# Distributed Minimum Cut Approximation

Mohsen Ghaffari
MIT
Cambridge, MA, USA
ghaffari@mit.edu

Fabian Kuhn
University of Freiburg
Freiburg, Germany
kuhn@cs.uni-freiburg.de

## Abstract

We study the problem of computing approximate minimum edge cuts by distributed algorithms. We present two randomized approximation algorithms that both run in a standard synchronous message passing model where in each round, $O(\log n)$ bits can be transmitted over every edge (a.k.a. the CONGEST model). The first algorithm is based on a simple and new approach for analyzing random edge sampling, which we call *random layering technique*. For any any weighted graph and any $\epsilon \in (0, 1)$, the algorithm finds a cut of size at most $O(\epsilon^{-1}\lambda)$ in $O(D) + \tilde{O}(n^{1/2+\epsilon})$ rounds, where $\lambda$ is the minimum-cut size and the $\tilde{O}$-notation hides poly-logarithmic factors in $n$. In addition, using the outline of a centralized algorithm due to Matula [SODA '93], we present a randomized algorithm to compute a cut of size at most $(2 + \epsilon)\lambda$ in $\tilde{O}((D + \sqrt{n})/\epsilon^5)$ rounds for any $\epsilon > 0$. The time complexities of our algorithms almost match the $\tilde{\Omega}(D + \sqrt{n})$ lower bound of Das Sarma et al. [STOC '11], thus leading to an answer to an open question raised by Elkin [SIGACT-News '04] and Das Sarma et al. [STOC '11].

To complement our upper bound results, we also strengthen the $\tilde{\Omega}(D + \sqrt{n})$ lower bound of Das Sarma et al. by extending it to unweighted graphs. We show that the same lower bound also holds for unweighted multigraphs (or equivalently for weighted graphs in which $O(w \log n)$ bits can be transmitted in each round over an edge of weight $w$). For unweighted simple graphs, we show that computing an $\alpha$-approximate minimum cut requires time at least $\tilde{\Omega}(D + \sqrt{n}/\alpha^{1/4})$.

# 1  Introduction

Finding minimum cuts or approximately minimum cuts are classical and fundamental algorithmic graph problems with many important applications. In particular, minimum edge cuts and their size (i.e., the edge connectivity) are relevant in the context of networks, where edge weights might represent link capacities and therefore edge connectivity can be interpreted as the throughput capacity of the network and decomposing a network using small cuts helps designing efficient communication strategies and finding communication bottlenecks (see, e.g., [16, 24]). Both exact and approximate variants of the minimum cut problem have received extensive attention in the domain of centralized algorithms (cf. Section 1.1 for a brief review of the results in the centralized setting). This line of research has lead to (almost) optimal centralized algorithms with running times $\tilde{O}(m+n)$ [15] for the exact version and $O(m+n)$ [21] for obtaining constant-factor approximations. Here, $m$ is the number of edges and $n$ is the number of nodes.

As indicated by Elkin [5] and Das Sarma et al. [3], the problem has remained essentially open in the distributed setting. In the LOCAL model [23] where in each round, a message of unbounded size can be sent over each edge, the problem has trivial time complexity of $\Theta(D)$ rounds, where $D$ is the (unweighted) diameter of the network. The problem is therefore more interesting and also practically more relevant in models where messages are of some bounded size $B$. The standard model incorporating this restriction is the CONGEST model [23], a synchronous message passing model, where in each time unit, $B$ bits can be sent over every link (in each direction). It is often assumed that $B = \Theta(\log n)$. The only known non-trivial result is an elegant lower bound by Das Sarma et al. [3] showing that any $\alpha$-approximation of the minimum cut in weighted graphs requires at least $\Omega(D+\sqrt{n/(B\log n)})$ rounds.

**Our Contribution:**  In this paper, we present two randomized distributed minimum-cut approximation algorithms for undirected weighted graphs. These algorithms have time complexities that almost match the lower bound proven in [3]. In addition, we also extend the lower bound of [3] to unweighted graphs and multigraphs.

Our first algorithm, presented in Section 4, with high probability[1] finds a cut of size at most $O(\varepsilon^{-1}\lambda)$, for any $\epsilon \in (0,1)$ and where $\lambda$ is the edge connectivity, i.e., size of the minimum cut in the network. The time complexity of the algorithm is $O(D) + O(n^{1/2+\epsilon}\log^3 n \log\log n \log^* n)$. This algorithm is based on a simple and novel approach for analyzing random edge sampling. Random edge sampling is a tool that has proven extremely successful also for studying the minimum cut problem in the centralized setting (see, e.g., [16]). Our analysis is based on *random layering*, and we believe that the approach might also be useful for studying other connectivity-related questions. Assume that each edge $e \in E$ of an unweighted multigraph $G = (V, E)$ is independently sampled and added to a subset $E' \subset E$ with probability $p$. Because there is at least one edge cut of size $\lambda$, for $p \le \frac{1}{\lambda}$, the graph $G' = (V, E')$ induced by the sampled edges is disconnected with at least a constant probability. In Section 3, we use the random layering technique to show that if $p = \Omega(\frac{\log n}{\lambda})$, the sampled graph $G'$ is connected with high probability. This bound is optimal and was known previously, with two elegant proofs due to Lomonosov-Polesskii [20], and Karger [11]. Our proof is simple and self-contained and it also serves as a basis for our algorithm presented in Section 4.

The second algorithm, presented in Section 5, finds a cut with size at most $(2+\varepsilon)\lambda$, for any constant $\varepsilon > 0$, in time $O((D + \sqrt{n}\log^* n)\log^2 n \log\log n \cdot \frac{1}{\varepsilon^5})$. This algorithm combines the general approach of Matula's centralized $(2+\varepsilon)$-approximation algorithm [21] with Thurimella's algorithm for sparse edge-connectivity certificates [26] and with the famous random edge sparsification technique of Karger (see e.g., [12]).

To complement our upper bounds, we also extend the lower bound of Das Sarma et al. [3] to unweighted graphs and multigraphs. If the minimum cut problem, or more generally problems related to small edge cuts and edge connectivity, occur in a distributed context, then often the weights of the edges correspond to their capacities. It therefore seems reasonable to assume that over a link of twice the capacity, we can also transmit twice the amount of data in a single time unit. Consequently, it makes sense to assume that over an edge of weight (or capacity) $w \ge 1$, $O(w \log n)$ bits can be transmitted per round (or equivalently that such a link corresponds to $w$ parallel links of unit capacity). The lower bound of [3] critically depends on having links with (very) large weight over which in each round only $O(\log n)$ bits can be transmitted. We generalize the approach of [3] and obtain the same lower bound result as in [3] for the weaker setting where edge weights correspond to edge capacities (i.e., the setting that can be modelled using unweighted multigraphs). Formally, we show that if $Bw$ bits can be

---

[1]Throughout the paper, we use the phrase *with high probability* (w.h.p) to indicate that an event has probability at least $1 - \frac{1}{n}$.

transmitted over every edge of weight $w \geq 1$, for every $\lambda \geq 1$ and every $\alpha \geq 1$, there are $\lambda$-edge-connected networks with diameter $O(\log n)$ on which computing an $\alpha$-approximate minimum cut requires time at least $\Omega\big(\sqrt{n/(B \log n)}\big)$. Further, for unweighted simple graphs, we show that computing an $\alpha$-approximate minimum cut in $\lambda$-edge-connected networks of diameter $O(\log n)$ requires at least time $\Omega\big(\frac{\sqrt{n/(B \log n)}}{(\alpha\lambda)^{1/4}}\big)$.

In addition our technique also allows us to obtain a structural result regarding $\lambda$-edge-connected graphs with small diameter. We show that for every $\lambda > 1$, there are $\lambda$-edge-connected graphs $G$ with diameter $O(\log n)$ such that for any partition of the edges of $G$ into spanning subgraphs, all but $O(\log n)$ of the spanning subgraphs have diameter at least $\Omega(n)$ (in the case of unweighted multigraphs) or $\Omega(n/\lambda)$ (in the case of unweighted simple graphs). As a corollary, we also get that when sampling each edge of such a graph with probability $p \leq \gamma/\log n$ for a sufficiently small constant $\gamma > 0$, with at least a positive constant probability, the subgraph induced by the sampled edges has diameter at least $\Omega(n)$ (in the case of unweighted multigraphs) and $\Omega(n/\lambda)$ (in the case of unweighted simple graphs).

## 1.1 Related Work in the Centralized Setting

We next present a brief overview of the known results in the centralized setting. Starting in the 1950s [4, 7], the traditional approach to the minimum cut problem was to use max-flow algorithms (cf. [6] as a good reference and see [16, Section 1.3] for a concise summary). In the 1990s, three new approaches were introduced which go away from the flow-based method and lead to faster algorithms: The first method is based on a matroid characterization of the min-cut problem and was presented by Gabow [8]. Gabow's algorithm [8] finds a minimum cut in $O(m + \lambda^2 n \log \frac{n}{m})$ steps, for any unweighted (but possibly directed) graph with edge connectivity $\lambda$. The second approach applies to (possibly) weighted but undirected graphs and is based on repeatedly identifying and contracting edges outside the minimum cuts until a minimum cut becomes apparent (e.g., [10, 16, 22]). The beautiful *random contraction algorithm* (RCA) of Karger [10] falls into this category. In the basic version of RCA the following procedure is repeated $O(n^2 \log n)$ times: contract uniform random edges one by one until only two nodes remain. The edges between these two nodes correspond to a cut in the original graph, which is a minimum cut with probability at least $1/O(n^2)$. Karger and Stein [16] also present a more efficient implementation of the same basic idea, leading to total running time of $O(n^2 \log^3 n)$. The third method, which again applies to (possibly) weighted but undirected graphs, is due to Karger [14] and is based on a "semiduality" between minimum cuts and maximum spanning tree packings. This third method leads to the best known centralized minimum-cut algorithm [15] with running time $O(m \log^3 n)$.

For the approximation version of the problem (in undirected graphs), the main known results are as follows. Matula [21] presents an algorithm that finds a $(2 + \varepsilon)$-minimum cut for any constant $\varepsilon > 0$ in time $O((m+n)/\varepsilon)$. This algorithm is based on a graph search procedure called *maximum adjacency search*. Based on a modified version of the random contraction algorithm, Karger [13] presents an algorithm that finds a $(1 + \varepsilon)$-minimum cut in time $O(m + n \log^3 n/\varepsilon^4)$.

# 2 Preliminaries

**Notations and Definitions:** We usually work with an undirected weighted graph $G = (V, E, w)$, where $V$ is a set of $n$ vertices, $E$ is a set of (undirected) edges $e = \{v, u\}$ for $u, v \in V$, and $w : E \to \mathbb{R}^+$ is a mapping from edges $E$ to positive real numbers, i.e., for each edge $e \in E$, $w(e)$ denotes the weight of edge $e$. In the special case where we consider unweighted graphs, we simply assume $w(e) = 1$ for each edge $e \in E$.

For a given non-empty proper subset $C \subset V$, we define the cut $(C, V \setminus C)$ as the set of edges in $E$ that have one endpoint in set $C$ and the other endpoint in $V \setminus C$. The size of this cut, denoted by $w(C)$ is the sum of the weights of the edges in set $(C, V \setminus C)$. The edge-connectivity $\lambda(G)$ of the graph is defined as the minimum size of $w(C)$ as $C$ ranges over all nonempty proper subsets of $V$. A cut $(C, V \setminus C)$ is called $\alpha$-*minimum*, for an $\alpha \geq 1$, if $w(C) \leq \alpha\lambda(G)$. When clear from the context, we sometimes use $\lambda$ to indicate $\lambda(G)$.

**Communicaton Model and the Problem Statements:** We use a standard *message passing model* (a.k.a. the CONGEST model [23]), where the execution proceeds in synchronous rounds and in each round, each node can send a message of size $B$ to each of its neighbors. A typical value for $B$ is $B = \Theta(\log n)$ and for all our upper bounds, we will for simplicity assume that $B = \Theta(\log n)$.[2] For the upper bounds, we further assume that

---

[2]Note that by choosing $B = b \log n$ for some $b \geq 1$, in all our upper bounds, the term that does not depend on $D$ could be improved by a factor $\sqrt{b}$.

$B$ is large enough so that a constant number of node identifiers and edge weights can be packed into a single message. For $B = \Theta(\log n)$, this implies that for each edge $e \in E$, we have $\frac{w(e)}{n^{-\Theta(1)}} \in \{1, \ldots, n^{\Theta(1)}\}$. Without loss of generality, we will further assume that edge-weights are normalized and each edge-weight is in range $\{1, \ldots, n^{\Theta(1)}\}$. Thus, we can also view each weighted graph as a multi-graph with all edge-weights being 1 and with edge multiplicity at most $n^{\Theta(1)}$.

For our lower bounds, we assume a weaker model where, $B \cdot w(e)$ bits can be sent in each round over each edge $e$. To make sure that at least $B$ bits can be transmitted over each edge, we assume that the weights are scaled such that $w(e) \geq 1$ for each edge $e$. For integer weights, this is equivalent to assuming that the network graph is an unweighted multigraph where each edge $e$ of weight $w(e)$ corresponds to $w(e)$ parallel unit-weight edges.

In the problem of computing an *α-approximation of the minimum cut*, the goal is to find a cut $(C^*, V \setminus C^*)$ that is $\alpha$-minimum. To indicate this cut in the distributed setting, each node $v$ should know whether $v \in C^*$ or not. We also consider the problem of *α-approximation of the edge-connectivity*, where all nodes must output an estimate $\tilde{\lambda}$ of $\lambda$ such that $\tilde{\lambda} \in [\lambda, \lambda\alpha]$. In our randomized algorithms for these problems, time complexities are fixed deterministically and the correctness guarantees are required to hold with high probability.

## 2.1 Black-Box Algorithms

In this paper, we make frequent use of a *connected component identification* algorithm due to Thurimella [26], which itself builds on the minimum spanning tree algorithm of Kutten and Peleg [18]. Given a graph $G(V, E)$ and a subgraph $H = (V, E')$ such that $E' \subseteq E$, Thurimella's algorithm identifies the connected components of $H$ by assigning a label $\ell(v)$ to each node $v \in V$ such that two nodes get the same label iff they are in the same connected component of $H$. The time complexity of the algorithm is $O(D + \sqrt{n} \log^* n)$ rounds, where $D$ is the (unweighted) diameter of $G$. Moreover, it is easy to see that the algorithm can be made to produce labels $\ell(v)$ such that $\ell(v)$ is equal to the smallest (or the largest) id in the connected component of $H$ that contains $v$. Furthermore, using this "smallest-id in the component" version twice, once for graph $G$ with subgraph $G$ and once for graph $G$ with subgraph $H$, and then comparing the labels (and broadcasting an indicator symbol if they are unequal), nodes can know whether subgraph $H$ is connected or not, in the same time complexity $O(D + \sqrt{n} \log^* n)$. We refer to this as Thurimella's *connectivity-tester* algorithm. Finally, we remark that the same algorithms can also be used to solve $k$ independent instances of the connected component identification problem or $k$ independent instances of the connectivity-testing problem in $O(D + k\sqrt{n} \log^* n)$ rounds. This is achieved by pipe-lining the messages of the broadcast parts of different instances.

# 3 Random Edge Sampling and The Random Layering Technique

In this section, we study the process of randomly removing (or equivalently randomly keeping) edges of a arbitrary unweighted multigraph and we present a simple technique, which we call *random layering*, for analyzing the connectivity of the graph obtained through the edge sampling process. This technique also forms the basis of our main min-cut approximation algorithm that we present in the next section.

**Edge Sampling** Consider an arbitrary unweighted multigraph $G = (V, E)$. Given a probability $p \in [0, 1]$, we define an *edge sampling experiment* as follows: choose subset $S \subseteq E$ by including each edge $e \in E$ in set $S$ independently with probability $p$. We call the graph $G' = (V, S)$ *the sampled subgraph* in this experiment.

In this section, we use the *random layering technique* to provide an answer to the following *network reliability* question: "How large do we have to choose $p$ as a function of minimum-cut size $\lambda$ such that the sampled graph is connected with high probability?"[3] Note that for a fixed cut of size $\lambda$, the expected number of sampled edges is $\lambda p$ and if none of its edges gets sampled, then the sampled graph is disconnected. Thus, just considering a single cut of size $\lambda$, we see that if $p \leq \frac{1}{\lambda}$, then the probability that the sampled subgraph is connected is at most $\frac{1}{e}$. We show that $p \geq \frac{20 \log n}{\lambda}$ suffices to obtain a connected sampled graph with high probability. Note that this is non-trivial as a graph has exponential many cuts. It is easy to see that this bound is also asymptotically tight (see [20]).

**Theorem 3.1.** *Consider an arbitrary unweighted multigraph $G = (V, E)$ with edge connectivity $\lambda$ and choose subset $S \subseteq E$ by including each edge $e \in E$ in set $S$ independently with probability $p$. If $p \geq \frac{20 \log n}{\lambda}$, then the sampled subgraph $G' = (V, S)$ is connected with probability at least $1 - \frac{1}{n}$.*

---

[3]Alternatively, this question can be phrased as, how large should the edge-connectivity $\lambda$ of a network be such that it remains connected w.h.p. if each edge fails with probability $1 - p$.
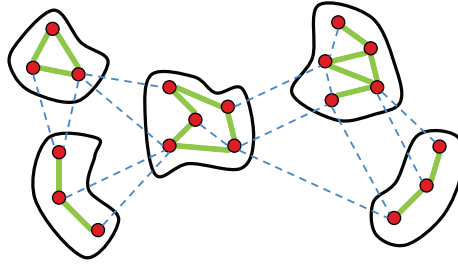
Figure 1: Graph $G_i$ and its connected components. The green solid links represent edges in $S_{i-}$, i.e., the sampled edges in layers 1 to $i$, and the blue dashed links represent $E \setminus S_{i-}$.

We remark that this result was known prior to this paper, via two different proofs: Lomonosov and Polesskii [20] and Karger [11]. The Lomonosov-Polesskii proof [20] shows that among the graphs of a given edge-connectivity $\lambda$, a cycle of length $n$ with edges of multiplicity $\lambda/2$ has the smallest probability of remaining connected under random sampling. This is achieved by coupling each given graph $G$ of edge-connectivity $\lambda$ with this special cycle-graph, and showing that throughout the process of sampling, as long as the special cycle-graph remains connected, $G$ is also connected. Karger's proof [11] uses the powerful fact that the number of $\alpha$-minimum cuts is at most $O(n^{2\alpha})$ and then uses basic probability concentration arguments (Chernoff and union bounds) to show that, w.h.p., each cut has at least one sampled edge. Note that graph has exponential many cuts. There are many known proofs for the $O(n^{2\alpha})$ upper bound on the number of $\alpha$-minimum cuts (see [15]); an elegant argument follows from Karger's *random contraction algorithm* [10].

Our proof of Theorem 3.1, which we present next, is simple and self-contained, and it is the only one of the three approaches that also extends to the case of random vertex failures[4] [1, Theorem 1.5].

*Proof of Theorem 3.1.* Let $L = 20 \log n$. For each edge $e \in E$, we independently choose a uniform random *layer number* from the set $\{1, 2, \ldots, L\}$. Intuitively, we add the sampled edges, layer by layer, and show that by adding the sampled edges of each layer, the number of connected components goes down by at least a constant factor, with at least a constant probability, and independently of the previous layers. Thus, after $L = \Theta(\log n)$ layers, connectivity is achieved with high probability.

To formalize this intuition, we first present some notations. For each $i \in \{1, \ldots, L\}$, let $S_i$ be the set of sampled edges with layer number $i$ and let $S_{i-} = \cup_{j=1}^{i} S_j$, the set of all sampled edges with a layer number in $\{1, \ldots, i\}$. Let $G_i = (V, S_{i-})$ and let $M_i$ be the number of connected components of graph $G_i$. To prove the theorem, we show that $M_L = 1$, w.h.p. Note that for any $i \in [1, L-1]$, since $S_{i-} \subseteq S_{(i+1)-}$, we have $M_{i+1} \leq M_i$. Consider the indicator variable $X_i$ such that $X_i = 1$ iff $M_{i+1} \leq 0.87 M_i$ or $M_i = 1$. As the core of the proof, we show the following claim, after which, a Chernoff bound completes the proof.

**Claim 3.2.** *For each layer $i \in [1, \ldots, L-1]$, and each subset $T \subseteq E$, we have $\Pr[X_i = 1 | S_{i-} = T] \geq 1/2$.*

To prove this claim, we use *the principle of deferred decisions* [17] to study the two random processes of sampling edges and choosing their layer numbers. More specifically, we consider the following process: first, each edge is sampled and given layer number 1 with probability $p/L$. Then, each remaining edge is sampled and given layer number 2 with probability $\frac{p/L}{1-p/L} \geq p/L$. Similarly, after determining the sampled edges of layers 1 to $i$, each remaining edge is sampled and given layer number $i+1$ with probability $\frac{p/L}{1-(i\,p)/L} \geq p/L$. At the end, i.e., after doing this for $L$ layers, any remaining edge is considered not sampled and it receives a random layer number from $\{1, 2, \ldots, L\}$. It is easy to see that in this process, each edge is independently sampled with probability exactly $p$ and each edge $e$ gets a uniform random layer number from $\{1, 2, \ldots, L\}$, chosen independently of the other edges and also independently of whether $e$ is sampled or not.

Now fix a layer $i \in [1, \ldots, L-1]$ and a subset $T \subseteq E$. Let $S_{i-} = T$ and consider graph $G_i = (V, S_{i-})$. Figure 1 presents an example graph $G_i$ and its connected components. If $M_i = 1$ meaning that $G_i$ is connected, then $X_i = 1$. Otherwise, suppose that $M_i \geq 2$. For each component $\mathcal{C}$ of $G_i$, call the component *bad* if

---

[4]In that problem, the question is, how large the vertex sampling probability $p$ has to be chosen, as a function of vertex connectivity $k$, so that the vertex-sampled graph is connected, w.h.p. The extension to the vertex version of the problem requires major modifications and leads to $p = \Omega(\frac{\log n}{\sqrt{k}})$ being a sufficient condition. Refer to [1, Section 3] for details.

$(\mathcal{C}, V \setminus \mathcal{C}) \cap S_{i+1} = \emptyset$. That is, $\mathcal{C}$ is bad if after adding the sampled edges of layer $i + 1$, $\mathcal{C}$ does not get connected to any other component. We show that $\Pr[\mathcal{C} \text{ is bad}] \leq \frac{1}{e}$. Since $G$ is $\lambda$-edge connected, we have $w(C) \geq \lambda$. Moreover, none of the edges in $(\mathcal{C}, V \setminus \mathcal{C})$ is in $S_{i-}$. Thus, using the principle of deferred decisions as described, each of the edges of the cut $(\mathcal{C}, V \setminus \mathcal{C})$ has probability $\frac{p/L}{1-(i\,p)/L} \geq p/L$ to be sampled and given layer number $i + 1$, i.e., to be in $S_{i+1}$. Since $p \geq \frac{20 \log n}{\lambda}$, the probability that none of the edges $(\mathcal{C}, V \setminus \mathcal{C})$ is in set $S_{i+1}$ is at most $(1 - p/L)^{\lambda} \leq 1/e$. Thus, $\Pr[\mathcal{C} \text{ is bad}] \leq \frac{1}{e}$.

Let $Z_i$ be the number of bad components of $G_i$. Since each component is bad with probability at most $1/e$, we have $\mathbb{E}[Z_i] \leq M_i/e$. Using Markov's inequality, we get $\Pr[Z_i \geq 2M_i/e] \leq 1/2$. Since each component that is not bad gets connected to at least one other component (when we look at graph $G_{i+1}$), we have $M_{i+1} \leq Z_i + \frac{(M_i - Z_i)}{2} = \frac{M_i + Z_i}{2}$. Therefore, with probability at least $1/2$, we have $M_{i+1} \leq \frac{1 + 2/e}{2} M_i < 0.87 M_i$. This means that $\Pr[X_i = 1] \geq 1/2$, which concludes the proof of Claim 3.2.

From Claim 3.2, we get that $\mathbb{E}[\sum_{i=1}^{L-1} X_i] \geq 10 \log n$. A Chernoff bound then shows that $\Pr[\sum_{i=1}^{L-1} X_i \geq 5 \log n] \geq 1 - \frac{1}{n}$. This means that with high probability, $M_L \leq \frac{n}{2^{10 \log n}} = 1$. That is, with high probability, $G_L = (V, S) = (V, S_{L-}) = G'$ is connected. This completes the proof. □

Theorem 3.1 provides a simple approach for finding an $O(\log n)$-approximation of the edge-connectivity of a network graph $G$ in $O(D + \sqrt{n} \log^2 n \log^* n)$ rounds. The proof appears in Appendix A.

**Corollary 3.3.** *There exists a distributed algorithm with round complexity $O(D + \sqrt{n} \log^2 n \log^* n)$ that, in any unweighted multi-graph $G = (V, E)$, finds an approximation $\tilde{\lambda}$ of edge-connectivity such that w.h.p. $\tilde{\lambda} \in [\lambda, \lambda \cdot \Theta(\log n)]$.*

# 4 Distributed Minimum Cut Approximation by Random Layering

In this section, we present our min-cut approximation algorithm that uses the random layering technique.

**Theorem 4.1.** *There is a distributed algorithm that, for any $\epsilon \in (0, 1)$, finds an $O(\epsilon^{-1})$-minimum cut in $O(D) + O(n^{0.5+\epsilon} \log^3 n \log \log n \log^* n)$ rounds, with high probability.*

We present the outline of the algorithm and all its major ideas. We defer putting the pieces together for getting the claimed round complexity to the proof of Theorem 4.1 in Appendix B.

## 4.1 Algorithm Outline

Our approximation algorithm is based on closely studying the sampled graph when the edge-sampling probability is between the two extremes of $\frac{1}{\lambda}$ and $\frac{\Theta(\log n)}{\lambda}$. Throughout this process, we collect a set $\mathcal{F}$ of $O(n \log n)$ cuts such that, with at least a 'reasonably large probability', $\mathcal{F}$ contains at least one 'small' cut.

**The Crux of the Algorithm:** Sample edges with probability $p = \frac{\epsilon \log n}{2\lambda}$ for a small $\epsilon \in (0, 1)$. Also, assign each edge to a random layer in $[1, \dots, L]$, where $L = 20 \log n$. For each layer $i \in [1, \dots, L - 1]$, let $S_i$ be the set of sampled edges of layer $i$ and let $S_{i-} = \cup_{j=1}^{i} S_j$. For each layer $i \in [1, \dots, L - 1]$, for each component $\mathcal{C}$ of graph $G_i = (V, S_{i-})$, add the cut $(\mathcal{C}, V \setminus \mathcal{C})$ to the collection $\mathcal{F}$. Since in each layer we add at most $n$ new cuts and there are $L = O(\log n)$ layers, we collect $O(n \log n)$ cuts in total. We show that with probability at least $n^{-\epsilon}/2$, at least one of the cuts in $\mathcal{F}$ is an $O(\epsilon^{-1})$-minimum cut. Note that repeating the experiment for $\Theta(n^{\epsilon} \log n)$ times is enough to get the guarantee that an $O(\epsilon^{-1})$-minimum cut is found with high probability.

**Theorem 4.2.** *Consider performing the above sampling and layering experiment with edge sampling probability $p = \frac{\epsilon \log n}{2\lambda}$ for $\epsilon \in (0, 1)$ and $L = 20 \log n$ layers. Then, $\Pr[\mathcal{F} \text{ contains an } O(\epsilon^{-1})\text{-minimum cut}] \geq n^{-\epsilon}/2$.*

*Proof of Theorem 4.2.* Fix an edge sampling probability $p = \frac{\epsilon \log n}{2\lambda}$ for a $\epsilon \in (0, 1)$ and let $\alpha = 40\epsilon^{-1}$. We say that a sampling and layering experiment is *successful* if $\mathcal{F}$ contains an $\alpha$-minimum cut or if the sampled graph $G_L = (V, S_{L-})$ is connected. We first show that each experiment is *successful* with probability at least $1 - \frac{1}{n}$. The proof of this part is very similar to that of Theorem 3.1.

For an arbitrary layer number $1 \leq i \leq L - 1$, consider graph $G_i = (V, S_{i-})$. If $M_i = 1$ meaning that $G_i$ is connected, then $G_L$ is also connected. Thus, in that case, the experiment is successful and we are done. In the more

interesting case, suppose $M_i \geq 2$. For each component $\mathcal{C}$ of $G_i$, consider the cut $(\mathcal{C}, V \setminus \mathcal{C})$. If any of these cuts is $\alpha$-minimum, then the experiment is successful as then, set $\mathcal{F}$ contains an $\alpha$-minimum cut. On the other hand, suppose that for each component $\mathcal{C}$ of $G_i$, we have $w(\mathcal{C}) \geq \alpha\lambda$. Then, for each such component $\mathcal{C}$, each of the edges of cut $(\mathcal{C}, V \setminus \mathcal{C})$ has probability $\frac{p/L}{1-(i\,p)/L} \geq p/L$ to be in set $S_{i+1}$ and since $w(\mathcal{C}) \geq \alpha\lambda$, where $\alpha = 20\epsilon^{-1}$, the probability that none of the edges of this cut in set $S_{i+1}$ is at most $(1-p/L)^{\alpha\lambda} \leq e^{\frac{p}{L}\cdot\alpha\lambda} = e^{-\frac{\epsilon\log n}{2\lambda}\cdot\frac{1}{L}\cdot\frac{40}{\epsilon}\cdot\lambda} = 1/e$. Hence, the probability that component $\mathcal{C}$ is *bad* as defined in the proof of Theorem 3.1 (i.e., in graph $G_{i+1}$, it does not get connected to any other component) is at most $1/e$. The rest of the proof can be completed exactly as the last paragraph of of the proof of Theorem 3.1, to show that $\Pr[\textit{experiment is successful}] \geq 1 - 1/n$.

Thus we have a bound on the probability that $\mathcal{F}$ contains an $\alpha$-minimum cut or that the sampled graph $G = (V, S_{L-})$ is connected. However, in Theorem 4.2, we are only interested in the probability of $\mathcal{F}$ containing an $\alpha$-minimum cut. Using a union bound, we know that

$$\Pr[\textit{experiment is successful}] \leq \Pr[\mathcal{F} \textit{ contains a } \alpha\textit{-minimum cut}] + \Pr[G_L \textit{ is connected}].$$

On the other hand, $\Pr[G_L \textit{ is connected}] \leq 1 - n^{-\epsilon}$ because, considering a single mininmum cut of size $\lambda$, the probability that none of the edges of this cut are sampled is $(1-\frac{\epsilon\log n}{2\lambda})^\lambda \geq n^{-\epsilon}$. In that case, the sampled subgraph is disconnected. Thus, we can conclude that $\Pr[\mathcal{F} \textit{ contains an } \alpha\textit{-minimum cut}] \geq (1 - 1/n) - (1 - n^{-\epsilon}) = n^{-\epsilon} - 1/n \geq n^{-\epsilon}/2$. $\qquad\square$

## 4.2   Testing the Cuts

So far we know that $\mathcal{F}$ contains an $\alpha$-minimum cut with a reasonable probability. We now need to devise a distributed algorithm to read (or test) the sizes of the cuts in $\mathcal{F}$ and find that $\alpha$-minimum cut, in $O(D) + \tilde{O}(\sqrt{n})$ rounds. In the remainder of this section, we explain our approach to this part.

Consider a layer $i \in [1, L-1]$ and the graph $G_i = (V, S_{i-})$. In a distributed setting, for each component $\mathcal{C}$ of $G_i$, $diam(\mathcal{C})$ rounds is clearly enough to read the size of the cut $(\mathcal{C}, V \setminus \mathcal{C})$ such that all the nodes in component $\mathcal{C}$ know this size. However, $diam(\mathcal{C})$ can be considerably larger than $D = diam(G)$ and thus, this method would not lead to a round complexity of $\tilde{O}(D + \sqrt{n})$. To overcome this problem, notice that we do not need to read the exact size of the cuts. Instead, it is enough to devise a *test* that *passes* w.h.p. if $w(\mathcal{C}) \leq \alpha\lambda$, and *does not pass* w.h.p. if $w(\mathcal{C}) \geq (1+\delta)\alpha\lambda$, for a small constant $\delta \in (0, 1/4)$. In the distributed realization of such a test, it would be enough if all the nodes in $\mathcal{C}$ consistently know whether the test passed or not. Next, we explain a simple algorithm for such a test. This test itself uses random edge sampling. Given such a test, in each layer $i \in [1, \ldots, L-1]$, we can test all the cuts and if any cut passes the test, meaning that, w.h.p., it is a $((1+\delta)\alpha)$-minimum cut, then we can pick one of the cuts.[5]

**Lemma 4.3.** *Given a subgraph $G' = (V, E')$ of the network graph $G = (V, E)$, a threshold $\kappa$ and $\delta \in (0, 1/4)$, there exists a randomized distributed cut-tester algorithm with round complexity $\Theta\big(D + \frac{1}{\delta^2}\sqrt{n}\log n\log^* n\big)$ such that, w.h.p., for each node $v \in V$, we have: Let $\mathcal{C}$ be the connected component of $G'$ that contains $v$. If $w(\mathcal{C}) \leq \kappa/(1+\delta)$, the test passes at $v$, whereas if $w(\mathcal{C}) \geq \kappa(1+\delta)$, the test does not pass at $v$.*

Pseudo-code is presented in Algorithm 2 in Appendix B. We first run Thurimella's connected component identification algorithm (refer to Section 2.1) on graph $G$ for subgraph $G'$, so that each node $v \in V$ knows the smallest id in its connected component of graph $G'$. Then, each node $v$ adopts this label $componentID$ as its own id (temporarily). Thus, nodes of each connected component of $G'$ will have the same id. Now, the test runs in $\Theta(\log^2 n/\delta^2)$ *experiments*, each as follows: in the $j^{th}$ experiment, for each edge $e \in E \setminus E'$, put edge $e$ in set $E_j$ with probability $p' = 1 - 2^{-\frac{1}{\kappa}}$. Then, run Thurimella's algorithm on graph $G$ with subgraph $H_j = (V, E' \cup E_j)$ and with the new ids twice, such that at the end, each node $v$ knows the smallest and the largest id in its connected component of $H_j$. Call these new labels $\ell_j^{min}(v)$ and $\ell_j^{max}(v)$, respectively. For a node $v$ of a component $\mathcal{C}$ of $G_i$, we have that $\ell_j^{min}(v) \neq v.id$ or $\ell_j^{max}(v) \neq v.id$ if and only if at least one of the edges of cut $(\mathcal{C}, V \setminus \mathcal{C})$ is sampled in $E_j$, i.e., $(\mathcal{C}, V \setminus \mathcal{C}) \cap E_j \neq \emptyset$. Thus, each node $v$ of each component $\mathcal{C}$ knows whether $(\mathcal{C}, V \setminus \mathcal{C}) \cap E_j \neq \emptyset$ or not. Moreover, this knowledge is consistent between all the nodes of component $\mathcal{C}$. After $\Theta(\log n/\delta^2)$ experiments, each node $v$ of component $\mathcal{C}$ considers the test *passed* iff $v$ noticed $(\mathcal{C}, V \setminus \mathcal{C}) \cap E_j \neq \emptyset$ in at most half of the experiments. We defer the calculations to the proof of Lemma 4.3 in Appendix B.

---

[5]This can be done for example by picking the cut which passed the test and for which the related component has the smallest id among all the cuts that passed the test.

# 5 Distributed Minimum Cut Approximation: Matula's Approach

In [21], Matula presents an elegant centralized algorithm that for any constant $\varepsilon > 0$, finds a $(2 + \varepsilon)$-minimum cut, and has running time $O(|V| + |E|)$. In this section, we explain how with the help of a few additional elements, this general approach can be used in the distributed setting, to find a $(2 + \varepsilon)$-minimum cut in $O\big((D + \sqrt{n}\log^* n)\log^2 n \log\log n \cdot \frac{1}{\varepsilon^5}\big)$ rounds.

We first need to introduce the concept of sparse certificates for edge connectivity.

**Definition 5.1.** *For a given unweighted multi-graph $H = (V_H, E_H)$ and a value $k > 0$, a set $E^* \subseteq E_H$ of edges is a* sparse certificate *for $k$-edge-connectivity of $H$ if (1) $|E^*| \leq k|V_H|$, and (2) for each edge $e \in E_H$, if there exists a cut $(\mathcal{C}, V \setminus \mathcal{C})$ of $H$ such that $|(\mathcal{C})| \leq k$ and $e \in (\mathcal{C}, V \setminus \mathcal{C})$, then we have $e \in E^*$.*

Thurimella [26] presents a simple distributed algorithm that finds a sparse certificate for $k$-edge-connectivity of network graph $G$ in $O(k(D + \sqrt{n}\log^* n))$ rounds. We modify this algorithm slightly to solve a generalized variant of the problem, which will be used in our min-cut approximation algorithm. We present the result that we get from the modification in the following lemma, but we defer the related details to Appendix C.

**Lemma 5.2.** *Let $E_c$ be a subset of the edges of the network graph $G$ and define the virtual graph $G' = (V', E')$ as the multi-graph that is obtained by contracting all the edges of $G$ that are in $E_c$. Using the modified version of Thurimella's certificate algorithm, we can find a set $E^* \subseteq E \setminus E_c$ that is a sparse certificate for $k$-edge-connectivity of $G'$, in $O(k(D + \sqrt{n}\log^* n))$ rounds.*

Following the general approach of Matula's centralized algorithm [21], and with the help of sparse certificate algorithm of Lemma 5.2 and random sparsification technique of Karger [11], we get the following result[6].

**Theorem 5.3.** *There is a distributed algorithm that, for any constant $\varepsilon > 0$, finds a $(2 + \varepsilon)$-minimum cut in $O\big((D + \sqrt{n}\log^* n)\log^2 n \log\log n \cdot \frac{1}{\varepsilon^5}\big)$ rounds.*

*Proof Sketch.* We assume that nodes know a $(1 + \varepsilon/10)$-factor approximation $\tilde{\lambda}$ of the edge connectivity $\lambda$, and explain a distributed algorithm with round complexity $O((D + \sqrt{n}\log^* n)\log^2 n \cdot \frac{1}{\varepsilon^4})$. Note that this assumption can be removed at the cost of a $\Theta(\frac{\log\log n}{\log(1+\varepsilon/10)}) = \Theta(\log\log n \cdot \frac{1}{\varepsilon})$ factor increase in round complexity by trying $\Theta(\frac{\log\log n}{\varepsilon})$ exponential guesses $\tilde{\lambda}(1 + \varepsilon/10)^i$ for $i \in [0, \Theta(\frac{\log\log n}{\varepsilon})]$ where $\tilde{\lambda}$ is an $O(\log n)$-approximation of the edge-connectivity, which can be found by Corollary 3.3.

For simplicity, we first explain an algorithm that finds a $(2+\varepsilon)$-minimum cut in $O(\lambda(D+\sqrt{n}\log^* n)\log n \cdot \frac{1}{\varepsilon^2})$ rounds. Then, we explain how to reduce the round complexity to $O((D + \sqrt{n}\log^* n)\log^2 n \cdot \frac{1}{\varepsilon^4})$.

Pseudo-code is given in Algorithm 3 in Appendix C. First, we compute a sparse certificate $E^*$ for $\tilde{\lambda}(1 + \varepsilon/5)$-edge-connectivity of $G$, using Thurimella's algorithm. Now consider the graph $H = (V, E \setminus E^*)$. We have two cases: either (a) $H$ has at most $|V|(1 - \varepsilon/10)$ connected components, or (b) there is a connected component $\mathcal{C}$ of $H$ such that $w(\mathcal{C}) \leq \frac{2\lambda(1+\varepsilon/10)(1+\varepsilon/5)}{1-\varepsilon/10} \leq (2 + \varepsilon)\lambda$. Note that if (a) does not hold, case (b) follows because $H$ has at most $(1 + \varepsilon/5)\tilde{\lambda}|V|$ edges.

In Case (b), we can find a $(2 + \varepsilon)$-minimum cut by testing the connected components of $H$ versus threshold $\kappa = \tilde{\lambda}(2 + \varepsilon/3)$, using the Cut-Tester algorithm presented in Lemma 4.3. In Case (a), we can solve the problem recursively on the virtual graph $G' = (V', E')$ that is obtained by contracting all the edges of $G$ that are in $E_c = E \setminus E^*$. Note that this contraction process preserves all the cuts of size at most $\tilde{\lambda}(1 + \varepsilon/5) \geq \lambda$ but reduces the number of nodes (in the virtual graph) at least by a $(1 - \varepsilon/10)$-factor. Consequently, $O(\log(n)/\varepsilon)$ recursions reduce the number of components to at most 2 while preserving the minimum cut.

The dependence on $\lambda$ can be removed by considering the graph $G_S = (V, E_S)$, where $E_S$ independently contains every edge of $G$ with probability $\Theta\big(\frac{\log n}{\varepsilon^2\lambda}\big)$. It can be shown that the edge connectivity of $G_S$ is $\Theta(\log(n)/\varepsilon^2)$ and a minimum edge cut of $G_S$ gives a $(1 + O(\varepsilon))$-minimum edge cut of $G$. The details appear in the full proof of the theorem in Appendix C. $\square$

---

[6]We remark that Matula [21] never uses the name *sparse certificate* but he performs *maximum adjacency search* which indeed generates a sparse certificate for edge connectivity.

# 6 Lower Bounds

As outlined in Sections 1 and 2, in the following, we describe a lower bound that allows to strengthen and generalize some of the lower bounds of Das Sarma et al. from [3]. Our lower bound uses the same basic approach as the lower bounds in [3]. The lower bounds of [3] are based on an $n$-node graph $G$ with diameter $O(\log n)$ and two distinct nodes $s$ and $r$. The proof deals with distributed protocols where node $s$ gets a $b$-bit input $x$, node $r$ gets a $b$-bit input $y$, and apart from $x$ and $y$, the initial states of all nodes is globally known. Slightly simplified, the main technical result of [3] (Simulation Theorem 3.1) states that if there is a randomized distributed protocol that correctly computes the value $f(x, y)$ of a binary function $f : \{0, 1\}^b \times \{0, 1\}^b \to \{0, 1\}$ with probability at least $1 - \varepsilon$ in time $T$ (for sufficiently small $T$), there is also a randomized $\varepsilon$-error two-party protocol for computing $f(x, y)$ with communication complexity $O(TB \log n)$. For our lower bounds, we need to extend the simulation theorem of [3] to a larger family of networks and to a slightly larger class of problems.

## 6.1 Generalized Simulation Theorem

**Distributed Protocols:** Given a weighted network graph $G = (V, E, w)$ ($\forall e \in E : w(e) \geq 1$), we consider distributed tasks for which each node $v \in V$ gets some private input $x(v)$ and every node $v \in V$ has to compute an output $y(v)$ such that the collection of inputs and outputs satisfies some given specification. To solve a given distributed task, the nodes of $G$ apply a distributed protocol. We assume that initially, each node $v \in V$ knows its private input $x(v)$, as well as the set of neighbors in $G$. Time is divided into synchronous rounds and in each round, every node can send at most $B \cdot w(e)$ bits over each of its incident edges $e$. We say that a given (randomized) distributed protocol solves a given distributed task with error probability $\varepsilon$ if the computed outputs satisfy the specification of the task with probability at least $1 - \varepsilon$.

**Graph Family $\mathcal{G}(n, k, c)$:** For parameters $n$, $k$, and $c$, we define the *family of graphs $\mathcal{G}(n, k, c)$* as follows. A weighted graph $G = (V, E, w)$ is in the family $\mathcal{G}(n, k, c)$ if and only if $V = \{1, \ldots, n\}$ and for all $h \in \{1, \ldots, n\}$, the total weight of edges between nodes in $\{1, \ldots, h\}$ and nodes in $\{h + k + 1, \ldots, n\}$ is at most $c$. We consider distributed protocols on graphs $G \in \mathcal{G}(n, k, c)$ for given $n$, $k$, and $c$. For an integer $\eta \geq 1$, we define $L_\eta := \{1, \ldots, \eta\}$ and $R_\eta := \{n - \eta + 1, \ldots, n\}$. Given a parameter $\eta \geq 1$ and a network $G \in \mathcal{G}(n, k, c)$, we say that a two-party protocol between Alice and Bob $\eta$-solves a given distributed task for $G$ with error probability $\varepsilon$ if a) initially Alice knows all inputs and all initial states of nodes in $V \setminus R_\eta$ and Bob knows all inputs and all initial states of nodes in $V \setminus L_\eta$, and b) in the end, Alice outputs $y(v)$ for all $v \in L_{n/2}$ and Bob outputs $y(v)$ for all $v \in R_{n/2}$ such that with probability at least $1 - \varepsilon$, all these $y(v)$ are consistent with the specification of the given distributed task. A two-party protocol is said to be *public coin* if Alice and Bob have access to a common random string. The proof of the following theorem appears in Appendix D.

**Theorem 6.1** (Generalized Simulation Theorem). *Assume we are given positive integers $n$, $k$, and $\eta$, a parameter $c \geq 1$, as well as a subfamily $\tilde{\mathcal{G}} \subseteq \mathcal{G}(n, k, c)$. Further assume that for a given distributed task and graphs $G \in \tilde{\mathcal{G}}$, there is a randomized protocol with error probability $\varepsilon$ that runs in $T \leq (n - 2\eta)/(2k)$ rounds. Then, there exists a public-coin two-party protocol that $\eta$-solves the given distributed task on graphs $G \in \tilde{\mathcal{G}}$ with error probability $\varepsilon$ and communication complexity at most $2BcT$.*

## 6.2 Information Dissemination Lower Bound

Our first application of Theorem 6.1 is a lower bound on a basic information dissemination task in $\lambda$-edge connected networks. We show that even if such networks have a small diameter, in general, for $s$ sufficiently large, disseminating $s$ bits of information requires time at least $\Omega(n/\lambda)$. As a corollary of our lower bound, we also obtain a lower bound on the diameter of the graph induced after independently sampling each edge of a $\lambda$-edge connected graph with some probability.

We start out by describing a generic construction to obtain graphs of the family $\mathcal{G}(n, k, c)$. Given some integer $n > 0$, we first define a fixed (unweighted) binary tree $T_n = (V, E_T)$ on the nodes $V = \{1, \ldots, n\}$. Hence, every node of $T_n$ has a (potentially empty) left subtree and a (potentially empty) right subtree and except for a distinguished root node, every node has one parent neighbor. Also, the structure of $T_n$ is chosen such that $T_n$ is as balanced as possible, i.e., the depth of $T_n$ is $\lceil \log_2 n \rceil$. Further the nodes $\{1, \ldots, n\}$ are placed in $T_n$ such that an in-order DFS traversal of $T_n$ (starting at the root) reproduces the natural order $1, 2, \ldots, n$. The tree $T_n$ can thus be seen as a binary search tree: Given any node $i$, for all nodes $j$ of the left subtree of $i$, it holds that $j < i$ and for all nodes $j$ of the right subtree of $i$, it holds that $j > i$.

**Lemma 6.2.** *Given an integer $p \in \{1, \ldots, n-1\}$, consider the cut $(S_p, V \setminus S_p)$, where $S_p = \{1, \ldots, p\}$. For every $p \in \{1, \ldots, n-1\}$, the number of edges between over the cut $(S_p, V \setminus T_p)$ is at most $\lceil \log_2 n \rceil$*

*Proof Sketch.* The depth of $T_n$ is $\lceil \log_2 n \rceil$. Further, on each of the $\lceil \log_2 n \rceil$ levels of $T_n$, at most one edge crosses the cut $(S_p, V \setminus S_p)$. □

Using the tree $T_n$, we can construct graphs from the family $\mathcal{G}(n, k, c)$ for $c = \lceil \log_2 n \rceil$. Let $\mathcal{H}(n, k)$ be the family of weighted graphs $H = (V, E_H, w_H)$ with node set $V = \{1, \ldots, n\}$ such that for all edges $\{i, j\} \in E_H$, $|j - i| \leq k$. Given a graph $H \in \mathcal{H}(n, k)$, we define a graph $G(H) = (V, E, w)$ with node set $V = \{1, \ldots, n\}$ as follows:

- The edge set $E$ of $G(H)$ is $E := E_H \cup E_T$.
- The weight $w(e)$ of an edge $e \in E$ is given as $w(e) := \max\{1, w_H(e)\}$.

**Lemma 6.3.** *Given a graph $H \in \mathcal{H}(n, k)$, graph $G(H) \in \mathcal{G}(n, k, c)$ for $c = \lceil \log_2 n \rceil$. Further, the diameter of $G(H)$ is $O(\log n)$.*

*Proof.* The fact that $G(H) \in \mathcal{G}(n, k, c)$ directly follows from the definition of the graph family $\mathcal{H}(n, k)$ and from Lemma 6.2. As $T_n$ is a subgraph of $G(H)$, the diameter of $G(H)$ is at most the diameter of $T_n$ which is $O(\log n)$. □

Based on this construction, we now state our first main lower bound result. Essentially, the following theorem follows from Theorem 6.1 by taking any $\lambda$-connected network $H \in \mathcal{H}(n, k)$ (choosing $k$ as small as possible), applying Lemma 6.3 to obtain a network $G \in \mathcal{G}(n, k, O(\log n))$, and by considering the problem of sending $K$ bits from node 1 to node $n$. A detailed proof appears in Appendix D.

**Theorem 6.4.** *For any $\lambda \geq 1$, there exist weighted simple $n$-node graphs (or equivalently unweighted multigraphs) $G = (V, E)$ with edge connectivity at least $\lambda$ and diameter $D = O(\log n)$ such that for two distinguished nodes $s, t \in V$, sending $K$ bits of information from $s$ to $t$ requires time at $\Omega\left(\min\left\{\frac{K}{B \log n}, n\right\}\right)$. For unweighted, simple graphs, a lower bound of $\Omega\left(\min\left\{\frac{K}{B \log n}, \frac{n}{\lambda}\right\}\right)$ holds for the same problem.*

As a corollary of Theorem 6.4, we also get an upper bound on the diameter of when partitioning a graph into edge-disjoint subgraphs. Again, for lack of space, the proof appears in Appendix D.

**Corollary 6.5.** *There are unweighted $\lambda$-edge connected simple graphs $G$ and unweighted $\lambda$-edge connected multigraphs $G'$ such that when partitioning the edges of $G$ (or $G'$) into $\ell \geq \gamma \log n$ spanning subgraphs, for a sufficiently large constant $\gamma$, at least one of the subgraphs has diameter $\Omega(n/\lambda)$ (in the case of $G$) and $\Omega(n)$ (in the case of $G'$).*

**Remark:** When partitioning the edges of a graph $G$ in a random way such that each edge is independently assigned to a uniformly chosen subgraph, each subgraph corresponds to the induced graph that is obtained if each edge of $G$ is sampled with probability $p = 1/\ell$. As a consequence, the diameter lower bounds of Corollary 6.5 also holds with at least constant probability when considering the graph obtained when sampling each edge of a $\lambda$-connected graph with probability $p = 1/\ell \leq 1/(\gamma \log n)$.

## 6.3 Lower Bound for Approximating Minimum Cut

It remains to prove our main lower bound results. We start by proving a lower bound on approximating minimum cut in weighted graphs (or equivalently in unweighted multigraphs).

**Theorem 6.6.** *In weighted graphs, for any $\alpha \geq 1$ and any $\lambda \geq 1$, computing an $\alpha$-approximate minimum cut requires at least $\Omega\left(D + \sqrt{n/(B \log n)}\right)$ rounds.*

*Proof.* We prove the theorem by reducing from the two-party set disjointness problem [2, 9, 25]. Assume that as input, Alice gets a set $X$ and Bob get a set $Y$ such that both $X$ and $Y$ are of size $p$ and the elements of $X$ and $Y$ are from a universe of size $O(p)$. It is known that for Alice and Bob need to exchange at least $\Omega(p)$ bits to determine whether $X$ and $Y$ are disjoint [9, 25]. This lower bound holds even for public coin randomized protocols with constant error probability and it also holds if Alice and Bob are given the promise that if $X$ and $Y$ intersect, they

intersect in exactly one element [25]. As a consequence, if Alice and Bob receive sets $X$ and $Y$ of size $p$ as inputs such that $|X \cap Y| = 1$, finding $X \cap Y$ also requires Alice and Bob to exchange $\Omega(p)$ bits.

Assume that there is a protocol to find an $\alpha$-minimum cut or to $\alpha$-approximate the size of a minimum cut in time $T$ with a constant error probability $\varepsilon$. In both cases, we show that Alice and Bob can use this protocol to efficiently solve set disjointness by simulating the distributed protocol on a specially designed network.

In the following, we first describe the construction of this network. Let $a$ and $b$ be two positive integer parameters. We construct a graph $G \in \mathcal{G}(n, \lambda, O(\log n))$ as follows. First, we construct a weighted graph $H = (V_H, E_H, w_H) \in \mathcal{H}(a, 1)$ as follows. The node set of $H$ is $V_H = \{1, \dots, a\}$ and there is an edge $e$ of weight $w_H(e) = \alpha\lambda + 1$ between nodes $i$ and $j$ if and only if $|i - j| = 1$. By Lemma 6.3, we can then get a graph $G(H) \in \mathcal{G}(a, 1, O(\log n))$. To get a graph $G$, we add $b$ additional copies of graph $H$. Call node $i$ in the original copy $(i, 0)$ and node $i$ in the $j^{th}$ additional copy node $(i, j)$. In each copy $j \geq 1$, we connect node $(1, j)$ with node $(1, 0)$ by an edge of weight $\lambda$. By renaming node $(i, j)$ to $\kappa(i, j) := j + (i - 1)(b + 1)$, we can see that graph $G$ is in $\mathcal{G}(a(b + 1), b + 1, O(\log n))$. In the following, let $n = a(b + 1)$ be the number of nodes of $G$. The first $b + 1$ nodes of $G$ are nodes $(1, j)$ for $0 \leq j \leq b$, the last $b + 1$ nodes of $G$ are nodes $(a, j)$ for $0 \leq j \leq b$. Note that graph $G$ is exactly $\lambda$-edge connected as any of the edges $\{(1, j), (1, 0)\}$ defines a cut of size $\lambda$. Note also that every cut which divides one of the copies of $H$ into two or more parts has size at least $\alpha\lambda + 1$.

Assume that Alice and Bob need to solve a set disjointness instance where $X \subset \{1, \dots, b\}$, $Y \subset \{1, \dots, b\}$, $|X \cap Y| \leq 1$, and $|X|, |Y| = \Omega(b)$. The graph $G$ is extended such that the minimum cut problem in $G$ represents the given set cover instance. For each $x \notin X$, the weight of the edge $\{(1, x), (1, 0)\}$ is increased to $\alpha\lambda + 1$. Further, for every $y \notin Y$, we add an edge $\{(a, y), (a, 0)\}$ of weight $\alpha\lambda + 1$. Now, if and only if $X \cap Y = \emptyset$, every copy of $H$ is connected to the first copy by a link of weight $\alpha\lambda + 1$. Therefore, if $X$ and $Y$ are disjoint, the size of a minimum cut is at least $\alpha\lambda + 1$ and if $X$ and $Y$ intersect, there is a cut of size $\lambda$.

Alice knows the initial states of nodes $(i, j)$ for all $i < a$ and thus for the nodes $(i, j)$ with $1 \leq \kappa(i, j) < n - b$ (i.e., all except the last $b + 1$ nodes) and Bob knows the initual states of nodes $(i, j)$ for all $i > 1$ and thus for the nodes $(i, j)$ with $b + 1 < \kappa(i, j) \leq n$ (i.e., all except the first $b + 1$ nodes). If we have $T < (n - 2)/(2(b + 1)) = O(n/b) = O(a)$ for the time complexity $T$ of the distributed minimum cut approximation protocol, Theorem 6.1 implies that Alice and Bob can $(b + 1)$-solve the distributed task of $\alpha$-approximating the minimum cut with total communication complexity at most $O(TB \log n)$. As a consequence, Alice and Bob can also solve the given set disjointness instance using the same protocol and from the known set disjointness communication complexity lower bound, we therefore get $TB \log n = \Omega(b)$. Choosing $a = \Theta(\sqrt{n/(B \log n)})$ and $b = \Theta(\sqrt{nB \log n})$ this implies the claimed lower bound for approximating the size of the minimum cut. Assuming that Alice and Bob already know that $|X \cap Y| = 1$, the communication complexity lower bound on finding $X \cap Y$ also implies the same lower bound for finding an $\alpha$-minimum cut, even if the size $\lambda$ of the minimum cut is known. $\qquad\square$

**Theorem 6.7.** *In unweighted, simple graphs, for any $\alpha \geq 1$ and any $\lambda \geq 1$, computing an $\alpha$-approximate minimum cut requires at least $\Omega\big(D + \sqrt{\frac{n}{B\sqrt{\alpha\lambda}\log n}}\big)$ rounds.*

*Proof Sketch.* The proof is essentially done in the same way as the proof of Theorem 6.6. We therefore only describe the differences between the proofs. Because in a simple unweighted graph, we cannot add edges with different weights and we cannot add multiple edges, we have to construct the graph differently. Let us first describe the simple, unweighted graph $H'$ corresponding to $H$ in the construction of Theorem 6.7. Instead of a being path of length $a$ with edges of weight $\alpha\lambda + 1$, $H'$ is a sequence of $a$ cliques of size $\lceil\sqrt{\alpha\lambda + 1}\rceil$. Adjacent cliques are connected by complete bipartite graphs (with at least $\alpha\lambda + 1$ edges). We again have $b + 1$ copies of $H'$, where copy 0 is augmented with a complete binary tree by using Lemma 6.3. Each edge $\{(1, 0), (1, j)\}$ of weight $\lambda$ is replaced by $\lambda$ edges between clique $(1, 0)$ and clique $(1, j)$. Edges of weight $\alpha\lambda + 1$ between nodes $(1, 0)$ and $(1, j)$ and between nodes $(a, 0)$ and $(a, j)$ are replaced by complete bipartite graphs between the respective cliques. Again, by simulating a minimum cut approximation algorithm on the constructed graph, Alice and Bob can solve a given set disjointness instance for a universe of size $b$. However, the number of nodes of the network in this case is $\Theta(ab\sqrt{\alpha\lambda})$ leading to the lower bound claimed by the theorem. $\qquad\square$

# 7 Acknowledgment

# References

[1] K. Censor-Hillel, M. Ghaffari, and F. Kuhn. A New Perspective on Vertex Connectivity. *arXiv, http://arxiv.org/abs/1304.4553*, 2013.

[2] A. Chattapodhyay and T. Pitassi. The story of set disjointness. *SIGACT News Complexity Theory Column*, 67, 2011.

[3] A. Das Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM Journal on Computing*, 41(5):1235–1265, 2012.

[4] P. Elias, A. Feinstein, and C. E. Shannon. Note on maximum flow through a network. *IRE Transactions on Information Theory IT-2*, pages 117–199, 1956.

[5] M. Elkin. Distributed approximation: a survey. *SIGACT News*, 35(4):40–57, 2004.

[6] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, USA, 2010 (first published in 1962).

[7] L. R. Ford and D. R. Fulkersonn. Maximal flow through a network. *Canad. J. Math.*, 8:399–404, 1956.

[8] H. N. Gabow. A matroid approach to finding edge connectivity and packing arborescences. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, STOC '91, pages 112–122, New York, NY, USA, 1991. ACM.

[9] B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discrete Math*, 5(4):545–557, 1992.

[10] D. R. Karger. Global min-cuts in $\mathcal{RNC}$, and other ramifications of a simple min-out algorithm. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, SODA '93, pages 21–30, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[11] D. R. Karger. Random sampling in cut, flow, and network design problems. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, STOC '94, pages 648–657, New York, NY, USA, 1994. ACM.

[12] D. R. Karger. Random sampling in cut, flow, and network design problems. In *Proc. 26th ACM Symp. on Theory of Computing (STOC)*, pages 648–657, 1994.

[13] D. R. Karger. Using randomized sparsification to approximate minimum cuts. In *Proc. 5th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 424–432, 1994.

[14] D. R. Karger. Minimum cuts in near-linear time. In *Proc. 28th ACM Symp. on Theory of Computing (STOC)*, pages 56–63, 1996.

[15] D. R. Karger. Minimum cuts in near-linear time. *J. ACM*, 47(1):46–76, Jan. 2000.

[16] D. R. Karger and C. Stein. An $\tilde{O}(n^2)$ algorithm for minimum cuts. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC '93, pages 757–765, New York, NY, USA, 1993. ACM.

[17] D. E. Knuth. *Stable Marriage and Its Relation to Other Combinatorial Problems: An Introduction to the Mathematical Analysis of Algorithms*. AMS, 1996.

[18] S. Kutten and D. Peleg. Fast distributed construction of $k$-dominating sets and applications. In *Proceedings of the fourteenth annual ACM symposium on Principles of distributed computing*, PODC '95, pages 238–251, New York, NY, USA, 1995. ACM.

[19] N. Linial. Locality in distributed graph algorithms. *SIAM J. on Computing*, 21(1):192–201, 1992.

[20] M. V. Lomonosov and V. P. Polesskii. Lower bound of network reliability. *Problems of Information Transmission*, 7:118–123, 1971.

[21] D. W. Matula. A linear time $2 + \varepsilon$ approximation algorithm for edge connectivity. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, SODA '93, pages 500–504, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[22] H. Nagamochi and T. Ibaraki. Computing edge-connectivity in multigraphs and capacitated graphs. *SIAM J. Discret. Math.*, 5(1):54–66, Feb. 1992.

[23] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.

[24] J. C. Picard and M. Queyranne. Selected applications of minimum cuts in networks. *INFOR.*, 20:19–39, 1982.

[25] A. A. Razborov. On the distributional complexity of disjointness. *Theor. Comp. Sci.*, 106:385–390, 1992.

[26] R. Thurimella. Sub-linear distributed algorithms for sparse certificates and biconnected components. *Journal of Algorithms*, 23(1):160 – 179, 1997.

# A  Missing Parts of Section 3

*Proof of Corollary 3.3.* The Pseudocode for the following algorithm is presented in Algorithm 1. We run $\Theta(\log^2 n)$ edge-sampling experiments: $\Theta(\log n)$ experiments for each sampling probability $p_j = 2^{-j}$ where $j \in [1, \Theta(\log n)]$. From Theorem 3.1, we know that, if $p_j \geq \Omega\left(\frac{\log n}{\lambda}\right)$, the sampled graph is connected with high probability. On the other hand, by focusing on just one minimum cut, we see that if $p_j \leq \frac{1}{\lambda}$, then the probability that the sampled graph is connected is at most $3/4$. Let $p^*$ be the smallest sampling probability $p_j$ such that at least $9/10$ of the sampling experiments with probability $p_j$ lead to sampled graph being connected. With high probability, $\tilde{\lambda} := \frac{1}{p^*}$ is an $O(\log n)$-approximation of the edge-connectivity. To check whether each sampled graph is connected, we use Thurimella's connectivity-tester (refer to Section 2.1), and doing that for $\Theta(\log^2 n)$ different sampled graphs requires $O(D + \sqrt{n} \log^2 n \log^* n)$ rounds. □

---

**Algorithm 1** An $O(\log n)$ Approximation Algorithm for the Edge-Connectivity

---

1: **for** $i = 1$ to $\log n$ **do**
2:      **for** $j = 1$ to $4 \log n$ **do**
3:          Choose subset $E_i^j \subseteq E$ by adding each edge $e \in E$ to $E_i^j$ independently with probability $2^{-i}$

4: Run Thurimella's connectivity-tester on graph $G$ with $\Theta(\log^2 n)$ subgraphs $H_i^j = (V, E_i^j)$, in $O(D + \sqrt{n} \log^2 n \log^* n)$ rounds.  ▷ Refer to Section 2.1 for Thurimella's connectivity-tester algorithm.

5: **for** $i = 1$ to $\Theta(\log n)$ **do**
6:      **for** $j = 1$ to $c \log n$ **do**
7:          **if** graph $G = (V, E_i^j)$ is connected **then**
8:              $X_i^j \leftarrow 1$
9:          **else**
10:             $X_i^j \leftarrow 0$
11:          $X_i \leftarrow \sum_{j=1}^{c \log n} X_i^j$
12: $i^* \leftarrow \arg\max_{i \in [1, \Theta(\log n)]}(X_i \geq 9c \log n / 10)$
13: $\tilde{\lambda} \leftarrow 2^{i^*}$

14: Return $\tilde{\lambda}$

---

# B  Missing Parts of Section 4

---

**Algorithm 2** Distributed cut tester vs. threshold $\kappa$ @ node $v$

---

     Given a subgraph $G' = (V, E')$ where $E' \subseteq E$, and a threshold $\kappa$

1: $v.componentID \leftarrow$ the smallest id in the component of $G'$ that contains $v$      ▷ Using Thurimella's Component Identification Alg.

2: **for** $j = 1$ **to** $c \log(n)/\delta^2$ **do**
3:      Choose subset $E_i \subseteq E \setminus E'$ by adding each edge $e \in E \setminus E'$ to $E_j$ independently with probability $1 - 2^{-\frac{1}{\kappa}}$

4: $\ell_j^{max}(v) \leftarrow$ the largest *componentID* in the connected component of $H_i = (V, E' \cup E_i)$ that contains $v$
5: $\ell_j^{min}(v) \leftarrow$ the smallest *componentID* in the connected component of $H_i = (V, E' \cup E_i)$ that contains $v$
                 ▷ Using Thurimella's Component Identification on the $\Theta(\log n)$ values of $i$, simultaneously. (cf. Section 2.1)

6: $X_i \leftarrow 0$
7: **for** $i = 1$ **to** $\alpha \log n$ **do**
8:      **if** $\ell_j^{max}(v) \neq v.componentID$ **or** $\ell_j^{min}(v) \neq v.componentID$ **then** $X_i \leftarrow X_i + 1$

9: Test passes @ node $v$ iff $X_i \leq \frac{c \log n}{2\delta^2}$

---

*Proof of Lemma 4.3.* If a cut $(\mathcal{C}, V \setminus \mathcal{C})$ has size at most $\kappa/(1+\delta)$, then the probability that $(\mathcal{C}, V \setminus \mathcal{C}) \cap E_j \neq \emptyset$ is at most $1 - (1 - p')^{\frac{\kappa}{1+\delta}} = 1 - 2^{-\frac{1}{1+\delta}} \leq 0.5 - \frac{\delta}{4}$. On the other hand, if cut $(\mathcal{C}, V \setminus \mathcal{C})$ has size at least $((1+\delta)\kappa)$,

then the probability that $(\mathcal{C}, V \setminus \mathcal{C}) \cap E_j \neq \emptyset$ is at least $1 - (1 - p')^{(1+\delta)\kappa} \geq 1 - 2^{-1+\delta} \geq 0.5 + \frac{\delta}{4}$. This $\Theta(\delta)$ difference between these probabilities gives us our basic tool for distinguishing the two cases. Since we repeat the experiment presented in Section 4.2 for $\Theta(\frac{\log n}{\delta^2})$ times, an application of Hoeffding's inequality shows that if cut $(\mathcal{C}, V \setminus \mathcal{C})$ has size at most $\kappa/(1 + \delta)$, the test passes w.h.p., and if cut $(\mathcal{C}, V \setminus \mathcal{C})$ has size at least $\kappa(1 + \delta)$, then, w.h.p., the test does not pass. $\qquad \square$

Here, we explain how to put the pieces presented in Section 4 together to get the claim of Theorem 4.1.

*Proof of Theorem 4.1.* For simplicity, we first explain an $O(\varepsilon^{-1})$ minimum-cut approximation algorithm with time complexity $O((D + \sqrt{n} \log^* n \log n) n^\epsilon \log^2 n \log \log n)$. Then, explain how to reduce it to the claimed bound of $O(D) + O(n^{0.5+\epsilon} \log^3 n \log \log n \log^* n)$ rounds.

We first find an $O(\log n)$ approximation $\tilde{\lambda}$ of $\lambda$, using Corollary 3.3, in time $O(D) + O(\sqrt{n} \log^*) \log^2 n)$. This complexity is subsumed by the complexity of the later parts. After this, we use $\Theta(\log \log n)$ guesses for a 2-approximation of $\lambda$ in the form $\lambda'_i = \tilde{C} 2^i$ where $i \in [-\Theta(\log \log n), \Theta(\log \log n)]$. For each such guess $\lambda'_i$, we have $n^\epsilon \log n$ *epochs* as follows:

In each *epoch*, we sample edges with probability $p = \frac{\epsilon \log n}{2\lambda'}$ and assign each edge to a random layer in $[1, \ldots, L]$, where $L = 20 \log n$. For each layer $i \in [1, \ldots, L - 1]$, we let $S_i$ be the set of sampled edges of layer $i$ and let $S_{i-} = \cup_{j=1}^i S_j$. Then, for each $i \in [1, \ldots, L]$, we use the Cut-Tester Algorithm 2 on graph $G$ with subgraph $G_i = (V, S_{i-})$, threshold $\kappa = 50\lambda'/\epsilon$, and with parameter $\delta = 1/8$. This takes $O((D + \sqrt{n} \log n \log^* n) \log n)$ rounds (for each layer). If in a layer, a component passes the test, it means its cut has size at most $O(\lambda'/\epsilon)$, with high probability. To report the results of the test, we construct a BFS tree rooted in a leader in $O(D)$ rounds and we convergecast the minimum *componentID* that passed the test, in time $O(D)$. We then broadcast this *componentID* to all nodes and all nodes that have this *componentID* define the cut that is $O(\lambda'/\epsilon)$-minimum, with high probability.

Over all the guesses, we know that there is a guess $\lambda'_j$ that is a 2-approximation of $\lambda$. In that guess, from Theorem 4.2 and a Chernoff bound, we know that at least one cut that is an $O(\epsilon^{-1})$-minimum cut will pass the test. We stop the process in the smallest guess for which a cut passes the test.

Finally, to reduce the time complexity to $O(D) + O(n^{0.5+\epsilon} \log^3 n \log \log n \log^* n)$ rounds, note that we can parallelize (i.e., pipeline) the $\Theta(n^\epsilon \log^2 n \log \log n)$ runs of Cut-Testing algorithm, which come from $\Theta(\log \log n)$ guesses $\lambda'_i$, $n^\epsilon \log n$ epochs for each guess, and $\Theta(\log n)$ layers in each epoch. We can do this pipe-lining simply because these instances of Cut-Testing do not depend on the outcomes of each other and $k$ instances of Thurimella's algorithms can be run together in time $O(D + k\sqrt{n} \log^* n)$ rounds (refer to Section 2.1). To output the final cut, when doing the convergecast of the Cut-Testing results on the BFS, we append the edge-connectivity guess $\lambda'_j$, epoch number, and layer number to the *componentID*. Then, instead of taking minimum on just *componentID*, we choose the *componentID* that has the smallest tuple (guess $\lambda'_j$, epoch number, layer number, *componentID*). Note that the smallest guess $\lambda'_j$ translates to the smallest cut size, and the other parts are simply for tie-breaking. $\qquad \square$

# C  Missing Parts of Section 5

*Proof of Lemma 5.2.* The idea of Thurimella's original sparse certificate-algorithm [26] is relatively simple: $E^*$ is made of the edges of $k$ MSTs that are found in $k$ iterations. Initially, we set $E^* = \emptyset$. In each iteration, we assign weight 0 to the edges in $E \setminus E^*$ and weight 1 to the edges in $E^*$. In each iteration, we find a new MST with respect to the new weights using the MST algorithm of [18], and add the edges of this MST to $E^*$. Because of the weights, each MST tries to avoid using the edges that are already in $E^*$. In particular, if in one iteration, there exist two edges $e, e'$, a cut $(\mathcal{C}, V \setminus \mathcal{C})$ such that $e, e' \in (\mathcal{C}, V \setminus \mathcal{C})$ and $e \in E^*$ but $e' \notin E^*$, then the new MST will not contain $e$ but will contain an edge $e'' \in (E \setminus E^*) \cap (\mathcal{C}, V \setminus \mathcal{C})$. This is because, MST will prefer $e''$ to $e$ and there is at least one such $e''$, namely edge $e'$. As a result, if there is a cut with size at most $k$, in each MST, at least one edge of the cut gets added to $E^*$, until all edges of the cut are in $E^*$.

To solve our generalized version of sparse certificate, we modify the algorithm in the following way. As before, we construct the set $E^*$ iteratively such that at the beginning $E^* = \emptyset$. In each iteration, we give weight 0 to edges of $E_c$, weight 1 to edges of $E \setminus (E_c \cup E^*)$ and weight 2 to edges in $E^*$. Moreover, in each iteration, if the newly found MST is $T$, we only add edges in $T \setminus E_c$ to the set $E^*$. Note that if for an edge $e = \{u, v\} \in E$,

nodes $u$ and $v$ correspond to the same node of the edge-contracted graph $G'$, then edge $e$ will never be added to $E^*$ as either it is in $E_c$ or $u$ and $v$ are connected via a path made of edges in $E_c$ and thus, in each MST, that path is always preferred to $e$. Moreover, if there is a cut $(\mathcal{C}, V \setminus \mathcal{C})$ of $G$ such that $(\mathcal{C}, V \setminus \mathcal{C}) \cap E_c = \emptyset$ and there are two edges $e, e' \in (\mathcal{C}, V \setminus \mathcal{C})$ such that $e \in E^*$ but $e' \notin E^*$, then the new MST will not contain $e$ but will contain an edge $e'' \in (E \setminus E^*) \cap (\mathcal{C}, V \setminus \mathcal{C})$. $\qquad\square$

---

**Algorithm 3** $(2 + \varepsilon)$-minimum cut approximation: Matula's Approach

---

Given a $(1 + \varepsilon/10)$-factor approximation $\tilde{\lambda}$ of $\lambda$

1: $E_c \leftarrow \emptyset, E^* \leftarrow E, \eta_{old} \leftarrow n, \eta_{new} \leftarrow 1$
2: **while** $(\eta \geq 2)$ & $(\eta_{new} \leq \eta_{old}(1 - \varepsilon/10))$ **do**
3: $\quad E_c \leftarrow E \setminus E^*$
4: $\quad E^* \leftarrow$ a sparse certificate for $\tilde{\lambda}(1 + \varepsilon/5)$-edge-connectivity of graph $G' = (V', E')$ obtained by contracting edges of $E_c$
5: $\quad \eta_{new} \leftarrow$ number of connected components of subgraph $H = (V, E \setminus E^*)$
6: **endwhile**
7: Test cuts defined by connected components of graph $H = (V, E \setminus E^*)$ versus threshold $\kappa = \tilde{\lambda}(2 + \varepsilon/3)$
8: Output the component that passes the test and contains the smallest id between such components

---

*Proof of Theorem 5.3.* We assume that nodes know a $(1 + \varepsilon/10)$-factor approximation $\tilde{\lambda}$ of the edge connectivity $\lambda$, and explain a distributed algorithm with round complexity $O((D + \sqrt{n} \log^* n) \log^2 n \cdot \frac{1}{\varepsilon^4})$. Note that this assumption can be removed at the cost of a $\Theta(\frac{\log \log n}{\log(1 + \varepsilon/10)}) = \Theta(\log \log n \cdot \frac{1}{\varepsilon})$ factor increase in round complexity by trying $\Theta(\frac{\log \log n}{\varepsilon})$ exponential guesses $\tilde{\lambda}(1 + \varepsilon/10)^i$ for $i \in [0, \Theta(\frac{\log \log n}{\varepsilon})]$ where $\tilde{\lambda}$ is an $O(\log n)$-approximation of the edge-connectivity, which can be found by Corollary 3.3.

For simplicity, we first explain an algorithm that finds a $(2 + \varepsilon)$-minimum cut in $O(\lambda(D + \sqrt{n} \log^* n) \log n \cdot \frac{1}{\varepsilon^2})$ rounds. Then, we explain how to reduce the round complexity to $O((D + \sqrt{n} \log^* n) \log^2 n \cdot \frac{1}{\varepsilon^4})$.

Pseudo-code is given in Algorithm 3 in Appendix C. First, we compute a sparse certificate $E^*$ for $\tilde{\lambda}(1 + \varepsilon/5)$-edge-connectivity for $G$, using Thurimella's algorithm. Now consider the graph $H = (V, E \setminus E^*)$. We have two cases: either (a) $H$ has at most $|V|(1 - \varepsilon/10)$ connected components, or (b) there is a connected component $\mathcal{C}$ of $H$ such that $w(\mathcal{C}) \leq \frac{2\lambda(1 + \varepsilon/10)(1 + \varepsilon/5)}{1 - \varepsilon/10} \leq (2 + \varepsilon)\lambda$. Note that if (a) does not hold, case (b) follows because $H$ has at most $(1 + \varepsilon/5)\tilde{\lambda}|V|$ edges.

In Case (b), we can find a $(2 + \varepsilon)$-minimum cut by testing the connected components of $H$ versus threshold $\kappa = \tilde{\lambda}(2 + \varepsilon/3)$, using the Cut-Tester algorithm presented in Lemma 4.3. In Case (a), we can solve the problem recursively on the virtual graph $G' = (V', E')$ that is obtained by contracting all the edges of $G$ that are in $E_c = E \setminus E^*$. Note that this contraction process preserves all the cuts of size at most $\tilde{\lambda}(1 + \varepsilon/5) \geq \lambda$ but reduces the number of nodes (in the virtual graph) at least by a $(1 - \varepsilon/10)$-factor. Consequently, $O(\log(n)/\varepsilon)$ recursions reduce the number of components to at most 2 while preserving the minimum cut.

The dependence on $\lambda$ can be removed by considering the graph $G_S = (V, E_S)$, where $E_S$ independently contains every edge of $G$ with probability $\Theta(\frac{\log n}{\varepsilon^2 \lambda})$. It can be shown that the edge connectivity of $G_S$ is $\Theta(\log(n)/\varepsilon^2)$ and a minimum edge cut of $G_S$ gives a $(1 + O(\varepsilon))$-minimum edge cut of $G$.

We now explain how to remove the dependence on $\lambda$ from the time complexity. Let $E_S$ be a subset of the edges of $G = (V, E)$ where each $e \in E$ is independently included in $E_S$ with probability $p = \frac{100 \log n}{\varepsilon^2} \cdot \frac{1}{\lambda}$. Then, using the edge-sampling result of Karger [11, Theorem 2.1][7], we know that with high probability, for each $\mathcal{C} \subseteq V$, we have

$$(1 - \varepsilon/3) \cdot |(\mathcal{C}, V \setminus \mathcal{C})| \cdot p \leq |(\mathcal{C}, V \setminus \mathcal{C}) \cap E_S| \leq (1 + \varepsilon/3) \cdot |(\mathcal{C}, V \setminus \mathcal{C})| \cdot p.$$

Hence, in particular, we know that graph $G_{new} = (V, E_S)$ has edge connectivity at least $\lambda p(1 - \varepsilon/3)$ and at most $\lambda p(1 + \varepsilon/3)$, i.e., $\lambda_{new} = \Theta(\log n \cdot \frac{1}{\varepsilon^2})$. Moreover, for every cut $(\mathcal{C}, V \setminus \mathcal{C})$ that is a $(1 + \varepsilon/3)$-minimum cut in graph $G_{new}$, we have that $(\mathcal{C}, V \setminus \mathcal{C})$ is a $(1 + \varepsilon)$-minimum cut in graph $G$. We can therefore solve the cut-approximation

---

[7]We emphasize that this result is non-trivial. The proof follows from the powerful bound of $O(n^{2\alpha})$ on the number of $\alpha$-minimum cuts [10] and basic concentration arguments (Chernoff and union bounds).

problem in graph $G_{new}$, where we only need to use sparse certificates for $\Theta(\log n \cdot \frac{1}{\varepsilon^2})$ edge-connectivity[8]. The new round complexity becomes $O\big((D + \sqrt{n} \log^* n) \log^2 n \cdot \frac{1}{\varepsilon^4}\big)$ rounds.

The above round complexity is assuming a $(1 + \varepsilon/10)$-approximation of edge-connectivity is known. Substituting this assumption with trying $\Theta(\log \log n/\varepsilon)$ guesses around the $O(\log n)$ approximation obtained by Corollary 3.3 (and outputting the smallest found cut) brings the round complexity to the claimed bound of $O((D + \sqrt{n} \log^* n) \log^2 n \log \log n \cdot \frac{1}{\varepsilon^5})$. □

# D   Missing Parts of Section 6

*Proof of Theorem 6.1.* We show that Alice and Bob can simulate an execution of the given distributed protocol to obtain outputs that are consistent with the specification of the given distributed task. First note that a randomized distributed algorithm can be modeled as a deterministic algorithm where at the beginning, each node $v$ receives s sufficiently large random string $r(v)$ as additional input. Assume that $R$ is the concatenation of all the random strings $r(v)$. Then, a randomized distributed protocol with error probability $\varepsilon$ can be seen as a deterministic protocol that computes outputs the satisfy the specification of the given task with probability at least $1 - \varepsilon$ over all possible choices of $R$. (A similar argument has also been used, e.g., in [3, 19]).

Alice and Bob have access to a public coin giving them a common random string of arbitrary length. As also the set of nodes $V = \{1, \ldots, n\}$ of $G$ is known, Alice and Bob can use the common random string to model $R$ and to therefore consistently simulate all the randomness used by all $n$ nodes in the distributed protocol. Given $R$, it remains for Alice and Bob to simulate a deterministic protocol. If they can (deterministically) compute the outputs of some nodes of a given deterministic protocol, they can also compute outputs for a randomized protocol with error probability $\varepsilon$ such that the outputs are consistent with the specification of the distributed task with probability at least $1 - \varepsilon$.

Given a deterministic, distributed protocol on a graph $G \in \tilde{\mathcal{G}}$ with time complexity $T \leq (n - 2\eta)/(2k)$, we now describe a two-party protocol with communication complexity at most $2cB$ in which for each round $r \in \{0, \ldots, T\}$,

(I)  Alice computes the states of all nodes $i \leq n - \eta - r \cdot k$ at the end of round $r$, and

(II)  Bob computes the states of all nodes $i > \eta + r \cdot k$ at the end of round $r$.

Because the output $y(u)$ of every node $u$ is determined by $u$'s state after $T$ rounds, together with the upper bound on $T$, (I) implies that Alice can compute the outputs of all nodes $i \leq n/2$ and Bob can compute the outputs of all nodes $i > n/2$. Therefore, assuming that initially, Alice knows the states of node $i \leq n - \eta$ and Bob knows the states of nodes $i > \eta$, a two-party protocol satisfying (I) and (II) $\eta$-solves the distributed task solved by the given distributed protocol. In order to prove the claim of the theorem, it is thus sufficient to show that there exists a deterministic two-party protocol with communication complexity at most $2BcT$ satisfying (I) and (II).

In a deterministic algorithm, the state of a node $u$ at the end of a round $r$ (and thus at the beginning of round $r + 1$) is completely determined by the state of $u$ at the beginning of round $r$ and by the messages node $u$ receives in round $r$ from its neighbors. We prove I) and II) by induction on $r$. First note that (interpreting the initial state as the state after round 0), (I) and (II) are satisfied by the assumption that initially, Alice knows the initial states of all nodes $1, \ldots, n - \eta$ and Bob knows the initial states of all nodes $\eta + 1, \ldots, n$. Next, assume that (I) and (II) hold for some $r = r' \in \{0, \ldots, T - 1\}$. Based on this, we show how to construct a protocol with communication complexity at most $2Bc$ such that (I) and (II) hold for $r = r'+1$. We formally show how, based on assuming I) and (II) for $r = r'$, Alice can compute the states of nodes $i \leq n - \eta - (r' + 1)k$ using only $Bc$ bits of communication. The argument for Bob is done in exactly the same way. In order to compute the state of a node $i \leq n-\eta-(r'+1)k$ at the end of round $r' + 1$, Alice needs to know the state of node $i$ at the beginning of round $r' + 1$ (i.e., at the end of round $r'$) and the message sent by each neighbor $j$ in round $r' + 1$. Alice knows the state of $i$ at the beginning of round $r'$ and the messages of neighbors $j \leq n - \eta - r'k$ by the assumption that Alice already knows the states of all nodes $i \leq n - \eta - r'k$ at the end of round $r'$. By the definition of the graph family $\mathcal{G}(n, k, c)$, the total weight of edges between nodes $i \leq n - \eta - (r' + 1)k$ and nodes $j > n - \eta - r'k$ is at most $c$. The number of bits sent over these edges in round $r' + 1$ is therefore at most $cB$. Further, by assuming that also (II) holds for $r = r'$, Bob knows the states of all nodes $j' > \eta + r'k$. We have $n - \eta - r'k \geq n - \eta - (T - 1)k \geq n/2 + k > \eta + r'k$.

---

[8]Note that, solving the cut approximation on the virtual graph $G_{new}$ formally means that we set the weight of edges outside $E \setminus E_0$ equal to zero. However, we still use graph $G$ to run the distributed algorithm and thus, the round complexity depends on $diam(G) = D$ and not on the possibly larger $diam(G_{new})$.

Consequently, Bob has all the information to send Alice the missing $cB$ bits needed to compute the states of nodes $i \le n - \eta - (r' + 1)k$ at the end of round $r' + 1$. $\qquad\square$

*Proof of Theorem 6.4.* We consider the weighted simple graph $H = (V, E_H, w_H)$ with node set $V = \{1, \ldots, n\}$, edge set $E_H = \{i, j \in V : |i - j| = \lambda\}$, and $w_H(e) = \lambda$ for all $e \in E_H$, as well as the unweighted simple graph $H' = (V, E'_H)$ with node set $V = \{1, \ldots, n\}$ and edge set $E'_H = \{i, j \in V : |i - j| \le \lambda\}$. We have $H \in \mathcal{H}(n, 1)$ and $H' \in \mathcal{H}(n, \lambda)$ and therefore by Lemma 6.3, we also have $G(H) \in \mathcal{G}(n, 1, O(\log n))$ and $G(H') \in \mathcal{G}(n, \lambda, O(\log n))$. Further, $H$, as well as $H'$ are $\lambda$-edge connected and thus also $G(H)$ and $G(H')$ are at least $\lambda$-edge connected. In both cases, we choose the two distinguised nodes $s$ and $t$ as $s = 1$ and $t = n$. Sending $K$ bits from $s$ to $t$ can then be modelled as the following distributed task in $G(H)$. Initially node 1 gets an arbitrary $K$-bit string as input. The input of every other node of $G(H)$ is the empty string. To solve the task, the output of node $n$ has to be equal to the input of node 1 and all other nodes need to output the empty string.

Assume that there is a (potentially randomized) distributed protocol that solves the given information dissemination task in $T$ rounds with error probability $\varepsilon$. For $T \le (n - 2)/2$ (in the case of $H$) and $T \ge (n - 2)/(2\lambda)$ (in the case of $H'$), Theorem 6.1 therefore implies that there exists a public-coin two-party protocol that 1-solves the given task with communication complexity at most $O(BT \log n)$. In such a protocol, only Alice gets the input of node 1 and Bob has to compute the output of node $n$. Consequently Alice needs to send $K$ bits to Bob and we therefore need to have $BT \log n = \Omega(K)$. Together with the upper bounds on $T$ required to apply Theorem 6.1, the claim of the theorem then follows. $\qquad\square$

*Proof of Corollary 6.5.* Assume that we are given a graph $G = (V, E)$ and a partition of the edges $E$ into $\ell$ spanning subgraphs such that each subgraph has diameter at most $D$. Consider two nodes $s$ and $t$ of $G$. We can use the partition of $E$ to design a protocol for sending $K$ bits from $s$ to $t$ in $O(D + K/\ell)$ rounds as follows. The $K$ bits are divided into equal parts of sieze $K/\ell$ bits. Each part is in parallel sent on one of the $\ell$ parts. Using pipelining this can be done in time $T = O(D + K/\ell)$. From Theorem 6.4, we know that for simple graphs, $T = \Omega\left(\min\left\{\frac{K}{B \log n}, \frac{n}{\lambda}\right\}\right)$. Choosing $K = Bn \log(n)/\lambda$ and $\gamma$ sufficiently small then implies the claimed lower bound on $D$. The argument for multigraphs is done in the same way by using the stronger respective lower bound in Theorem 6.4 $\qquad\square$