Query Complexity of Global Minimum Cut

Arijit Bishnu
Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India

Arijit Ghosh
Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India

Gopinath Mishra
Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India

Manaswi Paraashar
Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata, India

Abstract

In this work, we resolve the query complexity of global minimum cut problem for a graph by designing a randomized algorithm for approximating the size of minimum cut in a graph, where the graph can be accessed through local queries like Degree, Neighbor, and Adjacency queries.

Given $\epsilon \in (0,1)$, the algorithm with high probability outputs an estimate $\hat{t}$ satisfying the following $(1-\epsilon)t \leq \hat{t} \leq (1+\epsilon)t$, where t is the size of minimum cut in the graph. The expected number of local queries used by our algorithm is $\min\left\{m+n, \frac{m}{t}\right\} \text{poly}\left(\log n, \frac{1}{\epsilon}\right)$ where n and m are the number of vertices and edges in the graph, respectively. Eden and Rosenbaum showed that $\Omega(m/t)$ local queries are required for approximating the size of minimum cut in graphs, but no local query based algorithm was known. Our algorithmic result coupled with the lower bound of Eden and Rosenbaum [APPROX 2018] resolve the query complexity of the problem of estimating the size of minimum cut in graphs using local queries.

Building on the lower bound of Eden and Rosenbaum, we show that, for all $t \in \mathbb{N}$, $\Omega(m)$ local queries are required to decide if the size of the minimum cut in the graph is t or $t-2$. Also, we show that, for any $t \in \mathbb{N}$, $\Omega(m)$ local queries are required to find all the minimum cut edges even if it is promised that the input graph has a minimum cut of size t. Both of our lower bound results are randomized, and hold even if we can make Random Edge queries in addition to local queries.

2012 ACM Subject Classification Mathematics of computing → Probabilistic algorithms; Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Query complexity, Global mincut

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2021.6

Category APPROX

Related Version *Previous Version*: `https://arxiv.org/abs/2007.09202`
Previous Version: `https://eccc.weizmann.ac.il/report/2020/108/`

1 Introduction

The global minimum cut (denoted MinCut) of a connected, unweighted, undirected and simple graph $G = (V, E)$, $|V| = n$ and $|E| = m$, is a partition of the vertex set V into two sets S and $V \setminus S$ such that the number of edges between S and $V \setminus S$ is minimized. Let Cut(G) denote this edge set corresponding to a minimum cut in G, and t denote $|\text{Cut}(G)|$. The problem is so fundamental that researchers keep coming back to it again and again across different models [23, 22, 27, 25, 28, 1, 29, 14, 12, 13, 21]. The algorithmic landscape for the minimum cut problem has been heavily influenced by Karger and Stein's work [22, 23] and algorithmic solutions for minimum cut across different models [27, 25, 28, 1, 29, 14, 12,

© Arijit Bishnu, Arijit Ghosh, Gopinath Mishra, and Manaswi Paraashar;
licensed under Creative Commons License CC-BY 4.0

13, 21] * have revisited their approach [22, 23]. Fundamental graph parameter estimation problems, like estimation of the number of edges [11, 17], triangles [7], cliques [8], stars [18], etc. have been solved in the local and bounded query models [16, 17, 24]. Estimation of the size of MinCut is also in the league of such fundamental problems to be solved in the model of local queries.

In property testing [15], a graph can be accessed at different granularities – the query oracle can answer properties about graph that are local or global in nature. Local queries involve the relation of a vertex with its immediate neighborhood, whereas, global queries involve the relation between sets of vertices. Recently using a global query, named Cut Query [29], the problem of estimating and finding MinCut was solved, but the problem of estimating or finding MinCut using local queries has not been solved. The fundamental contribution of our work is to resolve the query complexity of MinCut using local queries. We resolve both the estimation and finding variants of the problem. To start with, we formally define the query oracle models we would be needing for discussions that follow. Before proceeding further, we note that all the upper and lower bound results in this paper are randomized unless otherwise stated.

The query oracle models. We start with the most studied local queries and the random edge query for a graph $G = (V, E)$ where the vertex set V is known but the edge set E is unknown.

- Local Query
  - Degree query: given $u \in V$, the oracle reports the degree of u in V;
  - Neighbor query: given $u \in V$ and a positive integer i, the oracle reports the i-th neighbor of u, if it exists; otherwise, the oracle reports $\perp$;
  - Adjacency query: given u, $v \in V$, the oracle reports whether $\{u, v\} \in E$.
- Random Edge query: The query outputs an uniformly random edge of G.

Apart from the local queries mentioned, in the last few years, researchers have also used the Random Edge query [2, 3]. Notice that the randomness will be over the probability space of all edges, and hence, a random edge query is not a local query. This fact is also evident from the work of Eden and Rosenbaum [10]. We use Random Edge query in conjunction with local queries only for lower bound purposes. The other query oracle relevant for our discussion will be a *global query* called the Cut Query proposed by Rubinstein et al. [29] that was motivated by submodular function minimization. The query takes as input a subset S of the vertex set V and returns the size of the cut between S and $V \setminus S$ in the graph G.

Prologue. Our motivation for this work is twofold – MinCut is a fundamental graph estimation problem that needs to be solved in the local query oracle model and the lower bound of Eden and Rosenbaum [9] who extended the seminal work of Blais et al. [5] to develop a technique for proving query complexity lower bounds for graph properties via reductions from communication complexity. Using those techniques, for graphs that can be accessed by only local queries like Degree, Neighbor, Adjacency and Random Edge, Eden and Rosenbaum [9] showed that MinCut admits a lower bound of $\Omega(m/t)$ in general graphs, where m and t are the number of edges and the size of the minimum cut, respectively, in the graph. However, the query complexity of estimating MinCut (in general graphs)

* The list is to name a few and it is not exhaustive.

has remained elusive as there is no matching upper bound. It is surprising that the query complexity of a fundamental graph problem like MinCut has not been addressed before Eden and Rosenbaum [9].

In this work, we prove an upper bound of $\min\{m+n, \frac{m}{t}\}\text{poly}\left(\log n, \frac{1}{\epsilon}\right)$ for estimating MinCut using local queries only (and not Random Edge query). Observe that for $t \geq 1$, our upper bound almost matches with the lower bound given by Eden and Rosenbaum [9] if we ignore $\text{poly}\left(\log n, \frac{1}{\epsilon}\right)$ term. Note that the case of $t = 0$ is the Connectivity problem, where the objective is to decide whether the graph is connected. We build on the lower bound result for *estimating* MinCut by Eden et al. [9] to show that $\Omega(m)$ local queries are required if we want to determine the exact size of a MinCut or find a MinCut. This result implies that there is a separation between the problem of estimating the size of MinCut and the problem of finding a MinCut using local queries. On the other hand, Babai et al. [4] showed that Connectivity testing has a randomized communication complexity of $\Omega(n)$ [†]. This implies that any algorithm that solves Connectivity requires $\Omega(n/\log n)$ local queries. This is because Alice and Bob can deterministically simulate each local query by communicating at most $\log n$ bits. We have already discussed that $\Omega(m)$ local queries are needed to solve Connectivity. From the lower bounds of $\Omega(m)$ and $\Omega(n/\log n)$ for Connectivity along with the lower bound result of Eden and Rosenbaum [9] for estimating MinCut, our upper bound result on estimating MinCut (ignoring $\text{poly}\left(\log n, \frac{1}{\epsilon}\right)$ term) is tight - this settles the query complexity of MinCut using local queries.

Prior to our work, no local query based algorithm existed for MinCut. But it was Rubinstein et al. [29] who studied MinCut for the first time using Cut Query, a global query. They showed that there exists a randomized algorithm for finding a MinCut in G using $\widetilde{\mathcal{O}}(n)$[‡] Cut Query. The deterministic lower bound of $\Omega(n \log n)$ by Hajnal et al. [20], for Connectivity in communication complexity, implies that $\Omega(n)$ Cut Query are required by any deterministic algorithm to estimate MinCut. The randomized lower bound of $\Omega(n)$ by Babai et al. [4], for Connectivity in communication complexity, says that $\Omega(n/\log n)$ Cut Query are necessary for any randomized algorithm to estimate MinCut [§]. So, if we ignore polylogarithmic factors, the upper bound result by Rubinstein et al. for finding a MinCut along with the above discussed lower bound result for finding a MinCut, imply that there is no separation between the problem of estimating size of MinCut and the problem of finding a MinCut using Cut Query. On a different note, Graur et al. [19] showed a deterministic lower bound of $3n/2$ on the number of Cut Query for estimating MinCut.

Problem statements and results. We focus on two problems in this work.

Minimum Cut Estimation
Input: A parameter $\epsilon \in (0,1)$, and access to an unknown graph G via local queries
Output: A $(1 \pm \epsilon)$-approximation to $|\text{Cut}(G)|$.

Minimum Cut Finding
Input: Access to an unknown graph G via local queries
Output: Find a set Cut(G) .

[†] Here the edge set of the graph is partitioned among Alice and Bob. The objective of Alice and Bob is to determine whether the graph is connected by communicating.
[‡] $\widetilde{\mathcal{O}}(n)$ hides polylogarithmic terms in n.
[§] We would like to thank Troy Lee for pointing us to the papers of Hajnal et al. [20] and Babai et al. [4]

Our results are the following.

▶ **Theorem 1.1** (Minimum cut estimation using local queries). *There exists an algorithm, with* Degree *and* Neighbor *query access to an unknown graph $G = (V, E)$, that solves the minimum cut estimation problem with high probability. The expected number of queries used by the algorithm is* $\min\left\{m+n, \frac{m}{t}\right\} \text{poly}\left(\log n, \frac{1}{\epsilon}\right)$.

Building on the lower bound construction of Eden and Rosenbaum [9], we show that no nontrivial query algorithm exists for finding a minimum cut or even estimating the exact size of a minimum cut in graphs.

▶ **Theorem 1.2** (Lower bound for minimum cut finding, i.e., Cut(G)). *Let $m, n, t \in \mathbb{N}$ with $t \leq n-1$ and $2nt \leq m \leq \binom{n}{2}$* [¶]*. Any algorithm that has access to* Degree, Neighbor, Adjacency *and* Random Edge *queries to an unknown graph $G = (V, E)$ must make at least $\Omega(m)$ queries in order to find all the edges in a minimum cut of G with probability $2/3$.*

▶ **Theorem 1.3** (Lower bound for finding the exact size of the minimum cut, i.e., $|\text{Cut}(G)|$). *Let $m, n, t \in \mathbb{N}$ with $2 \leq t \leq n-2$ and $2nt \leq m \leq \binom{n}{2}$. Any algorithm that has access to* Degree, Neighbor, Adjacency *and* Random Edge *queries to an unknown graph $G = (V, E)$ must make at least $\Omega(m)$ queries in order to decide whether $|\text{Cut}(G)| = t$ or $|\text{Cut}(G)| = t-2$ with probability $2/3$.*

▶ Remark 1. Local queries show a clear separation in its power in finding MinCut as opposed to the estimation problem. This is established by using the tight lower bound of minimum cut estimation (viz. $\Omega(m/t)$ lower bound of Eden and Rosenbaum and our Theorem 1.1) vis-a-vis minimum cut finding as mentioned in our Theorems 1.2 and 1.3 on lower bound for finding Cut(G) . As noted earlier, there is no such separation between estimating and finding MinCut when Cut Query is used.

Notations. In this paper, we denote the set $\{1, \ldots, n\}$ by $[n]$. For ease of notation, we sometimes use $[n]$ to denote the set of vertices of a graph. We say $x \geq 0$ is an $(1 \pm \epsilon)$-approximation to $y \geq 0$ if $|x - y| \leq \epsilon y$. $V(G)$ and $E(G)$ would denote the vertex and edge sets when we want to make the graph G explicit, else we use V and E. For a graph G, $\text{Cut}(G)$ denotes the set of edges in a minimum cut of G. Let A_1, A_2 be a partition of V, i.e., $V = A_1 \cup A_2$ with $A_1 \cap A_2 = \emptyset$. Then, $\mathcal{C}_G(A_1, A_2) = \{\{u, v\} \in E \,:\, u \in A_1 \text{ and } v \in A_2\}$. The statement *with high probability* means that the probability of success is at least $1 - \frac{1}{n^c}$, where c is a positive constant. $\widetilde{\Theta}(\cdot)$ and $\widetilde{\mathcal{O}}(\cdot)$ hides a $\text{poly}\left(\log n, \frac{1}{\epsilon}\right)$ term in the upper bound.

Organization of the paper. Section 2 discusses the query algorithm for estimating the MinCut while Section 3 proves lower bounds on finding the MinCut. Section 4 concludes with a few observations.

2 Estimation algorithm

In this Section, we will prove Theorem 1.1. In Section 2.1, we discuss about the intuitions and give the overview of our algorithm. We formalize the intuitions in Section 2.2.

[¶] As mentioned at the beginning of the introduction, $n.m$ and t denote the number of vertices, number of edges and the size of MinCut in G, respectively.

2.1 Overview of our algorithm

We start by assuming that a lower bound $\hat{t}$ on $t = |\textsc{Cut}(G)|$ is known. Later, we discuss how to remove this assumption.

We generate a random subgraph H of G by sampling each edge of the graph G independently with probability $p = \Theta\left(\log n/\epsilon^2 \hat{t}\right)$ [‖]. Using Chernoff bound, we can show that any particular cut of size k, $k \geq t$, in G is *well approximated* in H with probability at least $n^{-\Omega(k/\hat{t})}$. With this idea, consider the following Algorithm, stated informally, for minimum cut estimation.

Algorithm-Sketch (works with $\hat{t} \leq t$)

Step-1: Generate a random subgraph H of G by sampling each edge in G independently with probability $p = \Theta\left(\log n/\epsilon^2 \hat{t}\right)$. Note that H can be generated by using $\widetilde{O}\left(m/\hat{t}\right)$ Degree and Neighbor queries in expectation. We will discuss it in Algorithm 1 in Section 2.2.

Step-2 Determine $|\textsc{Cut}(H)|$ and report $\widetilde{t} = \frac{|\textsc{Cut}(H)|}{p}$ as a $(1 \pm \epsilon)$-approximation of $|\textsc{Cut}(G)|$.

The number of queries made by the above algorithm is $\widetilde{O}\left(m/\hat{t}\right)$ in expectation. But it produces correct output only when the vertex partition corresponding to $\textsc{Cut}(G)$ and $\textsc{Cut}(H)$ are the same. This is not the case always. If we can show that all cuts in G are approximately preserved in H, then Algorithm-Sketch produces correct output with high probability. The main bottleneck to prove it is that the total number of cuts in G can be exponential. A result of Karger (stated in the following lemma) will help us to make Algorithm-Sketch work.

▶ **Lemma 2.1** (Karger [22]). *For a given graph G the number of cuts in G of size at most $j \cdot |\textsc{Cut}(G)|$ is at most n^{2j}.*

Using the above lemma along with Chernoff bound, we can show the following.

▶ **Lemma 2.2.** *Let G be a graph, $\hat{t} \leq t = |\textsc{Cut}(G)|$ and $\epsilon \in (0,1)$. If $H(V(G), E_p)$ is a subgraph of G where each edge in $E(G)$ is included in E_p with probability $p = \min\left\{\frac{200 \log n}{\epsilon^2 \hat{t}}, 1\right\}$ independently, then every cut of size k in G has size $pk(1 \pm \epsilon)$ in H with probability at least $1 - \frac{1}{n^{10}}$.*

The above lemma implies the correctness of Algorithm-Sketch, which is for minimum cut estimation when we know a lower bound $\hat{t}$ of $|\textsc{Cut}(G)|$. But in general we do not know any such $\hat{t}$. To get around the problem, we start guessing $\hat{t}$ starting from $\frac{n}{2}$ each time reducing $\hat{t}$ by a factor of 2. The guessing scheme gives the desired solution due to Lemma 2.2 coupled with the following intuition when $\hat{t} = \Omega(t \log n/\epsilon^2)$ – if we generate a random subgraph H of G by sampling each edge with probability $p = \Theta\left(\log n/\epsilon^2 \hat{t}\right)$, then H is disconnected with at least a constant probability. So, it boils down to a connectivity check in H. The intuition is formalized in the following lemma that can be proved using Markov's inequality.

▶ **Lemma 2.3.** *Let G be a graph with $|V(G)| = n$, $\hat{t} \geq \frac{2000 \log n}{\epsilon^2} |\textsc{Cut}(G)|$ and $\epsilon \in (0,1)$. If $H(V(G), E_p)$ be a subgraph of G where each edge in $E(G)$ is included in E_p independently with probability $p = \min\left\{\frac{200 \log n}{\epsilon^2 \hat{t}}, 1\right\}$, then H is connected with probability at most $\frac{1}{10}$.*

Before moving to the next section, we prove Lemmas 2.2 and 2.3 here.

[‖] Though p can be more than 1 here, we will make it explicit in the formal description

Proof of Lemma 2.2. If $p = 1$, we are done as the graph H is exactly the same as that of G. So, without loss of generality assume that the graph G is connected. Otherwise, the lemma holds trivially as $|\textsc{Cut}(G)| = 0$, i.e., $\hat{t} = 0$ and $p = 1$. Hence, for the rest of the proof we will assume that $p = \frac{200 \log n}{\epsilon^2 \hat{t}}$.

Consider a cut $\mathcal{C}_G(A_1, A_2)$ of size k in G. As we are sampling each edge with probability p, the expected size of the cut $\mathcal{C}_H(A_1, A_2)$ is pk. Using Chernoff bound (see Lemma B.1 in Section B), we get

$$\mathbb{P}\left(|\mathcal{C}_H(A_1, A_2) - pk| \geq \epsilon pk\right) \leq e^{-\epsilon^2 pk/3\hat{t}} = n^{-\frac{100k}{3\hat{t}}} \tag{1}$$

Note that here we want to show that every cut in G is approximately preserved in H. To do so, we will use Lemma 2.1 along with Equation (1) as follows. Let $Z_1, Z_2, \ldots, Z_\ell$ be the partition of the set of all cuts in G such that each cut in Z_j has the number of edges between $[j \cdot |\textsc{Cut}(G)|, (j+1)|\textsc{Cut}(G)|]$, where $\ell \leq \frac{n}{|\textsc{Cut}(G)|}$ and $j \leq \ell - 1$. From Lemma 2.1, $|Z_j| \leq n^{2j}$. Consider a particular $Z_j, j \in [\ell]$. Using the union bound along with Equation 1, the probability that there exists a cut in Z_j that is not approximately preserved in H is at most $\frac{1}{n^{11}}$. Taking union bound over all Z_j's, the probability that there exists a cut in G that is not approximately preserved is at most $\frac{1}{n^{10}}$. ◀

Proof of Lemma 2.3. Let $\mathcal{C}_G(A_1, A_2)$ be a minimum cut in G. Observe, $\mathbb{E}\left[|\mathcal{C}_H(A_1, A_2)|\right] = p\,|\mathcal{C}_G(A_1, A_2)| = p\,|\textsc{Cut}(G)|$. From Markov's inequality, we get $\mathbb{P}\left(G \text{ is connected}\right) \leq \mathbb{P}\left(|\mathcal{C}_G(A_1, A_2)| \geq 1\right) \leq \mathbb{E}[|\mathcal{C}_G(A_1, A_2)|] \leq \frac{1}{10}$. ◀

2.2 Formal Algorithm (Proof of Theorem 1.1)

In this Section, the main algorithm for minimum cut estimation is described in Algorithm 3 (Estimator) that makes multiple calls to Algorithm 2 (Verify-Guess). The Verify-Guess subroutine in turn calls Algorithm 1 (Sample) multiple times.

Given degree sequence of the graph G, that can be obtained using degree queries, we will first show how to independently sample each edge of G with probability p using only Neighbor queries.

Algorithm 1 Sample(D, p).

```
Input: $D = \{d(i) : i \in [n]\}$, where $d(i)$ denotes the degree of the $i$-th vertex in the
       graph $G$, and $p \in (0, 1]$.
Output: Return a subgraph $H(V, E_p)$ of $G(V, E)$ where each edge in $E(G)$ is
        included in $E_p$ with probability $p$.
Set $q = 1 - \sqrt{1-p}$ and $m = \frac{\sum_{i=1}^{n} d_i}{2}$;
for (each $i \in [n]$) do
    for (each $j \in [d(i)]$ with $d(i) > 0$) do
        // Let $r_j$ be the $j$-th neighbor of the $i$-th vertex;
        Add the edge $(i, r_j)$ to the set $E_p$ with probability $q$;
    end
end
Return the graph $H(V, E_p)$.
```

The following lemma proves the correctness of the above algorithm Sample(D, p).

▶ **Lemma 2.4.** SAMPLE(D, p) *returns a random subgraph* $H(V(G), E_p)$ *of* G *such that each edge* $e \in E$ *is included in* E_p *independently with probability* p*. Moreover, in expectation, the number of* NEIGHBOR *queries made by* SAMPLE(D, p) *is at most* $2pm$*.*

Proof. From the description of SAMPLE(D, p), it is clear that the probability that a particular edge $e \in E(G)$ is added to E_p with probability $1 - (1-q)^2 = p$.

Observe, $\mathbb{E}[|E_p|] = pm$. The bound on the number of NEIGHBOR queries now follows from the fact that SAMPLE(D, p) makes at most $2|E_p|$ many NEIGHBOR queries. ◀

One of the core ideas behind the proof of Theorem 1.1 is that, given an estimate $\hat{t}$ of t, we want to efficiently (in terms of number of local queries used by the algorithm) decide if $\hat{t} \leq t$ or if $\hat{t} \gtrsim \frac{\log n}{\epsilon^2} \times t$. Using Algorithm 2, we will show that this can be done using $\widetilde{\mathcal{O}}(m/\hat{t})$ many NEIGHBOR queries in expectation. Another interesting feature of Algorithm 2 is that, if estimate $\hat{t} \leq t$, then Algorithm 2 outputs an estimate which is a $(1 \pm \epsilon)$-approximation of t.

Algorithm 2 VERIFY-GUESS$(D, \hat{t}, \epsilon)$.

```
Input: $D = \{d(i) : i \in [n]\}$, where $d(i)$ denotes the degree of the $i$-th vertex in the graph $G$
       and $m = \frac{1}{2}\sum_{i=1}^{n} d(i) \geq n-1$. Also, a guess $\hat{t}$, with $1 \leq \hat{t} \leq \frac{n}{2}$, for the size of the
       global minimum cut in $G$, and $\epsilon \in (0,1)$.
Output: The algorithm should "ACCEPT" or "REJECT" $\hat{t}$, with high probability, depending
        on the following
  - If $\hat{t} \leq |\text{CUT}(G)|$, then ACCEPT $\hat{t}$ and also output a $(1 \pm \epsilon)$-approximation of $|\text{CUT}(G)|$
  - If $\hat{t} \geq \frac{200 \log n}{\epsilon^2} |\text{CUT}(G)|$, then REJECT $\hat{t}$

Set $p = \min\left\{\frac{200 \log^2 n}{\epsilon^2 \hat{t}}, 1\right\}$;
Set $\Gamma = 100 \log n$ and Call SAMPLE$(D, p)$ $\Gamma$ times;
// Let $H_i(V, E_p^i)$ be the output of $i$-th call to SAMPLE$(D, p)$, where $i \in [\Gamma]$
if (at least $\Gamma/2$ many $H_i'$s are disconnected) then
    REJECT $\hat{t}$
end
else if (all $H_i$'s are connected) then
    ACCEPT $\hat{t}$, find CUT$(H_i)$ for any $i \in [\Gamma]$, and return $\tilde{t} = \frac{|\text{CUT}(H_i)|}{p}$.
end
else
    Return FAIL.
    // When we cannot decide between "REJECT" or "ACCEPT" it will return FAIL
end
```

The following lemma proves the correctness of Algorithm 2. The lemmas used in proof are Lemmas 2.2, 2.3 and 2.4.

▶ **Lemma 2.5.** VERIFY-GUESS$(D, \hat{t}, \epsilon)$ *in expectation makes* $\widetilde{\mathcal{O}}\left(\frac{m}{\hat{t}}\right)$ *many* NEIGHBOR *queries to the graph* G *and behaves as follows:*

(i) *If* $\hat{t} \geq \frac{2000 \log n}{\epsilon^2} |\text{CUT}(G)|$*, then* VERIFY-GUESS$(D, \hat{t}, \epsilon)$ *rejects* $\hat{t}$ *with probability at least* $1 - \frac{1}{n^9}$*.*

(ii) *If* $\hat{t} \leq |\text{CUT}(G)|$*, then* VERIFY-GUESS$(D, \hat{t}, \epsilon)$ *accepts* $\hat{t}$ *with probability at least* $1 - \frac{1}{n^9}$*. Moreover, in this case,* VERIFY-GUESS$(D, \hat{t}, \epsilon)$ *reports an* $(1 \pm \epsilon)$*-approximation to* CUT(G)*.*

Proof. VERIFY-GUESS$(D, \hat{t}, \epsilon)$ calls SAMPLE(D, ϵ) for $\Gamma = 100 \log n$ times with p being set to $\min\left\{\frac{200 \log n}{\epsilon^2 \hat{t}}, 1\right\}$. Recall from Lemma 2.4 that each call to SAMPLE(D, p) makes in expectation at most $2pm$ many NEIGHBOR queries, and returns a random subgraph $H(V, E_p)$, where each edge in $E(G)$ is included in E_p with probability p. So, VERIFY-GUESS$(D, \hat{t}, \epsilon)$ makes in expectation $\mathcal{O}(pm \log n) = \widetilde{\mathcal{O}}\left(m/\hat{t}\right)$ many NEIGHBOR queries and generates Γ many random subgraphs of G. The subgraphs are denoted by $H_1(V, E_p^1), \ldots, H_\Gamma(V, E_p^\Gamma)$.

(i) Let $\hat{t} \geq \frac{2000 \log n}{\epsilon^2} |\text{CUT}(G)|$. From Lemma 2.3, we have that H_i will be connected with probability at most $\frac{1}{10}$. Observe that in expectation, we get that at least $\frac{9\Gamma}{10}$ many H_i's will be disconnected. By Chernoff bound (see Lemma B.1 in Section B), the probability that at most $\frac{\Gamma}{2}$ many H_i's are disconnected is at most $\frac{1}{n^{10}}$. Therefore, VERIFY-GUESS$(D, \hat{t}, \epsilon)$ rejects any $\hat{t}$ satisfying $\hat{t} \geq \frac{2000 \log n}{\epsilon^2} |\text{CUT}(G)|$ with probability at least $1 - \frac{1}{n^9}$.

(ii) Let $\hat{t} \leq |\text{CUT}(G)|$. Using Lemma 2.2, we have that every cut of size k in G has size $pk(1 \pm \epsilon)$ in H_i with probability at least $1 - \frac{1}{n^{10}}$. Therefore, with probability at least $1 - \frac{\Gamma}{n^{10}}$, for all $i \in [\Gamma]$, every cut of size k in G has size $pk(1 \pm \epsilon)$ in H_i. This implies that if $\hat{t} \leq |\text{CUT}(G)|$ then VERIFY-GUESS$(D, \hat{t}, \epsilon)$ accepts any $\hat{t}$ with probability at least $1 - \frac{1}{n^9}$. Moreover, for any H_i, observe that $\frac{|\text{CUT}(H_i)|}{p}$ is a $(1 \pm \epsilon)$-approximation to $|\text{CUT}(G)|$. Hence, when $\hat{t} \leq |\text{CUT}(G)|$, VERIFY-GUESS$(D, \hat{t}, \epsilon)$ also returns a $(1 \pm \epsilon)$ approximation to $|\text{CUT}(G)|$ with probability $1 - \frac{1}{n^9}$. ◀

ESTIMATOR(ϵ) (Algorithm 3) will estimate the size of the minimum cut in G using DEGREE and NEIGHBOR queries. The main subroutine used by the algorithm will be VERIFY-GUESS$(D, \hat{t}, \epsilon)$.

The following lemma shows that with high probability ESTIMATOR(ϵ) correctly estimates the size of the minimum cut in the graph G, and it also bounds the expected number of queries used by the algorithm.

▶ **Lemma 2.6.** ESTIMATOR*(ϵ) returns a* $(1 \pm \epsilon)$ *approximation to* $|\text{CUT}(G)|$ *with probability at least* $1 - \frac{1}{n^8}$ *by making in expectation* $\min\left\{m + n, \frac{m}{t}\right\}$ *poly*$\left(\log n, \frac{1}{\epsilon}\right)$ *queries and each query is either a* DEGREE *or a* NEIGHBOR *query to the unknown graph* G.

Proof. Without loss of generality, assume that n is a power of 2. If $m < n - 1$ or if there exists a $i \in [n]$ such that $d_i = 0$ then the graph G is disconnected. In this case the algorithm ESTIMATOR(ϵ) makes n DEGREE queries and returns the correct answer. Thus we assume that $m \geq n - 1$.

First, we prove the correctness and query complexity when the graph is connected, that is, $t \geq 1$. Note that ESTIMATOR(ϵ) calls VERIFY-GUESS$(D, \hat{t}, \epsilon)$ for different values of $\hat{t}$ starting from $\frac{n}{2}$. Recall that $\kappa = \frac{2000 \log n}{\epsilon^2}$. For a particular $\hat{t}$ with $\hat{t} \geq \kappa t$, VERIFY-GUESS$(D, \hat{t}, \epsilon)$ does not REJECT $\hat{t}$ with probability at most $\frac{1}{n^9}$ by Lemma 2.5 (i). So, by the union bound, the probability that VERIFY-GUESS$(D, \hat{t}, \epsilon)$ will either ACCEPT or FAIL for some $\hat{t}$ with $\hat{t} \geq \kappa t$, is at most $\frac{\log n}{n^9}$. Hence, with probability at least $1 - \frac{\log n}{n^9}$, we can say that VERIFY-GUESS$(D, \hat{t}, \epsilon)$ rejects all $\hat{t}$ with $\hat{t} \geq \kappa t$.

Observe that, from Lemma 2.5 (ii), the first time $\hat{t}$ satisfies the following inequality $\frac{t}{2} < \hat{t} \leq t$, VERIFY-GUESS$(D, \hat{t}, \epsilon)$ will accept $\hat{t}$ with probability at least $1 - \frac{1}{n^9}$. Therefore, for the first time VERIFY-GUESS$(D, \hat{t}, \epsilon)$ will either ACCEPT or FAIL, then $\hat{t}$ satisfies the following inequality $\frac{t}{2} < \hat{t} < \kappa t$ with probability at least $1 - \frac{\log n + 1}{n^9}$. Let $\hat{t}_0$ denote the first time VERIFY-GUESS returns ACCEPT or FAIL. From the description of ESTIMATOR(ϵ), note that, we get $\hat{t}_u$ by dividing $\hat{t}_0$ by κ. Note that, with probability at least $1 - \frac{1 + \log n}{n^9}$, we have

Algorithm 3 ESTIMATOR(ϵ).

```
Input: DEGREE and NEIGHBOR query access to an unknown graph $G$, and a
       parameter $\epsilon \in (0,1)$.
Output: Either returns a $(1 \pm \epsilon)$-approximation to $t = |\text{CUT}(G)|$ or FAIL
Find the degrees of all the vertices in $G$ by making $n$ many DEGREE queries;
// Let $D = \{d(1), \ldots, d(n)\}$, where $d(i)$ denotes the degree of the $i$-th vertex in $G$;
If $\exists i \in [n]$ such that $d(i) = 0$, then return $t = 0$ and QUIT. Otherwise, proceed as
 follows.
Find $m = \frac{1}{2} \sum_{i=1}^{n} d(i)$. If $m < n - 1$, return $t = 0$ and QUIT. Otherwise, proceed as
 follows.
Set $\kappa = \frac{2000 \log n}{\epsilon^2}$
Initialize $\hat{t} = \frac{n}{2}$.
while ($\hat{t} \geq 1$) do
    Call VERIFY-GUESS($D, \hat{t}, \epsilon$).
    if (VERIFY-GUESS($D, \hat{t}, \epsilon$) returns REJECT) then
        set $\hat{t} = \frac{\hat{t}}{2}$ and continue.
    end
    else
        // Note that in this case VERIFY-GUESS($D, \hat{t}, \epsilon$) either returns FAIL or
         ACCEPT.
        Set $\hat{t}_u = \max\left\{\frac{\hat{t}}{\kappa}, 1\right\}$.
        Call VERIFY-GUESS($D, \hat{t}_u, \epsilon$).
        if (VERIFY-GUESS($D, \hat{t}_u, \epsilon$) returns FAIL or REJECT) then
            return FAIL as the output of ESTIMATOR($\epsilon$)
        end
        else
            Let $\tilde{t}$ be the output of VERIFY-GUESS($D, \hat{t}_u, \epsilon$).
            Return $\tilde{t}$ as the output of ESTIMATOR($\epsilon$).
        end
    end
end
Output: Return that the graph $G$ is disconnected.
```

$\hat{t}_u < t$. We then call the procedure VERIFY-GUESS$(D, \hat{t}, \epsilon)$ with $\hat{t} = \hat{t}_u$. By Lemma 2.5 (ii), VERIFY-GUESS$(D, \hat{t}_u, \epsilon)$ will ACCEPT and report a $(1 \pm \epsilon)$ approximation to t with probability at least $1 - \frac{1}{n^9}$.

We will now analyze the number of DEGREE and NEIGHBOR queries made by the algorithm. We make an initial n many queries to construct the set D. Then at the worst case, we call VERIFY-GUESS$(D, \hat{t}, \epsilon)$ for $\hat{t} = \frac{n}{2}, \ldots, t'$ and $\hat{t} = \frac{t'}{\kappa} \geq \frac{t}{2\kappa}$, where $\frac{t}{2} < t' < \kappa t$. It is because VERIFY-GUESS$(D, \hat{t}, \epsilon)$ accepts $\hat{t}$ with probability $1 - \frac{1}{n^9}$ when the first time $\hat{t}$ satisfy the inequality $\hat{t} \leq t$. Hence, by Lemma 2.5 and the facts that $n \leq \frac{m}{t}$ and $\hat{t}_u \geq \frac{t}{2\kappa}$ with probability at least $1 - \frac{\log n + 1}{n^9}$, in expectation the total number of queries made by the algorithm is at most $n + \log n \cdot \left(1 - \frac{\log n+1}{n^9}\right) \cdot \widetilde{\mathcal{O}}\left(\frac{2\kappa m}{t}\right) + \log n \cdot \left(\frac{\log n+1}{n^9}\right) \cdot \widetilde{\mathcal{O}}(m) = \widetilde{\mathcal{O}}\left(\frac{m}{t}\right)$.

Note that each query made by ESTIMATOR(ϵ) is either a DEGREE or a NEIGHBOR query.

Now we analyze the case when $t = 0$. Observe that VERIFY-GUESS$(D, \hat{t}, \epsilon)$ rejects all $\hat{t} \geq 1$ with probability $1 - \frac{\log n}{n^9}$, and therefore, ESTIMATOR(ϵ) will report $t = 0$. As we have called

VERIFY-GUESS$(D, \hat{t}, \epsilon)$ for all $\hat{t} = \frac{n}{2}, \dots, 1$, the number of queries made by ESTIMATOR(ϵ), in the case when $t = 0$, is $\widetilde{\mathcal{O}}(m) + n$. Note that the additional term of n in the bound comes from the fact that to compute D, the algorithm needs to make n many DEGREE queries. ◀

3 Lower bounds

In this Section, we prove Theorems 1.2 and 1.3 using reductions from suitable problems in communication complexity. In Section 3.1, we discuss about two party communication complexity along with the problems that will be used in our reductions. We will discuss the proofs of Theorems 1.2 and 1.3 in Section 3.2.

3.1 Communication Complexity

In two-party communication complexity there are two parties, Alice and Bob, that wish to compute a function $\Pi : \{0,1\}^N \times \{0,1\}^N \to \{0,1\} \cup \{0,1\}^n$ [**]. Alice is given $\mathbf{x} \in \{0,1\}^N$ and Bob is given $\mathbf{y} \in \{0,1\}^N$. Let x_i (y_i) denotes the i-th bit of $\mathbf{x}$ $(\mathbf{y})$. While the parties know the function Π, Alice does not know $\mathbf{y}$, and similarly Bob does not know $\mathbf{x}$. Thus they communicate bits following a pre-decided protocol $\mathcal{P}$ in order to compute $\Pi(\mathbf{x}, \mathbf{y})$. We say a randomized protocol $\mathcal{P}$ computes Π if for all $(\mathbf{x}, \mathbf{y}) \in \{0,1\}^N \times \{0,1\}^N$ we have $\mathbb{P}[\mathcal{P}(\mathbf{x}, \mathbf{y}) = \Pi(\mathbf{x}, \mathbf{y})] \geq 2/3$. The model provides the parties access to common random string of arbitrary length. The cost of the protocol $\mathcal{P}$ is the maximum number of bits communicated, where maximum is over all inputs $(\mathbf{x}, \mathbf{y}) \in \{0,1\}^N \times \{0,1\}^N$. The communication complexity of the function is the cost of the most efficient protocol computing Π. For more details on communication complexity see [26]. We now define two functions k-INTERSECTION and FIND-k-INTERSECTION and discuss their communication complexity. Both these functions will be used in our reductions.

▶ **Definition 3.1** (FIND-k-INTERSECTION). Let $k, N \in \mathbb{N}$ such that $k \leq N$. Let $S = \{(\mathbf{x}, \mathbf{y}) \in \{0,1\}^N \times \{0,1\}^N : \sum_{i=1}^{N} x_i y_i = k\}$. The FIND-$k$-INTERSECTION function on N bits is a partial function and is defined as FIND-INT$_k^N : S \to \{0,1\}^N$, and is defined as FIND-INT$_k^N(\mathbf{x}, \mathbf{y}) = \mathbf{z}$, where $z_i = x_i y_i$ for each $i \in [N]$.

Note that the objective is that at the end of the protocol Alice and Bob know $\mathbf{z}$.

▶ **Definition 3.2** (k-INTERSECTION). Let $k, N \in \mathbb{N}$ such that $k \leq N$. Let $S = \{(\mathbf{x}, \mathbf{y}) : \sum\limits_{i=1}^{N} x_i y_i = k \text{ or } k-1\}$. The k-INTERSECTION function on N bits is a partial function denoted by INT$_k^N : S \to \{0,1\}$, and is defined as follows: INT$_k^N(\mathbf{x}, \mathbf{y}) = 1$ if $\sum_{i=1}^{N} x_i y_i = k$ and 0, otherwise.

In communication complexity, the k-INTERSECTION function on N bits when $k = 1$ is known as DISJOINTNESS function on N. The following lemmas follow easily from the communication complexity of DISJOINTNESS (see [26]).

▶ **Lemma 3.3.** *Let $k, N \in \mathbb{N}$ such that $k \leq cN$ for some constant $c < 1$. The randomized communication complexity of* FIND-k-INTERSECTION *function on N bits is $\Omega(N)$.*

▶ **Lemma 3.4.** *Let $k, N \in \mathbb{N}$ such that $k \leq cN$ for some constant $c < 1$. The randomized communication complexity of* k-INTERSECTION *function on N bits (*INT$_k^N$*) is $\Omega(N)$.*

[**] The co-domain of Π looks odd, as the the co-domain is $\{0,1\}$ usually. However, we need $\{0,1\} \cup \{0,1\}^n$ to take care of all the problems in communication complexity we discuss in this paper.

3.2 Proofs of Theorems 1.2 and 1.3

The proofs of Theorems 1.2 and 1.3 are inspired from the lower bound proof of Eden and Rosenbaum [9] for estimating MinCut [††].

Proof of Theorem 1.2. We prove by giving a reduction from Find-$t/2$-Intersection on N bits. Without loss of generality assume that t is even. Let $\mathbf{x}$ and $\mathbf{y}$ be the inputs of Alice and Bob. Note that $\sum\limits_{i=1}^{N} x_i y_i = t/2$.

We first discuss a graph $G_{(\mathbf{xy})}(V,E)$ that can be generated from $(\mathbf{x},\mathbf{y})$, such that $|V| = n$ and $|E| = m \geq 2nt$, and works as the "hard" instance for our proof. Note that $G_{(\mathbf{x},\mathbf{y})}$ should be such that no useful information about the MinCut can be derived by knowing only one of $\mathbf{x}$ and $\mathbf{y}$. Let $s = t + \sqrt{t^2 + (m-nt)/2}$ and $N = s^2$. In particular, $2t \leq s \leq 2t + 3\sqrt{m}$. Also, $s \geq \sqrt{m/2}$ and therefore $s = \Theta(\sqrt{m})$.

The graph $G_{(\mathbf{x},\mathbf{y})}$ and its properties:

$G_{(\mathbf{x},\mathbf{y})}$ has the following structure.

- $V = S_A \cup T_A \cup S_B \cup T_B \cup C$ such that $|S_A| = |T_A| = |S_B| = |T_B| = s$ and $|C| = n - 4s$. Let $S_A = \{s_i^A : i \in [s]\}$ and similarly $T_A = \{t_i^A : i \in [s]\}$, $S_B = \{s_i^B : i \in [s]\}$ and $T_B = \{t_i^B : i \in [s]\}$.
- Each vertex in C is connected to $2t$ different vertices in S_A.
- For $i,j \in [s]$: if $x_{ij} = y_{ij} = 1$, then $(s_i^A, t_j^B) \in E$ and $(s_i^B, t_j^A) \in E$; otherwise, $(s_i^A, t_j^A) \in E$ and $(s_i^B, t_j^B) \in E$.

▶ **Observation 3.5.** $G_{(\mathbf{x},\mathbf{y})}$ satisfies the following properties.

Property-1: The degree of every vertex in C is $2t$. For any $v \notin C$, the neighbors of v inside C are fixed irrespective of $\mathbf{x}$ and $\mathbf{y}$; and the number of neighbors outside C is $s \geq 2t$.

Property-2: There are t edges between the vertex sets $(C \cup S_A \cup T_A)$ and $(S_B \cup T_B)$, and removing them $G_{(\mathbf{x},\mathbf{y})}$ becomes disconnected.

Property-3: Every pair of vertices $(S_A \cup T_A \cup C)$ is connected by at least $3t/2$ edge disjoint paths. Also, every pair of vertices in $(S_B \cup T_B)$ is connected by at least $3t/2$ edge disjoint paths.

Property-4: The set of t edges between the vertex sets $(C \cup S_A \cup T_A)$ and $(S_B \cup T_B)$ forms the unique global minimum cut of $G(\mathbf{x},\mathbf{y})$,

Property-5: $x_{ij} = y_{ij} = 1$ if and only if (s_i^A, t_j^B) and (s_i^B, t_j^A) are the edges in the unique global minimum cut of $G_{(\mathbf{x},\mathbf{y})}$.

Proof. Property-1 and Property-2 directly follow from the construction. Now, we will prove Property-3. We first show that every pair of vertices $(S_A \cup T_A \cup C)$ is connected by at least $3t/2$ edge disjoint paths by breaking the analysis into the following cases.

(i) Consider $s_i^A, s_j^A \in S_A$, for $i,j \in [s]$. Under the promise that $\sum_{i=1}^{N} x_i y_i = t/2$, s_i^A, s_j^A have at least $s - t \geq 3t/2$ common neighbors in T_A and thus there are at least $3t/2$ edge disjoint paths connecting them.

(ii) Consider $s_i^A \in S_A$ and $t_j^A \in T_A$, for $i,j \in [s]$. Let $s_{j_1}^A, \ldots, s_{j_{3t/2}}^A$ be $3t/2$ distinct neighbors of t_j^A in S_A. Since, s_i^A has $3t/2$ common neighbors with each $s_{j_r}^A$, $r \in [3t/2]$, there is a matching of size $3t/2$. Denote this matching by $(t_{j_r}^A, s_{j_r}^A)$, $r \in [3t/2]$. Thus $(s_i^A, t_{j_r}^A), (t_{j_r}^A, s_{j_r}^A), (s_{j_r}^A, t_j^A)$, for $r \in [3t/2]$, forms a set of edge disjoint paths of size $3t/2$

[††] Note that Eden and Rosenbaum [9] stated the result in terms k-Edge Connectivity.

from s_i^A to t_j^A, each of length 3. In case s_i^A is one of the neighbors of t_j^A, then one of the $3t/2$ paths gets reduced to (s_i^A, t_j^A), a length 1 path that is edge disjoint from the remaining paths.

(iii) Consider $u, v \in C$. Let $u_1, \ldots, u_{2t} \in S_A$ and $v_1, \ldots, v_{2t} \in S_A$ be the neighbors of u and v respectively in S_A. If for some $i, j \in [2t]$, $u_i = v_j$ then $(u, u_i), (u_i, v_j), (v_j, v)$ is a desired path. Thus, assume $u_i \neq v_j$ for all $i, j \in [2t]$. For all $i \in [2t]$, since u_i and v_i have at least $3t/2$ common neighbors in T_A we can find $3t/2$ edge disjoint paths $(u_i, t_i^A), (t_i^A, v_i)$, where $t_i^A \in T^A$. Existence of $3t/2$ edge disjoint paths from $u \in C$ to $v \in S_A$ can be proved as in (i). and from $u \in C$ to $v \in T_A$ can be proved as in (ii).

Similarly, we can show that every pair of vertices in $(S_B \cup T_B)$ is connected by $3t/2$ many edge disjoint paths.

Observe that Property-4 follows from Property-3, and Property-5 follows from the construction of $G_{(\mathbf{x},\mathbf{y})}$ and Property-4. ◀

Now, by contradiction assume that there exists an algorithm $\mathcal{A}$ that makes $o(m)$ queries to $G_{(\mathbf{x},\mathbf{y})}$ and finds all the edges of a global minimum cut with probability 2/3. Now, we give a protocol $\mathcal{P}$ for Find-$t/2$-Intersection on N bits when the $\mathbf{x}$ and $\mathbf{y}$ are the inputs of Alice and Bob, respectively. Note that $x, y \in \{0,1\}^N$ such that $\sum_{i=1}^{N} x_i y_i = t/2$.

Protocol $\mathcal{P}$ for Find-$t/2$-Intersection

Alice and Bob run the query algorithm $\mathcal{A}$ when the unknown graph is $G_{(\mathbf{x},\mathbf{y})}$. Now we explain how they simulate the local queries and random edge query on $G_{(\mathbf{x},\mathbf{y})}$ by communication. We would like to note that each query can be answered deterministically.

Degree **query:** By Property-1, the degree of every vertex does not depend on the inputs of Alice and Bob, and therefore any degree query can be simulated without any communication.

Neighbor **query:** For $v \in C$, the set of $2t$ neighbors are fixed by the construction. So, any neighbor query involving any $v \in C$ can be answered without any communication. For $i \in [s]$ and $s_i^A \in S_A$, let $N_C(s_i^A)$ be the set of fixed neighbors of s_i^A inside C. So, by Property-1, $d(s_i^A) = \left|N_C(s_i^A)\right| + s$ [‡‡]. The labels of the neighbors of s_i^A are such that the first $\left|N_C(s_i^A)\right|$ many neighbors are inside C, and they are arranged in a fixed but arbitrary order. For $j \in [s]$, the $(|N_C(v)| + j)$-th neighbor of s_i^A is either t_j^B or s_j^A depending on whether $x_{ij} = y_{ij} = 1$ or not, respectively. So, any neighbor query involving vertex in S_A can be answered by 2 bits of communication. Similar arguments also hold for the vertices in $S_B \cup T_A \cup T_B$.

Adjacency **query:** Observe that each adjacency query can be answered by at most 2 bits of communication, and it can be argued like the Neighbor query.

Random Edge **query:** By Property-1, the degree of any vertex $v \in V$ is independent of the inputs of Alice and Bob. Alice and Bob use shared randomness to sample a vertex in V proportional to its degree. Let $r \in V$ be the sampled vertex. They again use shared randomness to sample an integer j in $[d(v)]$ uniformly at random. Then they determine the j-th neighbor of r using Neighbor query. Observe that this procedure simulates a Random Edge query by using at most 2 bits of communication.

Using the fact that $G_{(\mathbf{x},\mathbf{y})}$ satisfies Property-4 and 5, the output of algorithm $\mathcal{A}$ determines the output of protocol $\mathcal{P}$ for Find-$t/2$-Intersection. As each query of $\mathcal{A}$ can be simulated

[‡‡] $d(u)$ denotes the degree of the vertex u in $G_{(\mathbf{x},\mathbf{y})}$

by at most two bits of communication by the protocol $\mathcal{P}$, the number of bits communicated is $o(m)$. Recall that $N = s^2$ and $s = \Theta(\sqrt{m})$. So, the number of bits communicated by Alice and Bob in $\mathcal{P}$ is $o(N)$. This contradicts Theorem 3.3. ◀

Proof of Theorem 1.3. The proof of this theorem uses the same construction as the one used in the proof of Theorem 1.2. The "hard" communication problem to reduce from is $t/2$-INTERSECTION (see Definition 3.2) on N bits, where $N = s^2$ and $s = \Theta(\sqrt{m})$. ◀

4 Conclusion

Our work first and foremost closes a gap in the query complexity of a fundamental problem of finding a minimum cut using local queries. The strength of our algorithm lies in its simplicity – it uses existing ingredients in a fashion suitable for the query framework. The crucial idea was to ensure that cuts are preserved in a sparsified graph in a query framework. We discuss the application of our approach to other cut problems in Appendix A.

References

1. K. J. Ahn, S. Guha, and A. McGregor. 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. M. Aliakbarpour, A. S. Biswas, T. Gouleakis, J. Peebles, R. Rubinfeld, and A. Yodpinyanee. Sublinear-Time Algorithms for Counting Star Subgraphs via Edge Sampling. *Algorithmica*, 80(2):668–697, 2018.
3. S. Assadi, M. Kapralov, and S. Khanna. A Simple Sublinear-Time Algorithm for Counting Arbitrary Subgraphs via Edge Sampling. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference, ITCS*, pages 6:1–6:20, 2019.
4. L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory (preliminary version). In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science, FOCS*, pages 337–347, 1986.
5. E. Blais, J. Brody, and K. Matulef. Property Testing Lower Bounds via Communication Complexity. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC*, pages 210–220, 2011.
6. D. P. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 1st edition, 2009.
7. T. Eden, A. Levi, D. Ron, and C. Seshadhri. Approximately Counting Triangles in Sublinear Time. *SIAM J. Comput.*, 46(5):1603–1646, 2017.
8. T. Eden, D. Ron, and C. Seshadhri. On Approximating the Number of k-Cliques in Sublinear Time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 722–734, 2018.
9. T. Eden and W. Rosenbaum. Lower Bounds for Approximating Graph Parameters via Communication Complexity. In *Proceedings of the 21st International Conference on Approximation Algorithms for Combinatorial Optimization Problems, APPROX*, pages 11:1–11:18, 2018.
10. T. Eden and W. Rosenbaum. On Sampling Edges Almost Uniformly. In *Proceedings of the 1st Symposium on Simplicity in Algorithms, SOSA*, pages 7:1–7:9, 2018.
11. U. Feige. On Sums of Independent Random Variables with Unbounded Variance and Estimating the Average Degree in a Graph. *SIAM J. Comput.*, 35(4):964–984, 2006.
12. M. Ghaffari and B. Haeupler. Distributed algorithms for planar networks II: low-congestion shortcuts, mst, and min-cut. In *Proceedings of the 2016 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 202–219. SIAM, 2016.
13. M. Ghaffari and F. Kuhn. Distributed minimum cut approximation. In *Distributed Computing*, volume 8205 of *Lecture Notes in Computer Science*, pages 1–15, 2013.

14 M. Ghaffari and K. Nowicki. Massively parallel algorithms for minimum cut. In *Proceedings of the 39th Symposium on Principles of Distributed Computing*, pages 119–128. ACM, 2020.

15 O. Goldreich. *Introduction to Property Testing.* Cambridge University Press, 2017.

16 O. Goldreich, S. Goldwasser, and D. Ron. Property Testing and its Connection to Learning and Approximation. *J. ACM*, 45(4):653–750, 1998.

17 O. Goldreich and D. Ron. Approximating Average Parameters of Graphs. *Random Structures & Algorithms*, 32(4):473–493, 2008.

18 M. Gonen, D. Ron, and Y. Shavitt. Counting Stars and Other Small Subgraphs in Sublinear-Time. *SIAM Journal on Discrete Mathematics*, 25(3):1365–1411, 2011.

19 A. Graur, T. Pollner, V. Ramaswamy, and S. M. Weinberg. New Query Lower Bounds for Submodular Function Minimization. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference, ITCS*, volume 151, pages 64:1–64:16, 2020.

20 A. Hajnal, W. Maass, and G. Turán. On the communication complexity of graph properties. In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, STOC*, pages 186–191, 1988.

21 M. Kapralov, Y. T. Lee, C. Musco, C. Musco, and A. Sidford. Single pass spectral sparsification in dynamic streams. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 561–570, 2014.

22 D. R. Karger. Global Min-cuts in RNC, and Other Ramifications of a Simple Min-Cut Algorithm. In *Proceedings of the 4th Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, SODA*, pages 21–30, 1993.

23 D. R. Karger and C. Stein. An $\widetilde{\mathcal{O}}\left(n^2\right)$ Algorithm for Minimum Cuts. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing, STOC*, pages 757–765, 1993.

24 T. Kaufman, M. Krivelevich, and D. Ron. Tight Bounds for Testing Bipartiteness in General Graphs. *SIAM J. Comput.*, 33(6):1441–1483, 2004.

25 K. Kawarabayashi and M. Thorup. Deterministic edge connectivity in near-linear time. *J. ACM*, 66(1):4:1–4:50, 2019.

26 E. Kushilevitz. Communication complexity. In *Advances in Computers*, volume 44, pages 331–360. Elsevier, 1997.

27 A. McGregor. Graph Stream Algorithms: A Survey. *SIGMOD Rec.*, 43(1):9–20, 2014.

28 S. Mukhopadhyay and D. Nanongkai. Weighted Min-Cut: Sequential, Cut-Query, and Streaming Algorithms. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 496–509, 2020.

29 A. Rubinstein, T. Schramm, and S. M. Weinberg. Computing Exact Minimum Cuts Without Knowing the Graph. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference, ITCS*, pages 39:1–39:16, 2018.

A Application of our approach to other cut problems

Sublinear time algorithm for Global minimum cut. For simplicity, the algorithm (Algorithm 3) presented for estimating global minimum cut is $\widetilde{\mathcal{O}}(m)$. But, our algorithm can be adapted to get a sublinear time algorithm (with time complexity $\widetilde{\mathcal{O}}\left(\frac{m}{t}\right)$) for estimating the size of the global minimum cut in the graph. We will sample $\widetilde{\mathcal{O}}\left(\frac{m}{\widehat{t}}\right)$ random edges with replacement from the graph G using $\widetilde{\mathcal{O}}\left(\frac{m}{\widehat{t}}\right)$ using local queries, where $\widehat{t}$ is the guess for the size of the global minimum, rather than sampling each edge with probability $p = \widetilde{\mathcal{O}}\left(\frac{1}{\widehat{t}}\right)$ as we have done in the Algorithm 2. Observe that, after finding the degrees of all the vertices, a random edge can be generated using $\mathcal{O}(1)$ local queries. The rest of the algorithm and its analysis can be adapted directly. Therefore, we have the following result.

▶ **Theorem A.1** (Estimating Global minimum cut in sublinear time)**.** *There exists an algorithm, with* Degree *and* Neighbor *query access to an unknown graph $G = (V, E)$, that solves the minimum cut estimation problem with high probability. With high probability, the time complexity and the query complexity of the algorithm is* $\min\left\{m+n, \frac{m}{t}\right\} \text{poly}\left(\log n, \frac{1}{\epsilon}\right)$.

Global minimum r-way cut. Global minimum r-cut, for a graph $G = ([n], E)$, $|V| = n$ and $|E| = m$, is a partition of the vertex set $[n]$ into r-sets $S_1, \ldots, S_r$ such that the following is minimized: $|\{\{i,j\} \in E \, : \, \exists k, \ell \, (k \neq \ell) \in [r], \text{ with } i \in S_k \text{ and } j \in S_\ell\}|$.

Let $\textsc{Cut}_r(G)$ denote the set of edges corresponding to a minimum r-cut, i.e., the edges that goes across different partitions, and by the size of minimum r-cut, we mean $|\textsc{Cut}_r(G)|$. The sampling and verification idea used in the proof of Theorem 1.1 can be extended directly, together with [22, Corollary 8.2], to get the following result.

▶ **Theorem A.2.** *There exists an algorithm, with* Degree *and* Neighbor *query access to an unknown graph $G = ([n], E)$, that with high probability outputs a $(1 \pm \epsilon)$-approximation of the size of the minimum r-cut of G. The expected number of queries used by the algorithm is* $\min\left\{m+n, \frac{m}{t_r}\right\}$ *poly* $\left(r, \log n, \frac{1}{\epsilon}\right)$, *where* $t_r = |\textsc{Cut}_r(G)|$.

Minimum cuts in simple multigraphs. A graph with multiple edges between a pair of vertices in the graph but without any self loops are called *simple multigraphs*. If we have Degree and Neighbor query access [*] to simple multigraphs then we can directly get the following generalization of Theorem 1.1.

▶ **Theorem A.3** (Minimum cut estimation in simple multigraphs using local queries)**.** *There exists an algorithm, with* Degree *and* Neighbor *query access to an unknown simple multigraph $G = (V, E)$, that solves the minimum cut estimation problem with high probability. The expected number of queries used by the algorithm is* $\min\left\{m+n, \frac{m}{t}\right\}$ *poly* $\left(\log n, \frac{1}{\epsilon}\right)$, *where n is the number of vertices in the multigraph, m is the number of edges in the multigraph and t is the number of edges in a minimum cut.*

B Probability Results

▶ **Lemma B.1** (See [6])**.** *Let $X = \sum_{i \in [n]} X_i$ where X_i, $i \in [n]$, are independent random variables, $X_i \in [0, 1]$ and $\mathbb{E}[X]$ is the expected value of X. Then*

(i) *For* $\epsilon > 0$

$\Pr[|X - \mathbb{E}[X]| > \epsilon \mathbb{E}[X]] \leq \exp\left(-\frac{\epsilon^2}{3}\mathbb{E}[X]\right)$.

(ii) *Suppose $\mu_L \leq \mathbb{E}[X] \leq \mu_H$, then for $0 < \epsilon < 1$*

(a) $\Pr[X > (1+\epsilon)\mu_H] \leq \exp\left(-\frac{\epsilon^2}{3}\mu_H\right)$.

(b) $\Pr[X < (1-\epsilon)\mu_L] \leq \exp\left(-\frac{\epsilon^2}{2}\mu_L\right)$.

[*] For simple multigraphs, we will assume that the neighbors of a vertex are stored with multiplicities.