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

YU CHENG, Brown University, USA
MAX LI, Carnegie Mellon University, USA
HONGHAO LIN, Carnegie Mellon University, USA
ZI-YI TAI, Carnegie Mellon University, USA
DAVID P. WOODRUFF, Carnegie Mellon University, USA
JASON ZHANG, Carnegie Mellon University, USA

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 approximating cuts in balanced directed graphs. In this problem, we want to build a data structure that can provide $(1 \pm \varepsilon)$-approximation of cut values on a graph 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 the total weight in the other direction. We consider 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 in the for-each model to $\widetilde{\Omega}(n\sqrt{\beta}/\varepsilon)$ and we improve the previous $\Omega(n\beta/\varepsilon)$ lower bound in the for-all model to $\Omega(n\beta/\varepsilon^2)$. [1] This resolves the main open questions of (Cen et al., ICALP, 2021).

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

CCS Concepts: • **Theory of computation → Streaming, sublinear and near linear time algorithms**.

Additional Key Words and Phrases: Graph Sparsification, Data Structures, Minimum Cut, Distributed Algorithms, Communication Complexity

ACM Reference Format:
Yu Cheng, Max Li, Honghao Lin, Zi-Yi Tai, David P. Woodruff, and Jason Zhang. 2024. Tight Lower Bounds for Directed Cut Sparsification and Distributed Min-Cut. *Proc. ACM Manag. Data* 2, 2 (PODS), Article 85 (May 2024), 18 pages. https://doi.org/10.1145/3651148

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

Authors' addresses: Yu Cheng, yu_cheng@brown.edu, Brown University, 115 Waterman St, Providence, RI, USA, 02912; Max Li, mlli@andrew.cmu.edu, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA, USA, 15213; Honghao Lin, honghaol@andrew.cmu.edu, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA, USA, 15213; Zi-Yi Tai, ztai@andrew.cmu.edu, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA, USA, 15213; David P. Woodruff, dwoodruf@cs.cmu.edu, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA, USA, 15213; Jason Zhang, jasonz3@andrew.cmu.edu, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA, USA, 15213.

ACM 2836-6573/2024/5-ART85
https://doi.org/10.1145/3651148

1 INTRODUCTION

The notion of cut sparsifiers has been extremely influential. This was introduced by Benczúr and Karger [4] and is the following: given a graph $G = (V, E, w)$ with $n = |V|$ vertices, $m = |E|$ edges, and edge weights $w_e \geq 0$, together with a desired error parameter $\varepsilon > 0$, a cut sparsifier of G is a subgraph H on the same vertex set V with (possibly) different edge weights, such that the value of every cut in G is $(1 \pm \varepsilon)$-approximated by the corresponding cut in H. Benczúr and Karger [4] 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 [19] and the number of edges was improved to $O(n/\varepsilon^2)$ [3]; see also related work with different bounds for both cut and spectral sparsifiers [6, 10, 12, 14, 17, 18].

In the database community, a key result is the work of [1], which shows how to construct a sparsifer using $\widetilde{O}(n)/\epsilon^2$ linear measurements to $(1 + \varepsilon)$-approximate all cut values. Sketching massive graphs arises in various applications where there are entities and relationships between the entities, such as webpages and hyperlinks between them, people and their friendships, and IP addresses and data flows between them. As large graph databases are often distributed or stored on external memory, sketching algorithms are useful for reducing communication or memory usage in distributed and streaming models. We refer the readers to [15] 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 [2] relaxed the problem to outputting a data structure D, obtained by preprocessing the input graph G, such that given any fixed cut $S \subseteq V = [n] = \{1, 2, \ldots, n\}$, the value $D(S)$ is within a $(1 \pm \varepsilon)$ factor of the cut value of S in G with probability at least $2/3$, which can be amplified to $1 - 1/n$ by independently repeating the data structure $O(\log n)$ times and outputting the median estimate. Notice the order of quantifiers — the data structure need not be correct on all cuts, but just any fixed cut (chosen independently of its randomness) with high probability. This is referred to as the "for-each" model. Surprisingly, [2] showed that such a data structure exists for undirected graphs with poly(n)-bounded integer edge weights of size $\widetilde{O}(n/\varepsilon)$. Notice that the dependence on ε is now only linear, and [2] also shows an $\Omega(n/\varepsilon)$ lower bound for this problem. One might ask if the improved dependence on ε is coming from the relaxation of the original sparsification question to arbitrary data structures or to the relaxation to hold for a fixed cut with high probability. In fact [2] show that for any data structure, there is an $\Omega(n/\varepsilon^2)$ bit lower bound if it is required to approximate all cuts simultaneously; the latter is referred to as the "for-all" model. This lower bound was strengthened by a logarithmic factor in [6].

While the above results give a fairly complete picture for undirected graphs, a natural question is if similar improvements are possible for directed graphs. Indeed, this is the main question posed by [7]. That work observes that for directed graphs, even for the for-each model, there is an $\Omega(n^2)$ lower bound without any assumptions on the graph. Motivated by this, [7, 9, 11] introduce the notion of a β-balanced directed graph, meaning that for every directed cut $(S, V \setminus S)$, the total weight of edges from S to $V \setminus S$ is at most a β factor larger than that from $V \setminus S$ to S. This turned out to be a very useful notion for directed graphs, as [7] was able to show that in the for-each model, there is a $\widetilde{O}(n\sqrt{\beta}/\varepsilon)$ upper bound, while in the for-all model, there is a $\widetilde{O}(n\beta/\varepsilon^2)$ upper bound, thus giving non-trivial bounds for both problems for small β. The work of [7] also gave lower bounds: they showed an $\Omega(n\sqrt{\beta/\varepsilon})$ lower bound in the for-each model, and they showed 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 in their bounds for both models in terms of the dependence on ε. The main question left open of [7] is to determine the optimal dependence on ε, which we resolve in this work.

As observed in [2], one of the main ways of using a data structure in the for-each model is to solve the distributed minimum cut problem. Indeed, by using the fact that there are at most $n^{O(C)}$ cuts with value within a factor of C of the minimum cut, it is possible to run a cut sparsifier with constant ε in parallel with a data structure for general ε in the for-each model. Then, one can query the data structure on each of the at most $\text{poly}(n)$ $O(1)$-approximate minimum cuts output by the sparsifier, 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 the so-called Local Query Model, introduced in [16] and studied for minimum cut in [5, 8]. 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 assume there is an oracle and we access an edge through the following types of local queries:

(1) Degree query: given $u \in V$, the oracle reports the degree of u.
(2) Edge query: given $u \in V$ and index i, the oracle reports the i-th neighbor of u. If the edge does not exist, then it reports $\perp$.
(3) Adjacency query: given $u, v \in V$, the oracle reports whether $(u, v) \in E$.

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

Previous work [8] has shown an $\Omega(\frac{m}{k})$ lower bound and the main open question is what the dependence on ε should be. There is also an upper bound in [5] of $\widetilde{O}(\frac{m}{k\text{poly}(\varepsilon)})$ and a natural question is to close this gap.

1.1 Our Results.

We resolve the main open questions above.

Cut Sketch for Balanced (Directed) Graphs. Given an n-node β-balanced (directed) graph, the previous work of [7] gives an $\widetilde{O}(n\beta/\varepsilon^2)$ upper bound in the for-all model and an $\widetilde{O}(n\sqrt{\beta}/\varepsilon)$ upper bound for the for-each model, along with an $\Omega(n\beta/\varepsilon)$ lower bound and an $\Omega(n\sqrt{\beta/\varepsilon})$ lower bound, respectively, for these two models. In this work, we close these gaps and resolve the ε dependence, improving the lower bounds to asymptotically match the upper bounds for all parameters n, β, and ε. Formally, we have:

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

Theorem 1.2 (For-all Cut Sketch for Balanced Graphs). *Let $\beta \geq 1$ and $0 < \varepsilon < 1$ with $\beta/\varepsilon^2 \leq n/2$. Any $(1 \pm \varepsilon)$ for-all cut sketching algorithm must be of size $\Omega(n\beta/\varepsilon^2)$ bits.*

Query Complexity of Min-Cut in the Local Query Model.

We close the gap on the ε dependence in the query complexity of approximating $|\text{MinCut}(G)|$ up to a $(1 \pm \varepsilon)$-factor in the local query model by providing a tight $\Omega(\min\{m, \frac{m}{\varepsilon^2 k}\})$ lower bound, which improves the previous $\Omega(\frac{m}{k})$ lower bound in [8]. Formally, we have:

Theorem 1.3 (Approximating Min-Cut using Local Queries). *Any algorithm that estimates the size of the 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 k is the size of the minimum cut and m is the number of edges in G.*

To show the tightness of our lower bound, we also show that after a simple modification, the upper bound in [5] actually becomes $\widetilde{O}\left(\frac{m}{\varepsilon^2 k}\right)$, which means 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-Each Cut Sketch Lower Bound. Let $k = \sqrt{\beta}/\varepsilon$. At a high level, we partition the n nodes into $n/(2k)$ sub-graphs, and each sub-graph is a bipartite graph with two parts L and R with $|L| = |R| = \sqrt{\beta}/\varepsilon$. We then divide L and R into $\sqrt{\beta}$ 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 encode each entry in a string $s \in \{-1, 1\}^{1/\varepsilon^2}$ into a forward edge with weight 1 and a backward edge with weight $\frac{1}{\beta}$ to make the graph β-balanced. If we could approximately decode this string from our queries, 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 follow a standard decoding method for undirected graphs where we encode one bit s_i into one edge, then due to the backward edges, the total weight of a cut query would be $\Omega(1/\varepsilon^2)$, which results in an $\Omega(1/\varepsilon)$ additive error and does not allow us to obtain the value s_i. To address this, we instead encode $1/\varepsilon^2$ bits of information across $1/\varepsilon^2$ edges simultaneously, that is, we do not encode each bit s_i into a single edge. 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, even though each edge in $A \times B$ is used to encode many bits of z, the encoding of two different bits of z 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. We consider a similar construction for our for-all lower bound but now each sub-graph has $k = \beta/\varepsilon^2$ nodes. Each of the forward edges has weight 1 or 2 with equal probability and each backward edge has weight $\frac{1}{\beta}$. We similarly partition R into β clusters $|R_1| = |R_2| = \ldots = |R_\beta| = 1/\varepsilon^2$. We then attempt to follow the same idea as in [2] for undirected graphs, which reduces showing our lower bound to the following problem. Consider one node $\ell_i \in L$ and one random subset T of R_j where $|T| = \frac{|R_j|}{2}$. Let $N(\ell_i)$ denote the set of ℓ_i's neighbors v such that the weight from ℓ_i to v is equal to 2 (rather than equal to 1). Following a similar procedure as in [2], we show that if from a $(1 \pm c\varepsilon)$ for-all cut sketch we can distinguish whether $|N(\ell_i) \cap T| \geq \frac{1}{4\varepsilon^2} + \frac{c}{2\varepsilon}$ or $|N(\ell_i) \cap T| \leq \frac{1}{4\varepsilon^2} - \frac{c}{2\varepsilon}$, we can obtain the desired $\Omega(n\beta/\varepsilon^2)$ lower bound.

However, the reduced problem is still problematic. The difference in the values for the two cases is $1/\varepsilon$ while a corresponding cut query has value β/ε^4, which yields a much larger β/ε^3 error than the $1/\varepsilon$ error needed to carry out the reduction. To overcome this, note that up until now we have not utilized the property that the cut sketch preserves all cut values rather than a single cut. We then make use of the following crucial observation in [2]. For each of the nodes in L, in expectation there will be roughly a $\frac{1}{2}$-fraction of the ℓ_i satisfying $|N(\ell_i) \cap T| \geq \frac{1}{4\varepsilon^2} + \frac{c}{2\varepsilon}$ and $|N(\ell_i) \cap T| \leq \frac{1}{4\varepsilon^2} - \frac{c}{2\varepsilon}$. If we enumerate over all the possible $\beta/(2\varepsilon^2)$ subsets L then we will eventually be lucky enough to find a set $Q \subseteq L$ which contains almost all of the $\beta/(2\varepsilon^2)$ nodes ℓ_i for which $|N(\ell_i) \cap T| \geq \frac{1}{4\varepsilon^2} + \frac{c}{2\varepsilon}$ and now since there are $\beta/(2\varepsilon^2)$ such nodes, the slight c/ε bias will in total contribute $c\beta/\varepsilon^3$, which is enough to be detected even under an $O(\beta/\varepsilon^3)$ additive error.

We remark that unlike the undirected graph case where each sub-graph has $O(1/\varepsilon^2)$ nodes, for the directed graph the size of each sub-graph is necessarily dependent on the value of β, as in our construction the cut value of the backward edges has a linear dependence on the value β.

Query Complexity of Min-Cut in the Local Query Model. We use communication complexity, but unlike in previous work, here we consider the following 2SUM problem [20]. Specifically, given t L-length 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 guarantee that $\mathrm{INT}(x^i, y^i) = 0$ or α, as well as at least a constant fraction of the (x^i, y^i) satisfy $\mathrm{INT}(x^i, y^i) = \alpha$, where $\mathrm{INT}(x, y) = \sum_{i=1}^{L} x_i \wedge y_i$ is the number of indices where x and y are both 1, and $\mathrm{DISJ}(x, y)$ is the set-disjointness problem defined to be 1 if $\mathrm{INT}(x, y) = 0$ and 0 otherwise. Here L, t, and α are parameters that will be determined later.

Motivated by the work of [8], we construct our graph in a similar way, based on the vectors x^i and y^i. Then, we give a careful analysis of the size of the minimum cut of $G_{x,y}$, and show that under some conditions, the size of the minimum cut is exactly $2\sum_{i\in[t]} \mathrm{INT}(x^i, y^i)$, from which we get that a $(1 \pm \varepsilon)$-approximation of the Min-Cut yields an approximation of $\sum_{i\in[t]} \mathrm{DISJ}(x^i, y^i)$ up to a $\sqrt{\varepsilon}$ additive error, from which we get the desired lower bound.

We remark that to our knowledge, there are generally not enough lower bounds in sublinear algorithms that focus on the dependence on ε, and we believe that the techniques here can be useful for other problems as well.

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 in E that go 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 $w(\{u\}, S)$.

Given two n-dimensional vectors $u, v \in \mathbb{R}^n$, define $u \otimes v \in \mathbb{R}^{n^2}$ to be the tensor product of u and v. Given a matrix $A \in \mathbb{R}^{n\times d}$, we use A_i to denote the i-th row of A.

Directed Cut Sketches. We start with definitions of β-balanced graphs, for-all and for-each cut sketches [2, 4, 7, 19].

Definition 2.1 (β-Balanced). 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 $0 < \varepsilon < 1$ and let $G = (V, E, w)$ be a weighted directed graph. We say $\mathrm{sk}(G)$ is a $(1 \pm \varepsilon)$ for-all cut sketch of G if there exists a function f such that, 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 the notion of "for-each" cut sparsifiers and sketches. Instead of approximating the value of all cuts simultaneously, we only require that the value of any individual cut is preserved with high constant probability.

Definition 2.3 (For-Each Cut Sketch). Let $0 < \varepsilon < 1$ and let $G = (V, E, w)$ be a weighted directed graph. We say $\mathrm{sk}(G)$ is a $(1 \pm \varepsilon)$ for-each cut sketch of G if there exists a function f such that, for each $\emptyset \subset S \subset V$, with probability at least $2/3$ (over the randomness in f),

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

3 FOR-EACH CUT SKETCH

In this section, we prove an $\Omega(n\sqrt{\beta}/\varepsilon)$ lower bound on the size of a $(1\pm\varepsilon)$-approximate for-each sketch.

Theorem 3.1. *Let $\beta \geq 1$ and $0 < \varepsilon < 1$ with $\sqrt{\beta}/\varepsilon \leq n/2$. Any $(1\pm\varepsilon)$ for-each cut sketching algorithm for β-balanced n-node graphs must output $\widetilde{\Omega}(n\sqrt{\beta}/\varepsilon)$ bits.*

Our result uses the following communication complexity lower bound for a variant of the Index problem.

Lemma 3.2 ([13]). *Suppose that Alice has a random string $s \in \{-1,1\}^n$ and Bob has a random index $i \in [n]$. If Alice sends a single message to Bob from which Bob can recover s_i with probability at least $2/3$, then Alice must send $\Omega(n)$ bits to Bob.*

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

Lemma 3.3. *For any integer $k \geq 1$, there exists a matrix $M \in \{-1,1\}^{(2^k-1)^2 \times 2^{2k}}$ such that:*

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

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 $\langle H_i, H_j\rangle = 0$ 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 $(2^k-1)^2$ 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 therefore $\langle M_t, M_{t'}\rangle = \langle H_i \otimes H_j, H_{i'} \otimes H_{j'}\rangle = \langle H_i, H_{i'}\rangle\langle H_j, H_{j'}\rangle = 0$. □

We first prove a lower bound for the special case when $n = \Theta(\sqrt{\beta}/\varepsilon)$. Our proof for this special case introduces important building blocks for proving the general case (Theorem 3.1).

Lemma 3.4. *Suppose $n = \Theta(\sqrt{\beta}/\varepsilon)$. Any $(1\pm\varepsilon)$ for-each cut sketching algorithm must output $\widetilde{\Omega}(n\sqrt{\beta}/\varepsilon) = \widetilde{\Omega}(\beta/\varepsilon^2)$ bits.*

At a high level, we will reduce the Index problem to the for-each cut sketching problem. Suppose Alice has a string $s \in \{-1,1\}^{\Theta(\beta/\varepsilon^2)}$. We will construct a graph G to encode s such that Bob can recover s_i with constant probability from a $(1\pm\varepsilon)$ cut sketch of G. By the communication complexity lower bound of the Index problem (Lemma 3.2), the cut sketch must use $\widetilde{\Omega}(\beta/\varepsilon^2)$ bits.

Proof of Lemma 3.4. 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 use a complete bipartite graph G to encode s. Let L and R be 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 the same size, and similarly, we 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 $E(L_i, R_j) \cap E(L_{i'}, R_{j'}) = \varnothing$ for $(i,j) \neq (i',j')$. We fix i and j and focus on the encoding of $s_{i,j}$.

Recall that $|L_i| = |R_j| = 1/\varepsilon$ and $z = s_{i,j} \in \{-1,1\}^{(\frac{1}{\varepsilon}-1)^2}$. 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 $1/\beta$.

We 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 3.3 with $2^k = 1/\varepsilon$. Because $z_t \in \{-1,1\}$ is drawn uniformly at random, each coordinate of x is a sum of $O(1/\varepsilon^2)$ i.i.d. random variables of value ± 1. By standard application of the Chernoff bound and the union bound, we know that with probability at least $99/100$, $\|x\|_\infty \le c_1 \log(1/\varepsilon)/\varepsilon$ for some universal constant $c_1 > 0$. If this event happens, we set $w = \varepsilon x + c_1 \log(1/\varepsilon)\mathbf{1}$. Otherwise, we set $w = \mathbf{1}$ to indicate that the encoding failed. Note that in either case, w is a non-negative vector.

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

We will show that given a $(1 \pm \frac{c\varepsilon}{\log(1/\varepsilon)})$ cut sketch for some universal constant c, Bob can recover a specific bit of z using $O(1)$ cut queries, which implies an $\widetilde{\Omega}(\beta'/\varepsilon'^2)$ lower bound for $\beta' = \beta \log(1/\varepsilon)$ and $\varepsilon' = c\varepsilon/\log(1/\varepsilon)$.

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 sub-string $z = s_{i,j}$ 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 $1, \dots, 1/\varepsilon$ 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 $O(1)$ cut queries. More specifically, by Lemma 3.3, $M_t = h_A \otimes h_B$ 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 $O(\log(1/\varepsilon))$; 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 3.3, $\langle h_A, \mathbf{1} \rangle = \langle h_B, \mathbf{1} \rangle = 0$, so $|A| = |B| = \frac{|L_i|}{2} = \frac{|R_j|}{2} = \frac{1}{2\varepsilon}$. The total weight of the forward edges is $O(\log(1/\varepsilon)/\varepsilon^2)$, and the total weight of the backward edges is $\left(\frac{\sqrt{\beta}}{\varepsilon} - \frac{1}{2\varepsilon}\right)^2 \frac{1}{\beta} = O(1/\varepsilon^2)$, so the cut value $w(S, V \setminus S) = O(\log(1/\varepsilon)/\varepsilon^2)$. The cut sketch returns a $(1 \pm \frac{c\varepsilon}{\log(1/\varepsilon)})$ multiplicative approximation of $w(S, V \setminus S)$, which has $O(c/\varepsilon)$ additive error. After subtracting the total weight of backward edges, which is fixed, Bob has an estimate of $w(A,B)$ with $O(c/\varepsilon)$ additive error. Consequently, Bob can approximate $\langle w, M_t \rangle$ with $O(c/\varepsilon)$ additive error using 4 cut queries. Now consider the value of $\langle w, M_t \rangle$. By Lemma 3.3, $\langle M_t, \mathbf{1} \rangle = 0$ and the rows of M are orthogonal,

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

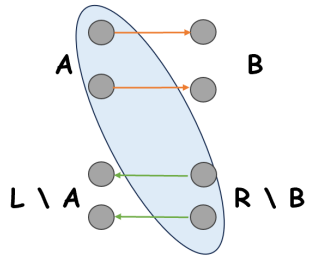

Fig. 1. For $S = A \cup (R \setminus B)$, the edges from S to $(V \setminus S)$: the forward edges are from A to B, each with weight $O(\log(1/\varepsilon))$; and the backward edges are from $(R \setminus B)$ to $(L \setminus A)$, each with weight $1/\beta$.

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

We next consider the case with general values of n, β, and ε, and prove Theorem 3.1.

Proof of Theorem 3.1. 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 that Alice has a random string $s \in \{-1, 1\}^{\Omega(nk)}$. We will show that Alice can encode s into a graph G such that

(i) G has n nodes and is $O(\beta \log(1/\varepsilon))$-balanced, and

(ii) Given a $(1 \pm \frac{c\varepsilon}{\log(1/\varepsilon)})$ for-each cut sketch of G and an index t, where c is a universal constant, Bob can recover s_t with probability at least 2/3.

Then, by Lemma 3.2, the cut sketch must use $\Omega(nk) = \Omega(n\sqrt{\beta}/\varepsilon) = \widetilde{\Omega}(n\sqrt{\beta'}/\varepsilon')$ bits for $\beta' = \beta \log(1/\varepsilon)$ and $\varepsilon' = c\varepsilon/\log(1/\varepsilon)$.

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 with size k. Let s be Alice's random string with length $\beta(\frac{1}{\varepsilon} - 1)^2(\ell - 1) = \Omega(k^2\ell) = \Omega(nk)$. We partition s into $(\ell - 1)$ strings $(s_i)_{i=1}^{\ell-1}$, each with k^2 bits. We then follow the same procedure as in Lemma 3.4 to encode s_i into a complete bipartite graph between V_i and V_{i+1}. Notice that we have $|s_i| = \beta(\frac{1}{\varepsilon} - 1)^2$ and $|V_i| = |V_{i+1}| = \sqrt{\beta}/\varepsilon$, which is the same setting as in Lemma 3.4.

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 $O(\beta \log(1/\varepsilon))$ times the weight of e. For every $u \in V_i$ and $v \in V_{i+1}$, the edge (u, v) has weight $\Theta(\log(1/\varepsilon))$ or 1, and the edge (v, u) has weight $1/\beta$.

We next consider the recovery process. Suppose Bob's index belongs to the sub-string s_i which is encoded by the sub-graph between V_i and V_{i+1}. As in the proof of Lemma 3.4, Bob only needs to approximate $w(A, B)$ for 4 pairs of (A, B) with $O(1/\varepsilon)$ additive error, where $A \subset V_i$, $B \subset V_{i+1}$, and $|A| = |B| = \frac{1}{2\varepsilon}$. To achieve this, Bob can query the cut value $w(S, V \setminus S)$ for $S = A \cup (V_{i+1} \setminus B) \bigcup_{j=i+2}^{t} V_j$. Consider the edges from S to $(V \setminus S)$. There are

- $\frac{1}{4\varepsilon^2}$ forward edges from A to B, each with weight $O(\log(1/\varepsilon))$.
- $\left(\frac{\sqrt{\beta}}{\varepsilon} - \frac{1}{2\varepsilon}\right)^2$ back 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 $\frac{1}{\beta}$.

Consequently, the cut value $w(S, V \setminus S) = O(\log(1/\varepsilon)/\varepsilon^2)$. Given a $(1 \pm \frac{c\varepsilon}{\log(1/\varepsilon)})$ cut sketch, after subtracting the total weight of the backward edges, Bob can approximate $w(A, B)$ with $O(c/\varepsilon)$

additive error. By repeating this 4 times for different (A, B) and choosing a sufficiently small constant c, Bob can recover $s_t \in \{-1, 1\}$. □

4 FOR-ALL CUT SKETCH

In this section, we prove an $\Omega(n\beta/\varepsilon^2)$ lower bound on the size of a $(1 \pm \varepsilon)$ for-all sketch.

Theorem 4.1. *Let $\beta \geq 1$ and $0 < \varepsilon < 1$ with $\beta/\varepsilon^2 \leq n/2$. 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 [2] and uses the following lower bound for an n-fold version of the Gap-Hamming problem.

Lemma 4.2 ([2]). *Consider the following distributional communication problem: Alice has as input h strings $s_1, \ldots, s_h \in \{0,1\}^{1/\varepsilon^2}$ of Hamming weight $\frac{1}{2\varepsilon^2}$, and 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) 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 $\leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$ for some universal constant c;
(3) 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 Bob a 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 must send $\Omega(h/\varepsilon^2)$ bits to Bob.

Before proving Theorem 4.1, we first consider a special case when $n = \beta/\varepsilon^2$.

Lemma 4.3. *Suppose $n = \Theta(\beta/\varepsilon^2)$. Any $(1 \pm \varepsilon)$ for-all cut sketching algorithm must output $\Omega(n\beta/\varepsilon^2) = \Omega(\beta^2/\varepsilon^4)$ bits.*

At a high level, we reduce the distributional Gap-Hamming problem (Lemma 4.2) to for-all cut sketching. Suppose Alice has h strings $s_1, s_2, \ldots, s_h \in \{0,1\}^{1/\varepsilon^2}$ where $h = \beta^2/\varepsilon^2$. We will construct a graph G to encode $s_1, s_2, \ldots, s_h$ such that given an index i and a string $t \in \{0,1\}^{1/\varepsilon^2}$, 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 from a $(1 \pm \varepsilon)$ for-all cut sketch of G. Consequently, by Lemma 4.2, the cut sketch must use $\widetilde{\Omega}(\beta^2/\varepsilon^4)$ bits.

Construction of G. We construct a $2n$-node complete bipartite graph G with two parts L and R where $|L| = |R| = \beta/\varepsilon^2$. We partition R into β disjoint sets $|R_1| = \ldots = |R_\beta| = 1/\varepsilon^2$. Consider the distributional Gap-Hamming problem in Lemma 4.2 with $h = \beta^2/\varepsilon^2$. For simplicity, we re-index the β^2/ε^2 strings as $s_{i,j}$ where $1 \leq i \leq \beta/\varepsilon^2$ and $1 \leq j \leq \beta$. Suppose that the n nodes in L are $\ell_1, \ell_2, \ldots, \ell_n$. We encode $s_{i,j}$ into the edges from the node ℓ_i to R_j. Specifically, for node ℓ_i and the v-th node in R_j, we add a *forward* edge (ℓ_i, v) with weight $s_{i,j}(v) + 1$ and a *backward* edge (v, ℓ_i) with weight $1/\beta$. Note that the encoding of each $s_{i,j}$ is independent since $E(\ell_i, R_j) \cap E(\ell_{i'}, R_{j'}) = \emptyset$ for $(i, j) \neq (i', j')$.

Determining $\Delta(s_{i,j}, t)$ from a for-all cut sketch of G. Suppose that Bob wants to know whether the Hamming distance $\Delta(s_{i,j}, t) \geq \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$ or $\Delta(s_{i,j}, t) \leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$. Let $N(\ell_i)$ denote the set of nodes $v \in R_j$ such that the forward edge from ℓ_i to v has weight 2. Let T be the set of nodes $v \in R_j$ such that $t(v) = 1$. (Recall that $t \in \{0,1\}^{1/\varepsilon^2}$.) We first consider the value of $|N(\ell_i) \cap T|$. We have

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

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

Let $S = \{\ell_i\} \cup (R \setminus T)$. Consider the cut value $w(S, V \setminus S)$. It contains 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 compute $w(\ell_i, T) = \frac{1}{\varepsilon^2} + |N(\ell_i) \cap T|$ and then recover $|N(\ell_i) \cap T|$. However, the issue is that Bob can only get a $(1\pm\varepsilon)$-approximation of $w(S, V\setminus S)$, which has $\Omega(\beta/\varepsilon^3)$ additive error because $w(S, V\setminus S) = \Theta(\beta/\varepsilon^4)$. Bob will not be able to distinguish the two cases as the difference between the two cases is only $\Theta(c/\varepsilon)$.

To overcome this, we follow the idea of [2]. In expectation, roughly half of $\ell_i \in L$ satisfies $|N(\ell_i) \cap T| \geq \frac{1}{4\varepsilon^2} + \frac{c}{2\varepsilon}$. If Bob enumerates all the possible subsets of L of size $\beta/(2\varepsilon^2)$, he will eventually find a set $Q \subseteq L$ containing all $\beta/(2\varepsilon^2)$ nodes ℓ_i such that $|N(\ell_i) \cap T| \geq \frac{1}{4\varepsilon^2} + \frac{c}{2\varepsilon}$. Now since there are $\beta/(2\varepsilon^2)$ nodes in Q, the additional $\frac{c}{2\varepsilon}$ will contribute $\frac{c\beta}{2\varepsilon^3}$ in total, which is enough to be detected even with $O(\beta/\varepsilon^3)$ additive error.

To prove Lemma 4.3, we will need the following two lemmas, which are essentially proved in [2]. The main difference is now $|L|$ becomes β/ε^2 rather than $1/\varepsilon^2$. The same arguments will go through, as the order statistics of the Binomial distribution continue to hold when the number of samples increases.

Lemma 4.4 (Claim 3.5 in [2]). *Let L_{high} denote the set*

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

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

Lemma 4.5 (Lemma 3.4 in [2]). *Bob can enumerate all subsets U of L of size $|U| = \beta/2\varepsilon^2$ and approximate $w(U, T)$ with additive error $O(\beta/\varepsilon^3)$. Let $Q \subset L$ be the subset that achieves the highest value in this process. Then, with probability at least 0.96, Q contains at least $\frac{4}{5}$-fraction of the nodes in L_{high}.*

We are now ready to prove Lemma 4.3.

Proof of Lemma 4.3. Bob enumerates all $U \subseteq L$ with $|U| = \frac{|L|}{2} = \frac{\beta}{2\varepsilon^2}$. Let $S = U \cup (R \setminus T)$. Consider the cut value $w(S, V \setminus S)$. It contains $\frac{\beta}{4\varepsilon^4}$ 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 the forward edges is $O(\beta/\varepsilon^4)$ and the total weight of the backward edges is fixed and is $O(\beta/\varepsilon^4)$.

Consequently, given a $(1 \pm O(\varepsilon))$ multiplicative approximation of $w(S, V \setminus S)$, Bob can subtract the weight of the backward edges and approximate $w(U, T)$ with additive error $O(\beta/\varepsilon^3)$. That is, Bob can obtain the subset Q described in Lemma 4.5. Finally, Bob decides $|N(\ell_i) \cap T| \geq \frac{1}{4\varepsilon^2} + \frac{c}{2\varepsilon}$ and thus $\Delta(s_{i,j}, t) \leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$ if $\ell_i \in Q$, and Bob decides $\Delta(s_{i,j}, t) \geq \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$ if $\ell_i \notin Q$.

We assume the events in Lemmas 4.4 and 4.5 indeed happen. We next analyze the error probability of Bob's decision. Suppose $\Delta(s_{i,j}, t) \leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$ and $|N(\ell_i) \cap T| \geq \frac{1}{4\varepsilon^2} + \frac{c}{2\varepsilon}$, then we have $\mathbf{Pr}[i \in Q] \geq \frac{|L_{\text{high}} \cap Q|}{|L_{\text{high}}|} \geq \frac{4}{5}$. Similarly, we can define $L_{\text{low}} = \{\ell_i \in L : |N(\ell_i) \cap T| \leq \frac{1}{4\varepsilon^2} - \frac{c}{2\varepsilon}\}$ and show that, when $\Delta(s_{i,j}, t) \geq \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$ and $|N(\ell_i) \cap T| \leq \frac{1}{4\varepsilon^2} - \frac{c}{2\varepsilon}$, we have $\mathbf{Pr}[i \notin Q] \geq \frac{|L_{\text{low}} \cap (L \setminus Q)|}{|L_{\text{low}}|} \geq \frac{4}{5}$.

Thus, Bob can distinguish between the two cases with error probability at most $1/5$, which means that Bob can solve distributional Gap-hamming with error probability at most $1/5 + 0.1 \leq 1/3$, which implies an $\Omega((\beta^2/\varepsilon^2) \cdot (1/\varepsilon^2)) = \Omega(\beta^2/\varepsilon^4)$ lower bound. □

We next consider the case with general values of n, β, and ε, and prove Theorem 4.1.

Proof of Theorem 4.1. Let $k = \beta/\varepsilon^2$. We assume w.l.o.g. that k is an integer and n is a multiple of k. Consider the distributional Gap-Hamming problem in Lemma 4.2 with $h = \Omega(n\beta)$. We will show that Alice can encode an $\Omega(n\beta)$-length binary string into a graph G such that

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

(ii) After receiving a string t, an index i, and a $(1 \pm c\varepsilon)$ for-all cut sketch of G, where c is a universal constant, Bob can distinguish whether $\Delta(s_{i,j}, t) \leq \frac{1}{2\varepsilon^2} - \frac{c}{\varepsilon}$ or $\Delta(s_{i,j}, t) \geq \frac{1}{2\varepsilon^2} + \frac{c}{\varepsilon}$ with probability at least $2/3$.

Then, by Lemma 4.2, the for-all cut sketch must use $\Omega(h/\varepsilon^2) = \Omega(n\beta/\varepsilon^2) = \Omega(n\beta'/\varepsilon'^2)$ bits for $\beta' = 2\beta$ and $\varepsilon' = c\varepsilon$.

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 with size k. Let $s_1, s_2, \cdots, s_h \in \{0,1\}^{1/\varepsilon^2}$ be Alice's random strings where $h = (t-1)(\beta^2/\varepsilon^2) = \Omega((n/k)(\beta^2/\varepsilon^2)) = \Omega(n\beta)$. We partition the h strings into $(t-1)$ disjoint sets $S_1, S_2, \cdots, S_{t-1}$, each having (β^2/ε^2) strings. We then follow the same procedure as in Lemma 4.3 to encode S_i into a complete bipartite graph between V_i and V_{i+1}. Recall that S_i has (β^2/ε^2) strings and $|V_i| = |V_{i+1}| = k = \beta/\varepsilon^2$, which is the same setting as in Lemma 4.3.

We first show 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, and the edge (v,u) has weight $1/\beta$.

We next show how Bob can distinguish between the two cases. Suppose Bob's index specifies a string encoded by the sub-graph between V_i and V_{i+1}. As in the proof of Lemma 4.3, we only need to show that given a $(1 \pm c\varepsilon)$ for-all cut sketch, for every subset $U \subset V_i$ with $|U| = \frac{|V_i|}{2}$ and for $T \subset V_{i+1}$, Bob can approximate $w(U,T)$ with additive error $O(\beta/\varepsilon^3)$. To see this, let $S = U \cup (V_{i+1} \setminus T) \bigcup_{j=i+2}^{t} V_j$. Consider the edges from S to $(V \setminus S)$. There 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 $\frac{1}{\beta}$.

Consequently, the cut value $w(S, V \setminus S) = O(\beta/\varepsilon^4)$. Given a $(1 \pm c\varepsilon)$ cut sketch, after subtracting the value of the backward edges, Bob can approximate $w(U,T)$ with $O(c\varepsilon(\beta/\varepsilon^4)) = O(c\beta/\varepsilon^3)$ additive error. By the same arguments as in Lemma 4.3, Bob can distinguish between the two cases of $\Delta(s_i, t)$ with probability at least $2/3$. □

5 LOCAL QUERY COMPLEXITY OF MIN-CUT

In this section, we present an $\Omega\left(\min\{m, \frac{m}{\varepsilon^2 k}\}\right)$ 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 5.1. *Any algorithm $\mathcal{A}$ that estimates the size of the global minimum cut of a graph G up to a $(1 \pm \epsilon)$ factor requires $\Omega(\min\{m, \frac{m}{\varepsilon^2 k}\})$ queries in expectation in the local query model where k is the size of the minimum cut and m is the number of edges in G.*

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 [20], we define the following variant of the 2-SUM(t, L, α) problem.

Definition 5.2. 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 $\mathrm{INT}(x,y) = \sum_{i=1}^{L} x_i \wedge y_i$ denote the number of indices where x and y are both 1. Let $\mathrm{DISJ}(x,y)$ denote whether x and y are disjoint. That is, $\mathrm{DISJ}(x,y) = 1$ if $\mathrm{INT}(x,y) = 0$, and $\mathrm{DISJ}(x,y) = 0$ if $\mathrm{INT}(x,y) \geq 1$.

Definition 5.3. Suppose Alice has t binary strings $(X^1, \ldots, X^t)$ where each string $X^i \in \{0,1\}^L$ has length L and likewise Bob has t strings $(Y^1, \ldots, Y^t)$ each of length L. $\mathrm{INT}(X^i, Y^i)$ is guaranteed to be either 0 or $\alpha \geq 1$ for each pair of strings (X^i, Y^i). Furthermore, at least $1/1000$ of the (X^i, Y^i) pairs are guaranteed to satisfy $\mathrm{INT}(X^i, Y^i) = \alpha$. In the 2-SUM$(t, L, \alpha)$ problem, Alice and Bob want to approximate $\sum_{i \in [t]} \mathrm{DISJ}(X^i, Y^i)$ up to additive error $\sqrt{t}$ with high constant probability.

Lemma 5.4. *To solve 2-SUM$(t, L, 1)$ with high constant probability, the expected number of bits Alice and Bob need to communicate is $\Omega(tL)$.*

Proof. [20] proved an expected communication complexity of $\Omega(tL)$ for 2-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 $(X^1, \ldots, X^t)$ and $(Y^1, \ldots, Y^t)$ 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 $\sum_{i \in [t]} \mathrm{DISJ}(X^i, Y^i)$ with additive error $\Theta(\sqrt{t})$. □

Theorem 5.5. *To solve 2-SUM(t, L, α) 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-SUM$(t, L/\alpha, 1)$ with Alice's strings $(X^1, \ldots, X^t)$ and Bob's strings $(Y^1, \ldots, Y^t)$ 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 $(X^{1,\alpha}, \ldots, X^{t,\alpha})$ and Bob has strings $(Y^{1,\alpha}, \ldots, Y^{t,\alpha})$ is an instance of 2-SUM(t, L, α). From Lemma 5.4, the communication complexity of 2-SUM$(t, L/\alpha, 1)$ is $\Omega(tL/\alpha)$. Thus, the communication complexity of 2-SUM(t, L, α) is $\Omega(tL/\alpha)$. □

5.2 Graph Construction

Inspired by the graph construction from [8], 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 $1 \leq i, j \leq \ell$. 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 5.6. *Given $x, y \in \{0,1\}^N$, if $\sqrt{N} \geq 3 \cdot \mathrm{INT}(x,y)$, 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 $\mathrm{MINCUT}(G) \geq \gamma$. 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 $\mathrm{MINCUT}(G_{x,y}) \geq 2\gamma$. By the construction of the graph, it is easy to see that $\mathrm{CUT}(A \cup A', B \cup B')$

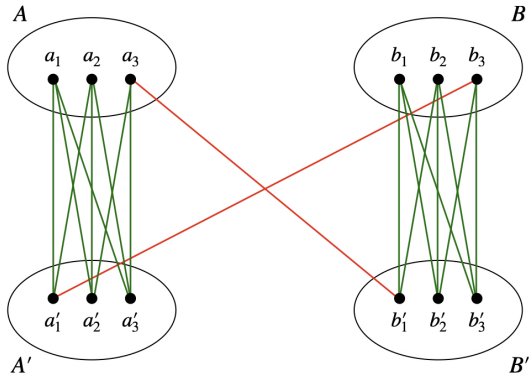

Fig. 2. Example of $G_{x,y}(V, E)$ where $x = 000000100$ and $y = 100010100$. The red edges represent the intersection at $x_{31} = y_{31} = 1$. The green edges represent all the non-intersections in x and y.

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 $\sqrt{N} \geq 3 \cdot \gamma$, then $G_{x,y}$ is 2γ-connected.

Similar to [8], 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.*

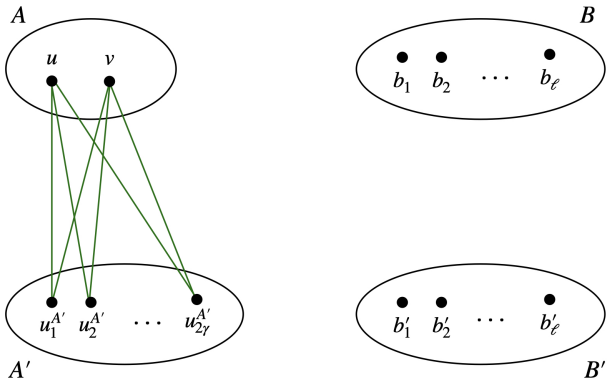

Fig. 3. $u, v \in A$. We omit all the (a_i, b'_j), (b_i, b'_j), and (b_i, a'_j) edges.

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.

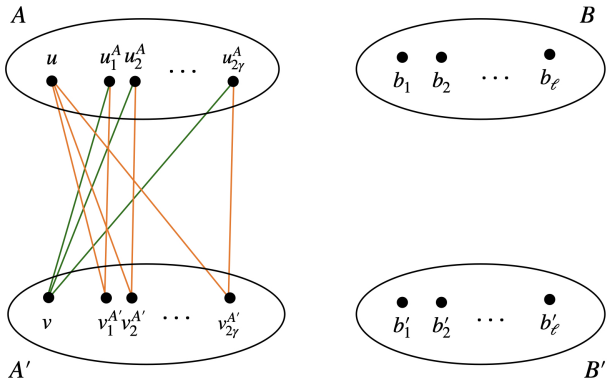

Fig. 4. $u \in A, v \in A'$. We omit all the (a_i, b'_j), (b_i, b'_j), and (b_i, a'_j) edges. The green edges exist since v has at least 2γ neighbors in A. The orange edges exist since u_i^A and u have at least 2γ common neighbors in A'.

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.

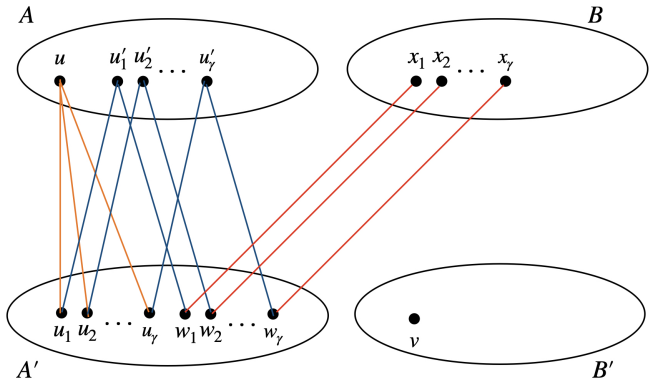

Fig. 5. $u \in A, v \in B'$. The first set of paths S_1 goes from $u \to u_i \to u'_i \to w_i \to x_i$. We omit the paths from x_i to v, as they are symmetric to the paths from w_i to u. Once we extend the paths from x_i to v, we have γ edge-disjoint paths from u to v. Note that the w_i and x_i may not be distinct.

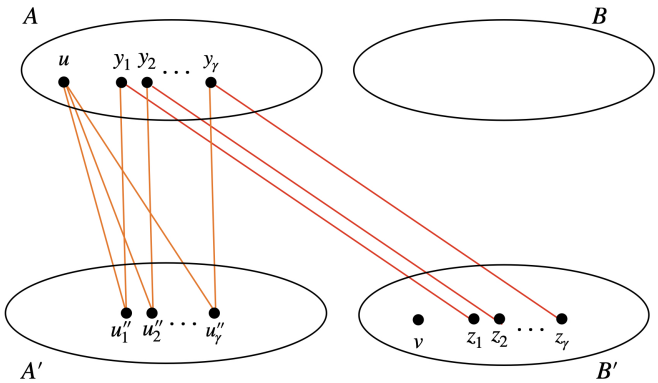

Fig. 6. $u \in A, v \in B'$. The second set of paths S_2 goes from $u \to u''_i \to y_i \to z_i$. We omit the paths from z_i to v, as they are symmetric to the paths from y_i to u. Once we extend the paths from x_i to v, we have γ edge-disjoint paths from u to v. Note that the y_i and z_i may not be distinct.

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-SUM(t, L, α) problem to MINCUT and derive a lower bound on the number of queries in the local query model.

Lemma 5.7. *Given $M, \lambda > 0$, and $0 < \varepsilon < 1$, suppose that we have any algorithm $\mathcal{A}$ 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 $\mathcal{B}$ that can approximate 2-SUM$(\varepsilon^{-2}, \varepsilon^2 M, \max\{\varepsilon^2\lambda, 1\})$ up to an additive error $\sqrt{\varepsilon^{-2}} = \varepsilon^{-1}$ using at most $O(T)$ bits of communication in expectation given $\sqrt{M} \geq 3\max\{\lambda, \varepsilon^{-2}\}$.*

Proof. We will show that the following algorithm $\mathcal{B}$ 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 $\varepsilon^{-2}(\varepsilon^2 M) = M$. 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 $\mathcal{A}(G_{x,y})$ and output $\left(\frac{1}{\varepsilon^2} - \frac{\mathcal{A}(G_{x,y})}{2\max\{\varepsilon^2\lambda,1\}}\right)$ as the solution to 2-SUM$(\varepsilon^{-2}, \varepsilon^2 M, \max\{\varepsilon^2\lambda, 1\})$.

For the 2-SUM problem, let $r = \varepsilon^{-2} - \sum_{i\in[\varepsilon^{-2}]} \mathrm{DISJ}(X^i, Y^i)$ be the number of string pairs with intersections. Since there are ε^{-2} pairs (X^i, Y^i), 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 $\mathrm{INT}(x, y) = r\max\{\varepsilon^2\lambda, 1\}$. Since $\sqrt{M} \geq 3\max\{\lambda, \epsilon^{-2}\} = 3\varepsilon^{-2}\max\{\varepsilon^2\lambda, 1\} \geq 3r\max\{\varepsilon^2\lambda, 1\} = 3 \cdot \mathrm{INT}(x, y)$, Lemma 5.6 is applicable to $G_{x,y}$ so that

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

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

To compare the complexities of $\mathcal{A}$ and $\mathcal{B}$, recall $\mathcal{A}$ is measured by degree, neighbor, and pair queries, whereas $\mathcal{B}$ is measured by bits of communication. Given the construction of $G_{x,y}$, as shown in [8], 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 $\mathcal{A}$'s queries can be simulated using up to 2 bits of communication in $\mathcal{B}$, $\mathcal{B}$ can use $O(T)$ bits of communication to simulate T queries in $\mathcal{A}$. So we have established a reduction from approximating 2-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 5.1.

Proof of Theorem 5.1. Given an instance of 2-SUM$(\varepsilon^{-2}, \varepsilon^2 m, \max\{\varepsilon^2 k, 1\})$, consider the same way of constructing the graph $G_{x,y}$ in Lemma 5.7. 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 $r = \sum_{i\in[\varepsilon^{-2}]} \mathrm{DISJ}(X^i, Y^i)$, 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 5.7 we obtain that any algorithm $\mathcal{A}$ 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 $\mathcal{A}$ 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 $\Omega(\min\{m, \frac{m}{\varepsilon^2 k}\})$ 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 [5], the authors presented an algorithm that uses $O(\frac{m}{k} \cdot \text{poly}(\log n, 1/\varepsilon))$ queries, where k is the size of the minimum cut. We will show that, despite their analysis giving a dependence of $1/\varepsilon^4$, a slight modification of their algorithm yields a dependence of $1/\varepsilon^2$. Formally, we have the following theorem.

Theorem 5.8 (essentially [5]). *There is an algorithm that solves the minimum cut query problem up to a $(1 \pm \varepsilon)$-multiplicative factor with high constant probability in the local query model. Moreover, the expected number of queries used by this algorithm is $\widetilde{O}(\frac{m}{\varepsilon^2 k})$.*

To prove Theorem 5.8, we first give a high-level description of the algorithm in [5]. The algorithm is based on the following sub-routine.

Lemma 5.9 ([5]). *There exists an algorithm Verify-Guess(D, t, ε) which makes $\widetilde{O}(\varepsilon^{-2}m/t)$ queries in expectation such that (here D is the degree of each node)*

(1) If $t \geq \frac{2000 \log n}{\varepsilon^2} \cdot k$, then Verify-Guess$(D, t, \varepsilon)$ rejects t with probability at least $1 - \frac{1}{\text{poly}(n)}$.
(2) If $t \leq k$, then Verify-Guess(D, t, ε) accepts t and outputs a $(1 \pm \varepsilon)$-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 Verify-Guess(D, t, ε) rejects t, set $t = t/2$ and repeat the process.
- if Verify-Guess(D, t, ε) accepts t, set $t = t/\kappa$ where $\kappa = \frac{2000 \log n}{\varepsilon^2}$. Let $\widetilde{k}$ = Verify-Guess(D, t, ε) 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 Verify-Guess$(D, t/\kappa, \varepsilon)$ will get the desired output. However, at a time in $t = \Theta(k/\kappa)$, the Verify-Guess procedure needs to make $\widetilde{O}(\frac{m}{\varepsilon^4 k})$ queries in expectation.

To avoid this, the crucial observation is that, during the above binary search process, the error parameter of Verify-Guess(D, t, ε) 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 Verify-Guess$(D, t/(c \log n), \varepsilon)$ will satisfy the error guarantee. Using the analysis in [5], we can show that the query complexity of the new algorithm is $\widetilde{O}(\frac{m}{\varepsilon^2 k})$.

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

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

[2] Andoni, A., Chen, J., Krauthgamer, R., Qin, B., Woodruff, D. P., and Zhang, Q. On sketching quadratic forms. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science (ITCS)* (2016), pp. 311–319.

[3] Batson, J. D., Spielman, D. A., and Srivastava, N. Twice-Ramanujan sparsifiers. *SIAM J. Comput. 41*, 6 (2012), 1704–1721.

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

[5] Bishnu, A., Ghosh, A., Mishra, G., and Paraashar, M. Query complexity of global minimum cut. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)* (2021), vol. 207 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 6:1–6:15.

[6] Carlson, C., Kolla, A., Srivastava, N., and Trevisan, L. Optimal lower bounds for sketching graph cuts. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms(SODA)* (2019), pp. 2565–2569.

[7] Cen, R., Cheng, Y., Panigrahi, D., and Sun, K. Sparsification of directed graphs via cut balance. In *48th International Colloquium on Automata, Languages, and Programming (ICALP)* (2021), vol. 198 of *LIPIcs*, pp. 45:1–45:21.

[8] Eden, T., and Rosenbaum, W. Lower bounds for approximating graph parameters via communication complexity. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)* (2018), vol. 116 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 11:1–11:18.

[9] Ene, A., Miller, G. L., Pachocki, J., and Sidford, A. Routing under balance. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC)* (2016), pp. 598–611.

[10] Fung, W. S., Hariharan, R., Harvey, N. J. A., and Panigrahi, D. A general framework for graph sparsification. *SIAM J. Comput. 48*, 4 (2019), 1196–1223.

[11] Ikeda, M., and Tanigawa, S. Cut sparsifiers for balanced digraphs. In *Approximation and Online Algorithms - 16th International Workshop (WAOA)* (2018), vol. 11312 of *Lecture Notes in Computer Science*, pp. 277–294.

[12] Kapralov, M., and Panigrahy, R. Spectral sparsification via random spanners. In *Innovations in Theoretical Computer Science (ITCS)* (2012), pp. 393–398.

[13] Kremer, I., Nisan, N., and Ron, D. Errata for: "on randomized one-round communication complexity". *Comput. Complex. 10*, 4 (2001), 314–315.

[14] Lee, Y. T., and Sun, H. An SDP-based algorithm for linear-sized spectral sparsification. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)* (2017), pp. 678–687.

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

[16] Rubinstein, A., Schramm, T., and Weinberg, S. M. Computing exact minimum cuts without knowing the graph. In *9th Innovations in Theoretical Computer Science Conference (ITCS)* (2018), vol. 94 of *LIPIcs*, pp. 39:1–39:16.

[17] Spielman, D. A., and Srivastava, N. Graph sparsification by effective resistances. *SIAM J. Comput. 40*, 6 (2011), 1913–1926.

[18] Spielman, D. A., and Teng, S. 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)* (2004), pp. 81–90.

[19] Spielman, D. A., and Teng, S. Spectral sparsification of graphs. *SIAM J. Comput. 40*, 4 (2011), 981–1025.

[20] Woodruff, D. P., and Zhang, Q. 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)* (2014), p. 718–733.

Received June 2023; revised August 2023; accepted September 2023