. Then there exists an algorithm* $\mathscr{B}$ *that can approximate* $2-\mathrm{SUM}\left(\varepsilon^{-2}, \varepsilon^2 M, \max\left\{\varepsilon^2\lambda, 1\right\}\right)$ *up to an additive error* $\sqrt{\varepsilon^{-2}} = \varepsilon^{-1}$ *using at most* $O(T)$ *bits of communication in expectation given* $\sqrt{M} \geq 3 \max\left\{\lambda, \varepsilon^{-2}\right\}$.

Proof.: We will show that the following algorithm $\mathscr{B}$ satisfies the above conditions:

1. Given Alice's strings $(X^1, \ldots, X^{\varepsilon^{-2}})$ each of length $\varepsilon^2 M$, let x be the concatenation of Alice's strings having total length $\varepsilon^{-2}\left(\varepsilon^2 M\right) = M$. Similarly let $y \in \{0, 1\}^M$ be the concatenation of Bob's strings.
2. Construct a graph $G_{x,y}$ as in Section 5.2 using the above concatenated strings as x, y.
3. Run $\mathscr{A}(G_{x,y})$ and output $\left(\frac{1}{\varepsilon^2} - \frac{\mathscr{A}(G_{x,y})}{2 \max\left\{\varepsilon^2\lambda, 1\right\}}\right)$ as the solution to $2-\mathrm{SUM}\left(\varepsilon^{-2}, \varepsilon^2 M, \max\left\{\varepsilon^2\lambda, 1\right\}\right)$.

For the 2-SUM problem, let $r = \varepsilon^{-2} - \sum_{i \in [\varepsilon^{-2}]} \mathrm{DISJ}\left(X^i, Y^i\right)$ be the number of string pairs with intersections. Since there are ε^{-2} pairs $\left(X^i, Y^i\right)$, r is at most ε^{-2}.

From our definition of 2-SUM, each intersecting string pair has $\max\{\varepsilon^2\lambda, 1\}$ intersections. x, y are formed by concatenations, so $\text{INT}(x, y) = r \max\{\varepsilon^2\lambda, 1\}$. Since $\sqrt{M} \geq 3 \max\{\lambda, \varepsilon^{-2}\} = 3\varepsilon^{-2}\max\{\varepsilon^2\lambda, 1\} \geq 3r \max\{\varepsilon^2\lambda, 1\} = 3 \cdot \text{INT}(x, y)$, Lemma 5.5 is applicable to $G_{x,y}$ so that

$$\text{MinCUT}(G_{x,y}) = 2r \max\{\varepsilon^2\lambda, 1\}.$$

Since $\mathscr{A}$ approximates MINCUT up to a $(1 \pm \varepsilon)$ factor, $\mathscr{A}(G_{x,y}) = 2r(1 \pm \varepsilon) \max\{\varepsilon^2\lambda, 1\}$. Thus, $\mathscr{B}$'s output to the 2-SUM problem is within $(\varepsilon^{-2} - r) \pm r\varepsilon = \sum_{i \in [\varepsilon^{-2}]} \text{DISJ}(X^i, Y^i) \pm r\varepsilon$. Recall that $r \leq \varepsilon^{-2}$. We can see that $\mathscr{B}$ approximates $2 - \text{SUM}(\varepsilon^{-2}, \varepsilon^2 M, \max\{\varepsilon^2\lambda, 1\})$ up to additive error ε^{-1}.

To compare the complexities of $\mathscr{A}$ and $\mathscr{B}$, recall $\mathscr{A}$ is measured by degree, neighbor, and pair queries, whereas $\mathscr{B}$ is measured by bits of communication. Given the construction of $G_{x,y}$, as shown in [ER18], degree, neighbor, and pair queries can each be simulated using at most 2 bits of communication:

- Degree queries: each vertex in $G_{x,y}$ has degree $\sqrt{M}$ so Alice and Bob do not need to communicate to simulate degree queries.
- Neighbor queries: assuming an ordering where a_i 's j'th neighbor is either a'_j or b'_j, Alice and Bob can exchange $x_{i,j}$ and $y_{i,j}$ with 2 bits of communication to simulate a neighbor query.
- Pair queries: Alice and Bob can exchange $x_{i,j}$ and $y_{i,j}$ with 2 bits of communication to determine whether edges (a_i, b'_j) and (b_i, a'_j) exist.

As each of $\mathscr{A}$'s queries can be simulated using up to 2 bits of communication in $\mathscr{B}$, $\mathscr{B}$ can use $O(T)$ bits of communication to simulate T queries in $\mathscr{A}$. So we have established a reduction from approximating $2 - \text{SUM}(\varepsilon^{-2}, \varepsilon^2 M, \max\{\varepsilon^2\lambda, 1\})$ up to additive error ε^{-1} to approximating MINCUT up to a $(1 \pm \varepsilon)$ multiplicative factor. □

We are now ready to prove Theorem 1.3.

Proof of Theorem 1.3.: Given an instance of $2 - \text{SUM}(\varepsilon^{-2}, \varepsilon^2 m, \max\{\varepsilon^2 k, 1\})$, consider the same way of constructing the graph $G_{x,y}$ in Lemma 5.6. From the construction of $G_{x,y}$, the number of edges is $2m$ since each of pair (x_i, y_i) corresponds to 2 edges. Using the promise from 2-SUM, we get that $r \geq \varepsilon^{-2}/1000$, where $r = \sum_{i \in [\varepsilon^{-2}]} \text{DISJ}(X^i, Y^i)$, which means that the size of the minimum cut of $G_{x,y}$ is $2r \cdot \max\{\varepsilon^2 k, 1\} \geq \Omega(\max\{k, \varepsilon^{-2}\})$. When $k \geq \varepsilon^{-2}$, we have that the size of the minimum cut of $G_{x,y}$ is $\Omega(k)$, and from Lemma 5.6 we obtain that any algorithm $\mathscr{A}$ that satisfies the guarantee on the distribution of $G_{x,y}$ must have $\Omega(m/(\varepsilon^2 k))$ queries in expectation. When $k < \varepsilon^{-2}$, the size of the minimum cut of $G_{x,y}$ is $\Omega(\varepsilon^{-2})$ and similarly we get that any algorithm $\mathscr{A}$ that satisfies the guarantee on the distribution

of $G_{x,y}$ must use $\Omega(m)$ queries in expectation. Combining the two, we finally obtain an $\Omega(\min\{m, \frac{m}{\epsilon^2 k}\})$ lower bound on the expected number of queries in the local query model. □

5.4 Almost Matching Upper Bound

In this section, we will show that our lower bound is tight up to logarithmic factors. In the work of [BGMP21], the authors presented an algorithm that uses $O(\frac{m}{k} \cdot \text{poly}(\log n, 1/\epsilon))$ queries, where k is the size of the minimum cut. We will show that, despite their analysis giving a dependence of $1/\epsilon^4$, a slight modification of their algorithm yields a dependence of $1/\epsilon^2$. Formally, we have the following theorem.

Theorem 5.7 (essentially [BGMP21]).—*There is an algorithm that solves the minimum cut query problem up to a* $(1 \pm \epsilon)$*-multiplicative factor with high constant probability in the local query model. Moreover, the expected number of queries used by this algorithm is* $\widetilde{O}\left(\frac{m}{\epsilon^2 k}\right)$.

To prove Theorem 5.7, we first give a high-level description of the algorithm in [BGMP21]. The algorithm is based on the following sub-routine.

Lemma 5.8 ([BGMP21]).—*There exists an algorithm* VERIFY-GUESS (D, t, ϵ) *which makes* $\widetilde{O}\left(\epsilon^{-2} m/t\right)$ *queries in expectation such that (here* D *is the degree of each node)*

1. *If* $t \geq \frac{2000 \log n}{\epsilon^2} \cdot k$, *then* $\text{Verify} - \text{Guess}(D, t, \epsilon)$ *rejects* t *with probability at least* $1 - \frac{1}{\text{poly}(n)}$.
2. *If* $t \leq k$, *then* $\text{Verify} - \text{Guess}(D, t, \epsilon)$ *accepts* t *and outputs a* $(1 \pm \epsilon)$*-approximation of* k *with probability at least* $1 - \frac{1}{\text{poly}(n)}$.

Given the above sub-routine, the algorithm initializes a guess $t = \frac{n}{2}$ for the value of the minimum cut k and proceeds as follows:

- if $\text{Verify} - \text{Guess}(D, t, \epsilon)$ rejects t, set $t = t/2$ and repeat the process.
- if $\text{Verify} - \text{Guess}(D, t, \epsilon)$ accepts t, set $t = t/\kappa$ where $\kappa = \frac{2000 \log n}{\epsilon^2}$. Let $\widetilde{k} = \text{Verify} - \text{Guess}(D, t, \epsilon)$ and return the value of $\widetilde{k}$ as the output.

To analyze the query complexity of the algorithm, notice that when VERIFY-GUESS first accepts t, we have that $\frac{k}{2} < t < \kappa k$. which means that $t/\kappa < k$ and hence one call to $\text{Verify} - \text{Guess}(D, t/\kappa, \epsilon)$ will get the desired output. However, at a time in $t = \Theta(k/\kappa)$, the VERIFY-GUESS procedure needs to make $\widetilde{O}\left(\frac{m}{\epsilon^4 k}\right)$ queries in expectation.

To avoid this, the crucial observation is that, during the above binary search process, the error parameter of $\text{Verify} - \text{Guess}(D, t, \epsilon)$ does not have to be set to ϵ. Using a

small constant β_0 is sufficient. This way, when VERIFY-GUESS (D, t, β_0) first accepts t, we have $\frac{k}{2} < t < c \log(n) \cdot k$, where c is a constant. Consequently, the output of $\text{Verify} - \text{Guess}(D, t/(c \log n), \varepsilon)$ will satisfy the error guarantee. Using the analysis in [BGMP21], we can show that the query complexity of the new algorithm is $\widetilde{O}\left(\frac{m}{\varepsilon^2 k}\right)$.

Acknowledgement

Yu Cheng is supported in part by NSF Award CCF-2307106. Honghao Lin and David Woodruff would like to thank support from the National Institute of Health (NIH) grant 5R01 HG 10798-2, and a Simons Investigator Award. Part of this work was done while D. Woodruff was visiting the Simons Institute for the Theory of Computing.

References

[ACK+16]. Andoni Alexandr, Chen Jiecao, Krauthgamer Robert, Qin Bo, Woodruff David P., and Zhang Qin. On sketching quadratic forms. In Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science (ITCS), pages 311–319, 2016. 2, 4, 5, 9, 10

[AGM12]. Kook Jin Ahn Sudipto Guha, and McGregor Andrew. Graph sketches: sparsification, spanners, and subgraphs. In Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS), pages 5–14, 2012. 2

[BGMP21]. Bishnu Arijit, Ghosh Arijit, Mishra Gopinath, and Paraashar Manaswi. Query complexity of global minimum cut. In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM), volume 207 of Leibniz International Proceedings in Informatics (LIPIcs), pages 6:1–6:15, 2021. 2, 3, 18, 19

[BK96]. Benczúr András A. and Karger David R.. Approximating s-t minimum cuts in Õ(n2) time. In Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC), pages 47–55, 1996. 1, 5

[BSS12]. Batson Joshua D., Spielman Daniel A., and Srivastava Nikhil. Twice-Ramanujan sparsifiers. SIAM J. Comput, 41(6):1704–1721, 2012. 1

[CCPS21]. Cen Ruoxu, Cheng Yu, Panigrahi Debmalya, and Sun Kevin. Sparsification of directed graphs via cut balance. In 48th International Colloquium on Automata, Languages, and Programming (ICALP), volume 198 of LIPIcs, pages 45:1–45:21, 2021. 2, 3, 4, 5

[CGP+23]. Chu Timothy, Gao Yu, Peng Richard, Sachdeva Sushant, Sawlani Saurabh, and Wang Junxing. Graph sparsification, spectral sketches, and faster resistance computation via short cycle decompositions. SIAM J. Comput, 52(6):S18–85, 2023. 2

[CKK+18]. Cohen Michael B., Kelner Jonathan A, Kyng Rasmus, Peebles John, Peng Richard, Rao Anup B, and Sidford Aaron. Solving directed Laplacian systems in nearly-linear time through sparse LU factorizations. In Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science (FOCS), pages 898–909, 2018. 2

[CKST19]. Carlson Charles, Kolla Alexandra, Srivastava Nikhil, and Trevisan Luca. Optimal lower bounds for sketching graph cuts. In Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 2565–2569, 2019. 1, 2

[EMPS16]. Ene Alina, Miller Gary L., Pachocki Jakub, and Sidford Aaron. Routing under balance. In Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC), pages 598–611, 2016. 2

[ER18]. Eden Talya and Rosenbaum Will. Lower bounds for approximating graph parameters via communication complexity. In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM), volume 116 of Leibniz International Proceedings in Informatics (LIPIcs), pages 11:1–11:18, 2018. 2, 3, 5, 13, 14, 17

[FHHP19]. Wai Shing Fung Ramesh Hariharan, Harvey Nicholas J. A., and Panigrahi Debmalya. A general framework for graph sparsification. SIAM J. Comput, 48(4):1196–1223, 2019. 1

[IT18]. Ikeda Motoki and Tanigawa Shin-ichi. Cut sparsifiers for balanced digraphs. In Approximation and Online Algorithms - 16th International Workshop (WAOA), volume 11312 of Lecture Notes in Computer Science, pages 277–294, 2018. 2, 3

[JS18]. Jambulapati Arun and Sidford Aaron. Efficient $\tilde{O}(n/\epsilon)$ spectral sketches for the Laplacian and its pseudoinverse. In Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 2487–2503, 2018. 2

[KNR01]. Kremer Ilan, Nisan Noam, and Ron Dana. Errata for: “on randomized one-round communication complexity”. Comput. Complex, 10(4):314–315, 2001. 6

[KP12]. Kapralov Michael and Panigrahy Rina. Spectral sparsification via random spanners. In Innovations in Theoretical Computer Science (ITCS), pages 393–398, 2012. 1

[LS17]. Lee Yin Tat and Sun He. An SDP-based algorithm for linear-sized spectral sparsification. In Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC), pages 678–687, 2017. 1

[McG14]. Andrew McGregor. Graph stream algorithms: a survey. ACM SIGMOD Record, 43(1):9–20, 2014. 2

[RSW18]. Rubinstein Aviad, Schramm Tselil, and Matthew Weinberg S. Computing exact minimum cuts without knowing the graph. In 9th Innovations in Theoretical Computer Science Conference (ITCS), volume 94 of LIPIcs, pages 39:1–39:16, 2018. 2

[SS11]. Spielman Daniel A. and Srivastava Nikhil. Graph sparsification by effective resistances. SIAM J. Comput, 40(6):1913–1926, 2011. 1

[ST04]. Spielman Daniel A. and Teng Shang-Hua. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC), pages 81–90, 2004. 1

[ST11]. Spielman Daniel A. and Teng Shang-Hua. Spectral sparsification of graphs. SIAM J. Comput, 40(4):981–1025, 2011. 1, 5

[SW19]. Saranurak Thatchaphol and Wang Di. Expander decomposition and pruning: Faster, stronger, and simpler. In Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 2616–2635, 2019. 2

[WZ14]. Woodruff David P. and Zhang Qin. An optimal lower bound for distinct elements in the message passing model. In Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), page 718–733, 2014. 5, 13

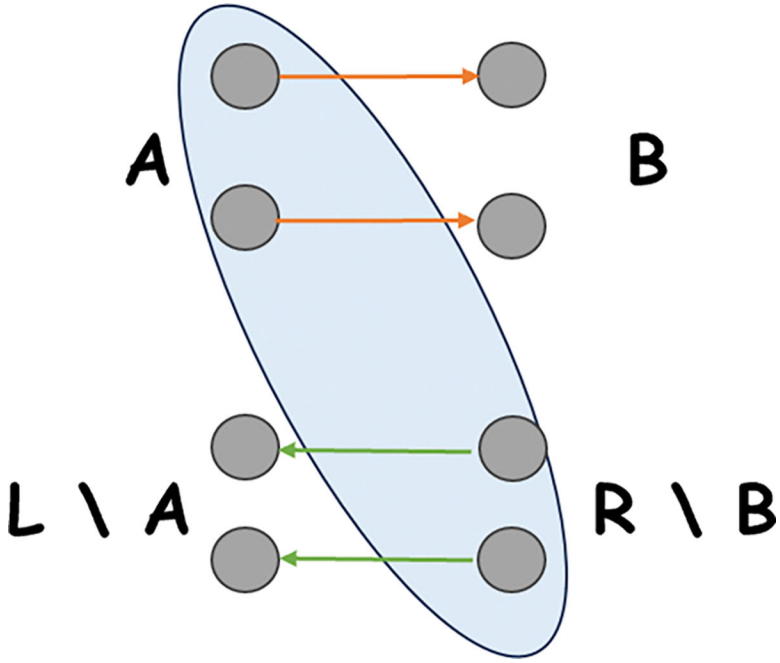

Figure 1:
For $S = A \cup (R \setminus B)$, the (directed) edges from S to $(V \setminus S)$ consist of the following: the forward edges from A to B, each with weight $\Theta(\log(1/\varepsilon))$, and the backward edges from $(R \setminus B)$ to $(L \setminus A)$, each with weight $1/\beta$.

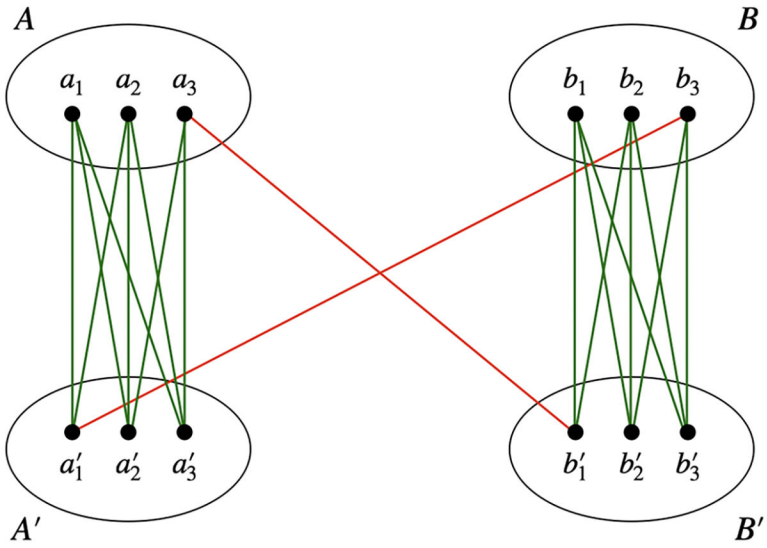

Figure 2:
Example of $G_{x,y}(V, E)$ where x = 000000100 and y = 100010100. The red edges represent the intersection at $x_{31} = y_{31} = 1$. The green edges represent all the non-intersections in x and y.

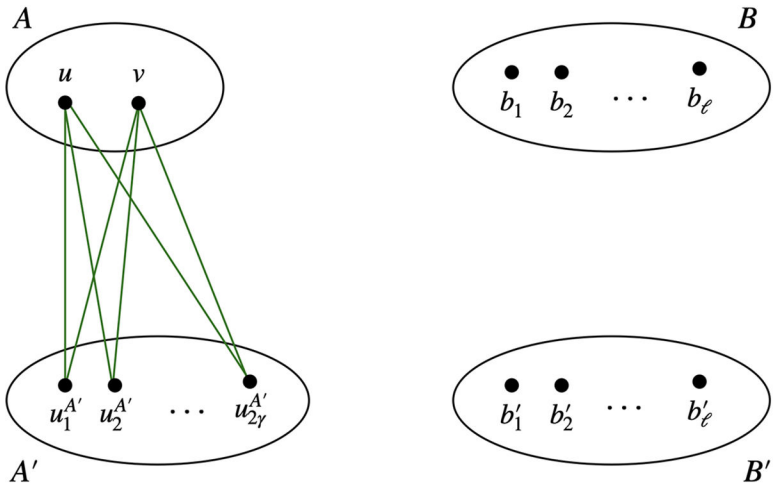

Figure 3:
$u, v \in A$. We omit all the (a_i, b_j'), (b_i, b_j'), and (b_i, a_j') edges.

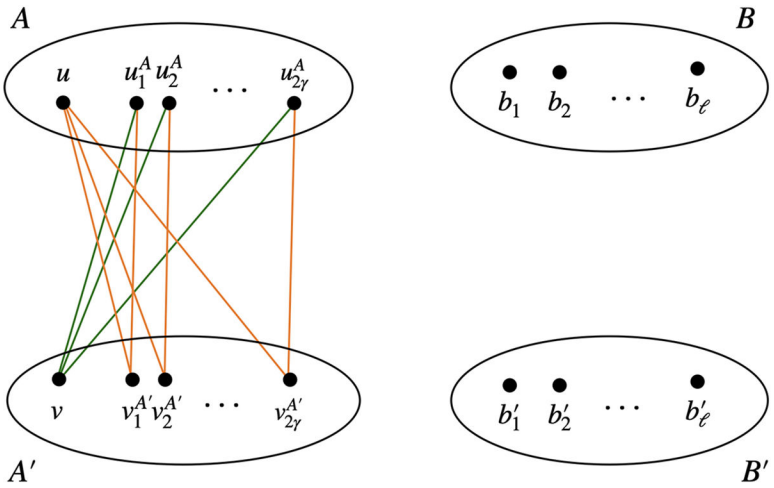

Figure 4:

$\in A$, $v \in A'$. We omit all the (a_i, b_j'), (b_i, b_j'), and (b_i, a_j') edges. The green edges exist since v has at least 2γ neighbors in A. The orange edges exist since u_i^A and u have at least 2γ common neighbors in A'.

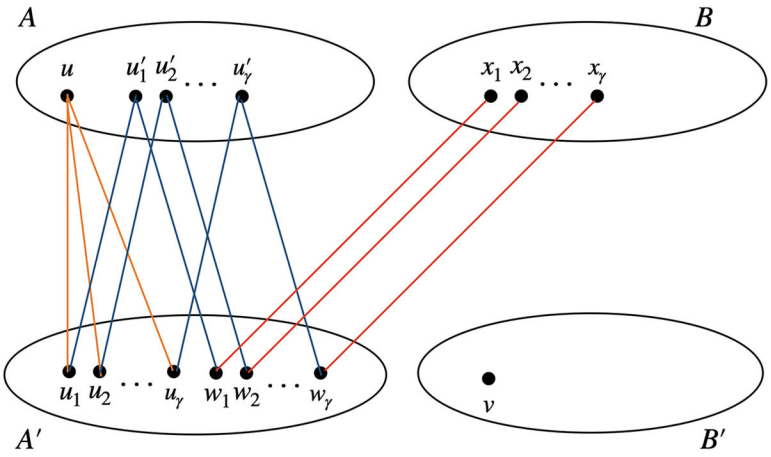

Figure 5:

$\in A$, $v \in B'$. The first set of paths S_1 goes from $u \to u_i \to u_i' \to w_i \to x_i$. We omit the paths from x_i to v, as they are symmetric to the paths from w_i to u. Once we extend the paths from x_i to v, we have γ edge-disjoint paths from u to v. Note that the w_i and x_i may not be distinct.

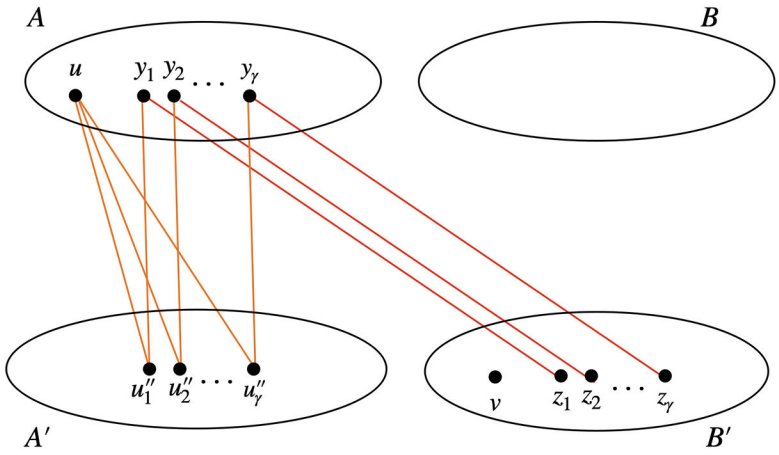

Figure 6:

$\in A$, $v \in B'$. The second set of paths S_2 goes from $u \to u_i^{'} \to y_i \to z_i$. We omit the paths from z_i to v, as they are symmetric to the paths from y_i to u. Once we extend the paths from x_i to v, we have γ edge-disjoint paths from u to v. Note that the y_i and z_i may not be distinct.