Small Cuts and Connectivity Certificates: A Fault Tolerant Approach

Merav Parter
Weizmann Institute, Rehovot, Israel
merav.parter@weizmann.ac.il

Abstract

We revisit classical connectivity problems in the CONGEST model of distributed computing. By using techniques from fault tolerant network design, we show improved constructions, some of which are even "local" (i.e., with $\widetilde{O}(1)$ rounds) for problems that are closely related to hard global problems (i.e., with a lower bound of $\Omega(Diam + \sqrt{n})$ rounds).

Distributed Minimum Cut: Nanongkai and Su presented a randomized algorithm for computing a $(1+\epsilon)$-approximation of the minimum cut using $\widetilde{O}(D + \sqrt{n})$ rounds where D is the diameter of the graph. For a sufficiently large minimum cut $\lambda = \Omega(\sqrt{n})$, this is tight due to Das Sarma et al. [FOCS '11], Ghaffari and Kuhn [DISC '13].

- **Small Cuts:** A special setting that remains open is where the graph connectivity λ is small (i.e., constant). The only lower bound for this case is $\Omega(D)$, with a matching bound known only for $\lambda \leq 2$ due to Pritchard and Thurimella [TALG '11]. Recently, Daga, Henzinger, Nanongkai and Saranurak [STOC '19] raised the open problem of computing the minimum cut in $\mathsf{poly}(D)$ rounds for any $\lambda = O(1)$. In this paper, we resolve this problem by presenting a surprisingly simple algorithm, that takes a completely different approach than the existing algorithms. Our algorithm has also the benefit that it computes *all* minimum cuts in the graph, and naturally extends to *vertex* cuts as well. At the heart of the algorithm is a graph sampling approach usually used in the context of fault tolerant (FT) design.
- **Deterministic Algorithms:** While the existing distributed minimum cut algorithms are randomized, our algorithm can be made deterministic within the same round complexity. To obtain this, we introduce a novel definition of universal sets along with their efficient computation. This allows us to derandomize the FT graph sampling technique, which might be of independent interest.
- **Computation of all Edge Connectivities:** We also consider the more general task of computing the edge connectivity of all the edges in the graph. In the output format, it is required that the endpoints u, v of every edge (u, v) learn the cardinality of the u-v cut in the graph. We provide the first sublinear algorithm for this problem for the case of *constant* connectivity values. Specifically, by using the recent notion of low-congestion cycle cover, combined with the sampling technique, we compute all edge connectivities in $\mathsf{poly}(D) \cdot 2^{O(\sqrt{\log n \log\log n})}$ rounds.

Sparse Certificates: For an n-vertex graph G and an integer λ, a λ-sparse certificate H is a subgraph $H \subseteq G$ with $O(\lambda n)$ edges which is λ-connected iff G is λ-connected. For D-diameter graphs, constructions of sparse certificates for $\lambda \in \{2, 3\}$ have been provided by Thurimella [J. Alg. '97] and Dori [PODC '18] respectively using $\widetilde{O}(D)$ number of rounds. The problem of devising such certificates with $o(D + \sqrt{n})$ rounds was left open by Dori [PODC '18] for any $\lambda \geq 4$. Using connections to fault tolerant spanners, we considerably improve the round complexity for *any* $\lambda \in [1, n]$ and $\epsilon \in (0, 1)$, by showing a construction of $(1-\epsilon)\lambda$-sparse certificates with $O(\lambda n)$ edges using only $O(1/\epsilon^2 \cdot \log^{2+o(1)} n)$ rounds.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Distributed algorithms

Keywords and phrases Connectivity, Minimum Cut, Spanners

Digital Object Identifier 10.4230/LIPIcs.DISC.2019.30

Funding *Merav Parter*: Support in part by an ISF grant no. 2084/18.

Acknowledgements I am thankful to DISC '19 reviewers for valuable input, and to Michal Dory for preliminary discussions on these problems. I am also grateful to Oded Goldriech for sparking my curiosity on the derandomization of the fault tolerant sampling approach. Finally, special thanks to Moni Naor and Karthik C. S. for useful discussions on universal sets.

1 Introduction

The connectivity of a graph is one of the most fundamental concept in graph theory and network reliability. In this paper, we revisit some classical connectivity problems in the CONGEST model of distributed computing via the lens of *fault tolerant* network design. We mainly focus on two problems: exact computation of small[1] edge (or vertex) cuts; and the computation of sparse connectivity certificates. Both of these problems have been studied thoroughly in the literature, and surprisingly still admit critically missing pieces. By using techniques from fault tolerant network design we considerably improve the state-of-the-art, as well as provide the first *deterministic* distributed algorithms for these problems.

1.1 Small Cuts

In the distributed minimum cut problem, given a graph G with edge connectivity λ, the goal is to identify at least one minimum cut, that is a collection of λ edges whose removal disconnect the graph. In the output format, each vertex should learn at least one possible minimum (edge) cut. We start by providing a brief history for the problem[2].

A Brief History. *(I) Upper Bounds:* The first non-trivial distributed algorithm for the minimum cut problem was given by Ghaffari and Kuhn [10]. They presented a randomized algorithm for computing a $(2+\epsilon)$ approximation of minimum cut using $\widetilde{O}_{\epsilon}(D+\sqrt{n})$ rounds, with high probability. Shortly after, Nanongkai and Su [16] improved the approximation ratio to $(1+\epsilon)$ with roughly the same round complexity. Recently, Daga et. al [5] provided an exact algorithm with sublinear round complexity which improves up on the state of the art in the regime of large cuts (i.e., for $\lambda = n^{\Omega(1)}$).

(II) Lower Bounds: In their seminal paper, Das-Sarma et al. [23] presented a lower bound of $\widetilde{\Omega}(D+\sqrt{n})$ rounds for the computation of an α-approximation of a weighted minimum cut which holds even for graphs with diameter $D = O(\log n)$. This lower bound applies only for weighted graphs with large weighted minimum cut of size $\Omega(\sqrt{n})$. Ghaffari and Kuhn [11] extended this lower bounds in two ways. First, they considered a weaker setting for weighted minimum cuts where the edge weights correspond to capacities, and thus nodes can exchange $O(w \log n)$ bits over edges of weight w in each round. They showed that even in this weaker model, the α-approximation of minimum cut in λ-edge connected graphs with $\lambda = \Theta(\sqrt{n})$ and diameter $O(\log n)$ requires $\widetilde{\Omega}(\sqrt{n/(\alpha \cdot \lambda)})$ rounds[3]. Observe that since in this construction $\lambda = \Theta(\sqrt{n})$, this lower bound can also be stated as $\Omega(\sqrt{\lambda})$ rather than $\Omega(D+\sqrt{n})$. In their second extension, Ghaffari and Kuhn attempted to capture also unweighted simple graphs. Here, they showed a lower bound of $\widetilde{\Omega}(D+\sqrt{n/(\alpha \cdot \lambda)})$ rounds,

[1] By small we mean of constant size.

[2] Although historically, the lower bound by Das-Sarma et al. [23] appeared before the upper bound algorithms, we reverse the order of presentation here.

[3] In the conference version of [10], a lower bound of $\widetilde{\Omega}(D+\sqrt{n})$ was mistakenly claimed for *any* $\lambda \geq 1$ and graphs with diameter $D = O(\log n)$. This was later on fixed in a modified arXiv version [11] and in Ghaffari's thesis [9].

for any $\lambda \geq 1$ but only for graphs with diameter $D = 1/\lambda \cdot \sqrt{n/(\alpha \cdot \lambda)}$. Again[4], with such a larger diameter, one can alternatively state this lower bound as $\Omega(\lambda \cdot D)$, rather than $\widetilde{\Omega}(D + \sqrt{n})$.

Computation of Small Cuts. The conclusion from the above discussion is that we still do not have matching bounds for the distributed minimum cut problem in cases where either (i) the value of the weighted minimum cut is $o(\sqrt{n})$, or (ii) the unweighted diameter is $o(\sqrt{n})$.

As most real-world networks admit small cuts [26], we are in particular intrigued by the complexity of computing the cuts in unweighted graphs with constant connectivity. Pritchard and Thurimella [21] showed an $O(D)$-round randomized algorithm for cut values up to 2. The problem of devising an $\mathsf{poly}(D)$ round algorithm for any constant $\lambda = O(1)$ was recently raised by Daga et al. [5]:

> *"A special case that deserves attention is when the graph connectivity is small. For example, is there an algorithm that can check whether an unweighted network has connectivity at most k in $\mathsf{poly}(k, D, \log n)$? ... Bounds in these forms are currently known only for $k \leq 2$."*

We answer this question in the affirmative by presenting a $\mathsf{poly}(D)$-round algorithm for any constant connectivity $\lambda = O(1)$. This algorithm in fact computes all possible minimum cuts in G, in the sense that for any min-cut set E' there is at least one vertex in the graph that knows E'. Turning to *vertex* cuts, [21] showed an $O(D + \Delta/\log n)$ round algorithm for computing the cut vertices. No exact algorithm is known for the case where the vertex connectivity is at least two. Our algorithm can be easily adapted to compute deterministically the (exact) vertex cuts in $\mathsf{poly}(D \cdot \Delta)$ rounds where Δ is the maximum degree.

1.2 Sparse Connectivity Certificates

For a given unweighted n-vertex D-diameter graph $G = (V, E)$ and integer $\lambda \geq 1$, a *connectivity certificate* is a subgraph $H \subseteq G$ satisfying that it is λ-edge (or vertex) connected iff G is λ-edge (or vertex) connected. The certificate is said to be *sparse* if H has $O(\lambda n)$ edges. Sparse certificates were introduced by Nagamochi and Ibaraki [15]. Thurimella [24] gave the first distributed construction of λ-sparse certificates using $O(\lambda \cdot (D + \sqrt{n}))$ rounds in the $\mathsf{CONGEST}$ model. For $\lambda = 2$, Censor-Hillel and Dory [3] showed[5] the randomized construction of a certificate with $O(n)$ edges using $O(D)$ rounds. In [7], Dory considered the case of $\lambda = 3$, and showed the construction of a certificate with $O(n \log n)$ edges and $O(D \log^3 n)$ rounds. These algorithms are randomized and are based on the cycle space sampling technique of Pritchard and Thurimella [21]. The problem of designing sparse certificates for any $\lambda \geq 4$ using $\widetilde{O}(D)$ rounds was left open therein [7].

In this paper, we provide an easy solution for this problem which takes only $\widetilde{O}(\lambda)$ rounds for any λ. This is based on the observation that fault tolerant spanners are in fact sparse connectivity certificates. As a result we get that the problem of designing sparse certificates is *local* rather than global (i.e., does not depend on the graph diameter). In the Our Results section we also improve the round complexity into $\widetilde{O}(1)$ (i.e., independent on λ) by loosing a small factor in the approximation.

[4] Also here the conference version [10] mistakenly claimed that the lower bound works even for graphs with diameter $D = O(\log n)$, and this was fixed in [11, 9].

[5] [3] studied the problem of the minimum k-edge-connected spanning subgraph (k-EECS), which for unweighted graphs implies the computation of connectivity certificates with a small number of edges.

1.3 Our Results

Distributed Computation of Small Minimum Cuts. We consider an unweighted D-diameter graph $G = (V, E)$ with edge connectivity $\lambda = O(1)$. We show a $\mathsf{poly}(D)$-round randomized algorithm to compute the minimum cut whose high level description can be stated in just few lines: Fix a vertex s and apply $\mathsf{poly}(D)$ iterations, where in iteration i we do as follows. (i) Sample a subgraph $G_i \subseteq G$ by adding each edge e into G_i independently with some fixed probability p. (ii) Compute a truncated BFS tree rooted at s up to depth $O(\lambda D)$ in G_i, and (iii) let each vertex t collect its s-t path in this tree (if such exists). Finally, after applying this procedure for $\mathsf{poly}(D)$ iterations, each vertex t computes locally the s-t cut on the subgraph that it has collected. The argument shows that every vertex t that is separated from s by some minimum cut E', can compute this set of edges w.h.p.

▶ **Theorem 1.** *Let G be an $\lambda = O(1)$ connected D-diam graph and max degree Δ. There exists a randomized minimum cut algorithm that runs in $\mathsf{poly}(D)$ rounds. In addition, with a small modification it computes the minimum vertex cut in $\mathsf{poly}(D \cdot \Delta)$ rounds.*

The algorithm is in fact stronger. Every vertex t also learns a collection $(\lambda - 1)$ edge disjoint paths from s (i.e., an integral flow from s). In addition, we do not compute only one minimum cut but rather for each minimum cut in G, there is at least one vertex that learns it.

Deterministic Computation of Small Cuts. So-far, the distributed minimum cut computation was inherently randomized. The randomized component of the algorithm of Thm. 1 is in the initial graph sampling in each iteration. To derandomize it, we introduce a new variant of *universal-sets*. We use this notion to explicitly compute, in polynomial time, a collection of $\mathsf{poly}(D)$ subgraphs $G_1, \ldots, G_k$ that have the same key properties as those obtained by the sampling approach. The polynomial computation is done *locally* at each vertex and thus does not effect the round complexity.

▶ **Theorem 2.** *One can compute small cuts deterministically in $\mathsf{poly}(D)$ rounds.*

This derandomization technique can be used to derandomize all other algorithms that are based on the fault tolerant (FT) sampling technique (e.g., [6],[27]), and it is therefore of interest also for centralized algorithms. Independently of our work, Alon, Chechik and Cohen [2] also studied the derandomization of algorithms that are based on the FT-sampling approach, their solution is different than ours.

For a summary on the computation of small cuts, see Table 1.

Table 1 State of the art results for exact distributed computation of small cuts.

	Min-Cut Value λ	**#Rounds**	**Type**
Pritchard & Thurimella [21]	2 Edges	$O(D)$	Rand.
Pritchard & Thurimella [21]	1 Vertex	$O(D + \Delta)$	Rand.
This Work	$O(1)$ Edges	$\mathsf{poly}(D)$	Det.
This Work	$O(1)$ Vertices	$\mathsf{poly}(D \cdot \Delta)$	Det.

Computation of Edge-Connectivities. We then turn to consider the more general task of computing the edge connectivity of all graph edges, up to some constant bound $\lambda = O(1)$. For an edge $e = (u, v)$, the *edge connectivity* of e is the size of the u-v minimum (edge) cut in G. In the output format for each edge $e = (u, v)$, its endpoints are required to learn the edge connectivity of e. Exact computation of all edge connectivites has been previously known only $\lambda \leq 2$ due to Pritchard and Thurimella [21]. They gave randomized algorithms for the case of $\lambda = 1, 2$ with round complexities of $O(D)$ and $O(D + \sqrt{n}\log^* n)$, respectively.

In this paper, we again take a completely different approach and show a *deterministic* algorithm with $\mathsf{poly}(D) \cdot 2^{\sqrt{\log n \log\log n}}$ rounds for computing all edge connectivities up to constant value of $\lambda = O(1)$. Our algorithm is based on two tools: (1) *low-congestion cycle cover* [17] and their distributed computation [18] ; and (2) the derandomization of the FT-sampling approach.

▶ **Theorem 3.** *For every D-diameter n-vertex graph $G = (V, E)$, w.h.p., the edge connectivity of all graphs edges up to $\lambda = O(1)$ can be computed in $\mathsf{poly}(D) \cdot 2^{O(\sqrt{\log n})}$ rounds. This algorithm can also be derandomized using $\mathsf{poly}(D) \cdot 2^{O(\sqrt{\log n \log\log n})}$ rounds.*

Sparse Connectivity Certificates. In the second part of the paper we consider the related problem of computing connectivity certificates. We first show that by a direct application of fault tolerant spanners, one can compute a λ-edge connectivity certificate with $O(\lambda n)$ edges using $\widetilde{O}(\lambda)$ rounds. This considerably improves and extends up on the previous constructions with $O(D)$ rounds that were limited *only* for $\lambda \in \{2, 3\}$.

▶ **Lemma 4.** *For every $\lambda \in \mathbb{N}_{\geq 1}$, there is a randomized algorithm that computes a λ connectivity certificate with $O(\lambda n)$ edges in $O(\lambda \log^{1+o(1)} n)$ rounds, with high probability.*

By plugging in the recent *deterministic* spanner construction of [12], one can compute λ-edge connectivity certificate deterministically with $\widetilde{O}(\lambda \cdot n)$ edges and $\lambda \cdot 2^{O(\sqrt{\log n})}$ rounds[6]. This answers the open problem raised by Dory [7] concerning the existence of efficient deterministic constructions of connectivity certificates.

To avoid the dependency in λ in the round complexity, we use the well known Karger's edge-sampling technique, and show:

▶ **Lemma 5.** *For every λ-connected graph and $\epsilon \in (0, 1)$, there is a randomized distributed algorithm that computes a $(1-\epsilon)\lambda$ connectivity certificate with $O(\lambda n)$ edges in $O(1/\epsilon^2 \cdot \log^{2+o(1)} n)$ rounds, with high probability.*

Table 2 summarizes the state of the art. Note that if one uses fault tolerant spanners resilient for *vertex* faults, we get $\widetilde{O}(\lambda)$-round algorithm for computing λ-vertex-certificates[7] with $O(\lambda^2 n)$ edges. For clarity of presentation, we mainly focus on the edge-connectivity certificates, but our results naturally extend to the vertex case as well.

Table 2 State of the art result for distributed computation of sparse connectivity certificates.

	Edge Connectivity λ	**Certificate Size**	**#Rounds**	**Type**
Thurimella [24]	Any	$\lambda \cdot n$	$\widetilde{O}(\lambda \cdot (D + \sqrt{n}))$	Det.
Pritchard & Thurimella [21]	2	$2n$	$O(D)$	Rand.
Dory [7]	3	$O(n \log n)$	$O(D \cdot \log^3 n)$	Rand.
This Work	Any	$O(\lambda n)$	$O(\lambda \cdot \log^{1+o(1)} n)$	Rand.
This Work	Approx. $(1-\epsilon)$	$O(\lambda \cdot n)$	$O(\log^{2+o(1)} n)$	Rand.
This Work	Any	$\widetilde{O}(\lambda \cdot n)$	$\lambda \cdot 2^{O(\sqrt{\log n \log\log n})}$	Det.

[6] Combining the recent result of [22] with [12] seems to improve the deterministic spanner construction to $\mathsf{poly}\log n$ rounds, and thus provide $\widetilde{O}(\lambda)$-round algorithm for λ-sparse certificates.

[7] I.e., a subgraph $H \subseteq G$ satisfying that H is λ-vertex connected iff G is λ-vertex connected.

Graph Notation. For a subgraph $G' \subseteq G$ and $u, v \in V(G')$, let $\pi(u, v, G')$ be the unique[8] shortest-path in G'. When G' is clear from the context, we may omit it and simply write $\pi(u, v)$. For $u, v \in G$, let $\mathrm{dist}(u, v, G)$ be the length of the shortest u-v path in G. For a vertex pair s, t and subgraph $G' \subseteq G$, let $\lambda(s, t, G')$ be the s-t cut in G'.

The Communication Model. We use a standard message passing model, the CONGEST model [19], where the execution proceeds in synchronous rounds and in each round, each node can send a message of size $O(\log n)$ to each of its neighbors.

2 Exact Computation of Small Cuts

Throughout, we consider unweighted multigraphs with diameter D, and (edge or vertex) connectivity at most $\lambda = O(1)$. Before presenting the algorithm we start by considering the following simpler task.

Warm Up: Cut Verification. In the cut verification problem, one is given a subset of edges E' where $|E'| \leq \lambda$, it is then required to test if $G \setminus E'$ is connected. As we will see there is a simple algorithm for this problem which is based on the following key lemma.

▶ **Lemma 6.** *Consider a D-diameter unweighted graph $G = (V, E)$ with maximum degree Δ. (1) If $u, v \in V$ are λ-edge connected[9] (i.e., the u-v cut is at least λ) then* $\mathrm{dist}(u, v, G \setminus F) \leq c \cdot \lambda \cdot D$ *for every edge sequence $F \subseteq E$, $|F| \leq \lambda - 1$ for some constant c.*
(2) If $u, v \in V$ are λ-vertex connected then $\mathrm{dist}(u, v, G \setminus F) \leq c \cdot \lambda \cdot \Delta \cdot D$ *for every vertex sequence $F \subseteq V$, $|F| \leq \lambda - 1$.*

Proof. Let T be an arbitrary BFS tree in G of diameter $O(D)$. We begin with (1). Fix a set of faults $F \subseteq E$, $|F| \leq \lambda - 1$, and let $P_{u,v,F}$ be the u-v shortest path in $G \setminus F$. Since u and v are λ-edge connected such path $P_{u,v,F}$ exists. We next bound the length of $P_{u,v,F}$. Consider the forest $T \setminus F$ which has at most λ connected components $C_1, \ldots, C_\ell$. We mark each vertex on $P_{u,v,F}$ with its component ID in the forest $T \setminus F$. Note that all vertices in the same component are connected by a path of length $O(D)$ in $G \setminus F$. We can then traverse the path $P_{u,v,F}$ from u and jump to the last vertex u_1 on the path (close-most to v) that belongs to the component of u. The length of this sub-path is $O(D)$ and using one connecting edge on $P_{u,v,F}$, we move to a vertex belonging to a new component in $T \setminus F$. Overall the path $P_{u,v,F}$ can be covered by λ path segments $P_1, \ldots, P_\ell$ such that the endpoints of each segments are in the *same* component in $T \setminus F$, each neighboring segments P_i and P_{i+1} are connected by an edge from $P_{u,v,F}$. Since each $|P_i| = O(D)$, we get that $|P_{u,v,F}| = O(\lambda \cdot D)$. Claim (2) follows the exact same argument with the only distinction is that when a vertex fails, the BFS tree might break up into $\Delta + 1$ components. Thus, for a subset $F \subseteq V$ of vertices with $|F| \leq \lambda - 1$, the tree T breaks into $O(\lambda \cdot \Delta)$ components. ◀

This lemma immediately implies an $O(\lambda D)$-round solution for the cut verification task: build a BFS tree T from an arbitrary source up to depth $O(\lambda \cdot D)$. Then T is a spanning tree iff E' does not disconnect the graph.

▶ **Corollary 7** (Cut Verification). *Given a set of λ edges E'. One can test if E' is a cut in G using $O(\lambda \cdot D)$ rounds.*

[8] Ties are broken is a consistent manner.
[9] Note that we do not require the graph G to be λ connected.

The following definition is useful for the description and analysis of our algorithm.

▶ **Definition 8** ((s, t) connectivity certificate)**.** *Given a graph G with minimum cut λ and a pair of vertices s and t, the* (s, t) connectivity certificate *is a subgraph $G_{s,t} \subseteq G$ satisfying that s and t are λ-connected in $G_{s,t}$ iff they are λ-connected in G.*

Whereas a-priori the size of the s-t connectivity certificate, measured by the number of edges, might be $\Omega(n)$, as will show later on, it is in fact bounded by $(\lambda D)^{O(\lambda)}$, hence $\mathsf{poly}(D)$ for $\lambda = O(1)$. With this definition in mind, we are now ready to present the minimum cut algorithm.

A $\mathsf{poly}(D)$-Round Randomized Algorithm. The algorithm has two phases. In the first phase, every vertex t computes its (s, t) certificate subgraph $G_{s,t}$ w.r.t. a given fixed source s. In the second phase, each vertex y locally computes its s-t cut in the subgraph $G_{s,t}$, and one of the output λ-size cut is broadcast to the entire network. Throughout, we assume w.l.o.g. that the value of the minimum cut λ is known, since $\lambda = O(1)$ this assumption can be easily removed.

The first phase has $\ell = O((\lambda D)^{2\lambda})$ iterations, or *experiments*. In each iteration i, the algorithm samples a subgraph G_i by including each edge $e \in G$ into G_i independently with probability $p = 1 - 1/(c(\lambda D))$ for some constant c (taken from Lemma 6). For a source vertex s (which is fixed in all iterations), a (truncated) BFS tree B_i rooted in s is computed in G_i up to depth $c \cdot \lambda \cdot D$. Next, every vertex in B_i learns its tree path from s by pipelining these edges downward the tree. This completes the description of an iteration. Let $G_{s,t} = \bigcup_{i=1}^{\ell} \pi(s, t, B_i)$.

In the second phase, every vertex t locally computes its s-t cut in the subgraph $G_{s,t}$. The edges of the minimum cut are those obtained by one of the vertices t whose s-t connectivity in $G_{s,t}$ is at most λ.

Correctness. For the correctness of the algorithm it will be sufficient to show that w.h.p. $G_{s,t}$ is an s-t connectivity certificate for every $t \in V$.

▷ Claim 9. For every t, w.h.p., $G_{s,t}$ is an s-t connectivity certificate.

Proof. Since $G_{s,t} \subseteq G$, it is sufficient to show that s and t are connected in $G_{s,t} \setminus F$ for any subset $F \subset E$, $|F| \leq \lambda$ satisfying that s and t are connected in $G \setminus F$. Fix such a triplet $\langle s, t, F\rangle$ where s and t are connected in $G \setminus F$. An iteration i is *successful* for $\langle s, t, F\rangle$ if

$$\pi(s, t, G \setminus F) \subseteq G_i \quad \text{and} \quad F \cap G_i = \emptyset.$$

Note that if iteration i is successful for $\langle s, t, F\rangle$, then the truncated BFS tree B_i contains t as $\text{dist}(s, t, G_i) = |\pi(s, t, G \setminus F)| \leq c \cdot \lambda \cdot D$, where the last inequality is due to Lemma 6(1). In addition, since $F \cap G_i = \emptyset$, it also holds that $\pi(s, t, B_i) \subseteq G_{s,t} \setminus F$.

It remains to show that with probability at least $1 - 1/n^{\Omega(\lambda)}$, every triplet $\langle s, t, F\rangle$ where s and t are connected in $G \setminus F$, has at least one successful iteration. The claim will then follow by applying the union bound over all $n^{2\lambda}$ triplets. Recall that each edge is sampled into G_i independently with probability p. Thus the probability that iteration i is successful for $\langle s, t, F\rangle$ is at least:

$$q = p^{(c \cdot \lambda D)} \cdot (1 - p)^{\lambda} = 1/(\lambda \cdot D)^{\lambda}.$$

Since there are ℓ independent experiments, the probability that all of them fail is $(1 - q)^{\ell} \leq 1/n^{\Omega(\lambda)}$, the claim follows. ◁

Finally, let t be a vertex such that s and t are not $(\lambda+1)$-connected in G. Thus, by the lemma above, $\lambda(s,t,G_{s,t}) \leq \lambda$ and the minimum cut computation applied locally by vertex t in $G_{s,t}$ outputs a subset F of at most λ edges. We claim that w.h.p. s and t are also not connected in $G \setminus F$. Assume otherwise, then by Lemma 6(1), $\text{dist}(s,t,G \setminus F) \leq c \cdot \lambda \cdot D$, thus by the argument above, w.h.p., there is an iteration i in which an s-t path that does not go through F is taken into $G_{s,t}$, leading to a contradiction that s and t are disconnected in $G_{s,t} \setminus F$.

▶ **Corollary 10.** *For every D-diameter unweighted graph $G=(V,E)$, $\lambda \geq 1$ and vertex pair $s,t \in V$, there exists an (s,t) certificate $G_{s,t} \subseteq G$ of size $(\lambda D)^{O(\lambda)}$.*

Round Complexity. Each of the ℓ iterations takes $O(\lambda \cdot D)$ rounds for computing the truncated BFS tree. Learning the edges along the tree path from the root also takes $O(\lambda \cdot D)$ rounds via pipeline, thus overall the round complexity is $(\lambda \cdot D)^{(c+2)\lambda} = \mathsf{poly}(D)$ for $\lambda = O(1)$.

Extension to Vertex Cuts The algorithm for computing vertex cuts is almost identical and requires minor adaptations. First, instead of having a single source vertex s, we will pick $\lambda+1$ arbitrary sources $s_1, \ldots, s_{\lambda+1}$ and will run an algorithm, which is very similar to the one described above, with respect to each source s_i. Note that since the vertex cut has size λ, then there is at least one vertex cut $V' \subset V$ of size λ that does not contain at least one of the sources s_i. In such a case, our algorithm will find the cut V' when running the below mentioned algorithm w.r.t the source s_i.

The algorithm for each source s_i works in iterations, where each iteration j samples into a subgraph G_j a collection of vertices rather than edges. That is, the subgraph G_j is defined by taking the induced graph on a sample of vertices, where each vertex gets sampled independently with probability $p' = (1 - 1/(c\lambda \cdot \Delta \cdot D))$ (the constant c is taken from Lemma 6(2)). Then a BFS tree B_j rooted at s_i is computed in G_j up to depth $c\lambda \cdot \Delta \cdot D$. Every vertex $v \in B_j$ collects its path from the root. Let $G_{s_i,t}$ be the union of all paths collected for each vertex t. In the second phase, the vertex t computes locally the s_i-t vertex-cut in $G_{s_i,t}$. The analysis is then identical to that of the edge case, where in particular, we get that w.h.p. $G_{s_i,t}$ is the vertex-connectivity certificate for every t.

2.1 Deterministic Min-Cut Algorithms via Universal Sets

Our goal in this section is to derandomize the FT-sampling technique by locally computing explicitly (at each node) a *small* family of graphs $\mathcal{G} = \{G_i \subseteq G\}$ such that in iteration i, the vertices will apply the computation on the graph G_i in the same manner as in the randomized algorithm. Here, however, the graph G_i is not sampled but rather computed locally by all the vertices. The family of subgraphs $\mathcal{G}$ is required to satisfy the following crucial property for $a = c \cdot D \cdot \lambda$ and $b = \lambda$:

For every two disjoint subsets of edges A, B with $|A| \leq a$ and $|B| \leq b$, there exists a subgraph $G_i \in \mathcal{G}$ satisfying that:

$$A \subseteq G_i \ \text{ and } \ B \cap G_i = \emptyset\ . \tag{1}$$

In our algorithms, the subset B corresponds to a set of edge faults, and A corresponds to an s-t shortest path in $G \setminus B$. Thus, $|B| \leq \lambda$ and by Lemma 6, $|A| = O(\lambda \cdot D)$. We begin with the following observation that follows by the probabilistic method.

▶ **Lemma 11.** *There exists a family of graphs $\mathcal{G} = \{G_i \subseteq G\}$ of size $O(a^{b+1} \cdot \log n)$ that satisfies Eq. (1) for every disjoint $A, B \subseteq E$ with $|A| \leq a$ and $|B| \leq b$.*

Proof. We will show that a random family $\mathcal{G}_R$ with $\ell = O(a^{b+1} \cdot \log n)$ subgraphs satisfies Eq. (1) with non-zero probability. Each subgraph G_i in $\mathcal{G}_R$ is computed by sampling each edge in G into G_i with probability of $p = (1 - 1/a)$.

The probability that G_i satisfies Eq. (1) for a fixed set A and B of size at most a and b (respectively) is $q = p^a \cdot (1-p)^b = 1/a^b$. The probability that none of the subgraph G_i satisfy Eq. (1) for A, B is at most $(1-q)^{a^{b+1} \cdot \log n} \leq 1/n^{3a}$. Thus, by doing a union bound over all n^{2a} possible subsets of A, B, we get that $\mathcal{G}_R$ satisfies Eq. (1) for all subsets with positive probability. The lemma follows. ◀

Lemma 11 already implies a deterministic minimum cut algorithm with $\mathsf{poly}(D)$ rounds, in case where nodes are allowed to perform *unbounded* local computation. Specifically, let every node compute locally, in a brute force manner, the family of graphs $\mathcal{G} = \{G_i \subseteq G\}$ of size $a^{b+1} \cdot \log n$. In each iteration i of the minimum-cut computation, nodes will use the graph $G_i \in \mathcal{G}$ to compute the truncated BFS tree, and collect their tree paths in these trees. Although the CONGEST model does allow for an unbounded local computation, it is still quite undesirable. To avoid this, we next describe an explicit *polynomial* construction of the graph family $\mathcal{G}$. This explicit construction is based on stating our requirements in the language of universal sets.

Universal Sets. A family of sets $\mathcal{S} = \{S \subseteq [n]\}$ is (n, k)-*universal* if every subset $S' \subseteq [n]$ of $|S'| = k$ elements is *shattered* by $\mathcal{S}$. That is, for each of the 2^k subsets $S'' \subseteq S'$ there exists a set $S \in \mathcal{S}$ such that $S' \cap S = S''$. Using linear codes, one can compute (n, k)-universal sets with $n^{O(k)}$ subsets. Alon [1] showed an explicit construction of size $2^{O(k^4)} \log n$ using the Justesen-type codes constructed by Friedman [8]. In our context, the parameter n corresponds to the number of graph edges, and each subset is a subgraph. The parameter k corresponds to the bound on the length of the path which is $O(\lambda \cdot D)$. Using the existing constructions lead to a family with $2^{\lambda \cdot D}$ subgraphs which is unfortunately super-linear, already for graphs of logarithmic diameter.

A New Variant of Universal Sets. We define a more relaxed variant of universal sets, for which a considerably improved size bounds can be obtained. In particular, for our purposes it is not really needed to fully shatter subsets of size k. Instead, for every set S' of k elements we would like that for every small subset $S'' \subseteq S$, $|S''| \leq b$ (which plays the role of the faulty edges), there will be a set S in the family satisfying that $S' \cap S = S' \setminus S''$. We call this variant FT-universal sets, formally defined as follows.

▶ **Definition 12** (FT-Universal Sets). *For integers n, a, b where $a \leq b \leq n$, a family of sets $\mathcal{S} = \{S \subset [1, n]\}$ is (n, a, b)-*universal *if for every two disjoint subsets $A \subset [1, n]$ and $B \subset [1, n]$ where $|A| \leq a$ and $|B| \leq b$, there exists a set $S \in \mathcal{S}$ such that (1) $A \subseteq S$ and (2) $B \cap S = \emptyset$.*

Our goal is compute a family of (n, a, b)-universal sets of cardinality $O(a^{b+1} \log n)$ in time $\mathsf{poly}(n, a^b)$. Towards that goal we will use the notion of perfect hash functions.

▶ **Definition 13** (Perfect Hash Functions). *For integers n and $k < n$ a family of hash functions $\mathcal{H} = \{h : [n] \rightarrow [\ell]\}$ is* prefect *if for every subset $S \subset [1, n]$ for $|S| \leq k$, there exists a function $h \in \mathcal{H}$ such that $h(i) \neq h(j), \quad \forall i, j \in S, i \neq j$.*

▶ **Definition 14** (Almost Pairwise Independence). *A family of functions $\mathcal{H}$ mapping domain $[n]$ to range $[m]$ is ϵ-almost pairwise independent if for every $x_1 \neq x_2 \in [n]$, $y_1, y_2 \in [m]$, we have:* $\Pr[h(x_1) = y_1$ *and* $h(x_2) = y_2] \leq (1+\epsilon)/m^2$.

▶ **Fact 15** ([25]). *For every $\alpha, \beta \in \mathbb{N}$ and $\epsilon \in (0,1)$, one can compute in* $\mathsf{poly}(\alpha \cdot 2^{\beta} \cdot 1/\epsilon)$ *an explicit family of ϵ-almost pairwise independent hash functions $\mathcal{H}_{\alpha,\beta} = \{h : \{0,1\}^{\alpha} \to \{0,1\}^{\beta}\}$ that contains $O(\alpha \cdot 2^{\beta})$ functions.*

We next show how to compute a family of (n,k)-perfect hash functions in polynomial time.

▷ Claim 16. One can compute an family of (n,k)-perfect hash functions $\mathcal{H} = \{h : [n] \to [2k^2]\}$ of cardinality $O(k^4 \log n)$ in time $\mathsf{poly}(n,k)$.

Proof. We use Fact 15 with $\alpha = \log n$, $\beta = 4\log k$, and $\epsilon = 0.1$, to get an ϵ-almost pairwise independent hash function family $\mathcal{H}_{\alpha,\beta} = \{h : \{0,1\}^{\alpha} \to \{0,1\}^{\beta}\}$. We now show that this family is perfect for subsets $S \in [1,n]$ of cardinality at most k. Fix a subset $S \in [1,n]$, $|S| \leq k$. By definition, for every $x_1, x_2 \in S$ and $y_1, y_2 \in [2k^2]$,

$$\Pr_{h \in \mathcal{H}}[h(x_1) = y_1 \text{ and } h(x_2) = y_2] \leq (1+\epsilon)/(4k^4).$$

Thus, the probability that a uniformly chosen random function $h \in \mathcal{H}_{a,b}$ collides on S is

$$\begin{aligned} \sum_{x_1 \neq x_2 \in S} \Pr_{h \in \mathcal{H}}[h(x_1) = h(x_2)] &\leq k^2 \cdot \max_{x_1 \neq x_2 \in S} \Pr_{h \in \mathcal{H}}[h(x_1) = h(x_2)] \\ &= k^2 \cdot \max_{x_1 \neq x_2 \in S} \sum_{y \in [2k^2]} \Pr[h(x_1) = h(x_2) = y] \leq 0.3 \ , \end{aligned} \tag{2}$$

by using the fact that $\Pr_{h \in \mathcal{H}}[h(x_1) = h(x_2) = y] \leq (1+\epsilon)/(4k^4)$ (see Def. 14). We get that there exists $h' \in \mathcal{H}_{\alpha,\beta}$ that has no collisions on S. As this holds for every S, the claim follows. ◁

Equipped with the polynomial construction of families of (n,k)-perfect hash functions, we next show how to compute our universal sets in polynomial time.

▶ **Lemma 17** (Small Universal Sets). *For every set of integers $b < a < n$, one can compute in* $\mathsf{poly}(n, a^b)$*, a family of universal sets $\mathcal{S}_{n,a,b}$ of cardinality $O(a^{b+1} \cdot \log n)$.*

Proof. Set $k = a + b$. We will use Claim 16 to compute an (n,k)-perfect family of hash functions $\mathcal{H} = \{h : [n] \to [2k^2]\}$. For every $h \in \mathcal{H}$ and for every subset $i_1, \ldots, i_b \in [1, 2k^2]$, define:

$$S_{h,i_1,i_2,\ldots,i_b} = \{\ell \in [n] \mid h(\ell) \notin \{i_1, i_2, \ldots, i_b\}\} \ .$$

Overall, $\mathcal{S}_{n,a,b} = \{S_{h,i_1,i_2,\ldots,i_b} \mid h \in \mathcal{H}, i_1, i_2, \ldots, i_b \in [1, 2k^2]\}$.

The size of $\mathcal{S}_{n,a,b}$ is bounded by $|\mathcal{H}| \cdot k^{2b} = O(k^{3b} \cdot \log n)$ as desired. We now show that $\mathcal{S}_{n,a,b}$ is indeed a family of universal sets for n, a, b. Since $\mathcal{H}$ is an (n,k) perfect family of hash functions, for every two disjoint subsets $A, B \subset [n]$, $|A| \leq a$ and $|B| \leq b$, there exists a function h that does not collide on $C = A \cup B$ (since $|C| \leq k$). That is, there exists a function $h \in \mathcal{H}$ such that $h(i) \neq h(j)$ for every $i, j \in C$, $i \neq j$. Thus, letting $B = \{s_1, \ldots, s_b\}$ and $i_1 = h(s_1), \ldots, i_b = h(s_b)$, we have that $h(s'_j) \notin \{i_1, \ldots, i_b\}$ for every $s'_j \in A$. Therefore, the subset $S_{h,i_1,i_2,\ldots,i_b}$ satisfies that $A \subseteq S_{h,i_1,i_2,\ldots,i_b}$ and $B \cap S_{h,i_1,i_2,\ldots,i_b} = \emptyset$. ◀

Deterministic Min-Cut Algorithm. Finally, we describe how to use FT-universal sets to get a $\mathsf{poly}(D)$-round distributed algorithm for exact computation of small cuts. The only randomized part of the algorithm above is in defining the $\ell = \mathsf{poly}(D)$ subgraphs G_i. Instead of sampling these subgraphs, each vertex computes them explicitly and locally. First, we rename all the edges to be in $[1, m]$. This can be easily done in $O(D)$ rounds. Now, each vertex locally computes a family of universal sets for parameters $m, k = O(\lambda \cdot D), q = \lambda$. By Lemma 17, the family $\mathcal{S}$ contains $(\lambda \cdot D)^{\lambda} = \mathsf{poly}(D)$ subsets in $[1, m]$. Each of the sets $S_i \in \mathcal{S}$ will be used as a subgraph G_i in the i^{th} iteration. That is, we iterate over all subsets (subgraphs) in $\mathcal{S}$. In iteration i, all vertices know the set S_i and thus can locally decide which of their incident edges is in G_i. The correctness now follows the exact same line as that of the randomized algorithm.

3 Computation of All Edge Connectivities

Finally, we consider the more general task of computing the edge connectivity of all graph edges up to some constant value λ. For an edge $e = (u, v)$, let $\lambda(e)$ be the u-v edge connectivity in G, that is, the number of edge-disjoint u-v paths in G. By using the recent notion of low-congestion cycle cover [17], we show:

▶ **Lemma 18** (Distributed All Edge Connectivities). *For every D-diameter graph G, there is a randomized distributed algorithm that w.h.p. computes all edge connectivities up to some constant value λ within $2^{O(\sqrt{\log n})} \cdot \mathsf{poly}(D)$ rounds. That is, in the output solution, the endpoints of every edge $e = (u, v)$ know the connectivity $\lambda(e)$ of this edge, as well as a certificate for that connectivity.*

Low Congestion Cycle Covers. A (d, c) cycle cover $\mathcal{C}$ is a collection of cycles of length at most d, such that each edge appears on at least one cycle and at most c cycles. We will use the recent deterministic distributed construction of cycle covers of [18].

▶ **Lemma 19** ([18], Distributed Cycle Cover). *For every bridgeless n-vertex graph $G = (V, E)$ with diameter D, one can compute a (d, c) cycle cover $\mathcal{C}$ with $\mathsf{d} = 2^{O(\sqrt{\log n})} \cdot D$ and $\mathsf{c} = 2^{O(\sqrt{\log n})}$, within $\widetilde{O}(\mathsf{d} \cdot \mathsf{c})$ rounds.*

Combining the lemma above with the centralized construction of nearly-optimal cycle covers of [17], we get:

▶ **Corollary 20** (Distributed Opt. Cycle Cover). *For an n-vertex graph $G = (V, E)$ (not necessarily connected), there is a randomized algorithm* ApproxCycleCover *that given the graph G and parameter D' computes w.h.p. a cycle collection $\mathcal{C}$ such that: (a) every edge e that lies on a cycle C_e in G of length at most D' is covered by a cycle $C' \in \mathcal{C}$ of length $2^{O(\sqrt{\log n})} \cdot |C_e|$, and (b) each edge appears on $2^{O(\sqrt{\log n})}$ cycles. In the output format of the algorithm, every edge e learns all the cycles in $\mathcal{C}$ that go through this edge. The round complexity of Alg.* ApproxCycleCover *is $2^{O(\sqrt{\log n})} \cdot D'$.*

The high level idea of Alg. ApproxCycleCover is based on the notion of neighborhood covers. Roughly speaking, the k neighborhood cover for a graph G is a collection of subgraphs $G_1, \ldots, G_\ell$ such that the following three properties hold: (1) for each vertex v, there is a subgraph G_i that contains its entire k-hop neighborhood, (2) the diameter of each graph G_i is $O(k \log n)$, and (3) each vertex appears on $O(\log n)$ subgraphs. Alg. ApproxCycleCover is obtained by applying the cycle cover algorithm of Theorem 19 on each subgraph G_i in the

k neighborhood cover of G, for every value $k = 2^j$, $j \in \{1, \lceil \log(D') \rceil\}$. This increases the total congestion of the cycles by at most a logarithmic factor. To see why this works, consider an edge e that lies on a cycle C_e of length $|C_e| \leq D'$ in G. Letting $|C_e| \in [2^{j-1}, 2^j]$, we have that C_e is fully contained in one of the subgraphs of a k neighborhood cover for $k = 2^j$. Hence, due to Theorem 19 the edge e is covered by a cycle of length $2^{O(\sqrt{\log n})}|C_e|$ as desired.

From Cycle Covers to Edge Connectivities. The algorithm for computing all the edge connectivities is based on combining the FT-sampling approach with Alg. ApproxCycleCover. In the high level, using the sampling technique, the algorithm attempts to compute not a single cycle for covering an edge $e = (u, v)$, but rather a collection of λ edge-disjoint cycles that covers this edge (i.e., the edge e is the only common edge in these cycles). If it fails in finding these edge disjoint cycles, it deduces that the u-v connectivity is less than λ. In the latter case, it also finds all u-v cuts in G.

Let $D' = 2c \cdot \lambda \cdot D + 1$. The algorithm consists of $\ell = O(\lambda \cdot D^{\lambda} \log n)$ iterations that we treat as *experiments*. In each experiment i, we sample each edge $e \in E(G)$ into G_i with probability $p = (1 - 1/(D')^{\lambda})$, and compute a cycle cover $\mathcal{C}_i$ by applying Alg. ApproxCycleCover on G_i with parameter D'. For every edge $e = (u, v)$, let $G_{u,v} = \bigcup_i \{C \mid e \in C, C \in \mathcal{C}_i\}$ be the union of all cycles that go through e. The nodes u, v compute the edge connectivity of e by *locally* computing the u-v cut in the subgraph $G_{u,v}$. For a pseudo-code of the algorithm see Algorithm 1. By a similar argument to that of Cl. 9, we show:

▷ Claim 21. The subgraph $G_{u,v}$ is a u-v connectivity certificate up to connectivity of λ.

Proof. Let $e = (u, v)$ such that u and v are λ-edge connected. In other words, u and v are $(\lambda - 1)$-connected in $G \setminus \{e\}$. Note that since the diameter of $G \setminus \{e\}$ is at most $2D$, by Lemma 6, for every $F \subseteq E(G) \setminus \{e\}$, $|F| \leq \lambda - 2$, we have that $\text{dist}(u, v, G \setminus (F \cup \{e\})) \leq 2c \cdot \lambda \cdot D$. Therefore, for any $F \subseteq E(G) \setminus \{e\}$, $|F| \leq \lambda - 2$ the subgraph $G \setminus F$ contains a cycle that covers e of length at most $D' = 2c \cdot \lambda \cdot D + 1$.

To show that $G_{u,v}$ is a λ-certificate for such a neighboring pair u, v (that is λ edge connected in G), it is sufficient to show that for every failing of at most $\lambda - 2$ edges F where $e \notin F$, $G_{u,v}$ contains a u-v path in $G_{u,v} \setminus (F \cup \{e\})$. Or in the other words, that $G_{u,v} \setminus F$ contains a cycle that covers e.

Fix a failing set F, where $e \notin F$ and $|F| \leq \lambda - 2$, and u and v are connected in $G \setminus (F \cup \{(u, v)\})$. We say that iteration i is successful for such a triplet $\langle u, v, F \rangle$ if $F \cap G_i = \emptyset$, $(u, v) \in G_i$ and $\pi(u, v, G \setminus (F \cup \{(u, v)\})) \subseteq G_i$. Note that in such a case, since G_i contains a cycle of length at most D' that covers e, Algorithm ApproxCycleCover computes a cycle $C \subseteq G_i$ that covers the edge $e = (u, v)$, and thus a u-v path in $G_i \setminus (F \cup \{e\})$ as desired. It remains to show that w.h.p. every triplet $\langle u, v, F \rangle$ has at least one successful iteration. Since each edge is sampled w.p. p into G_i, the iteration is successful with probability $\Omega(1/D^{\lambda})$. By simple application of the Chernoff bound, we get that the probability that a given triplet $\langle u, v, F \rangle$ does not have a successful iteration is at most $1/n^{c' \cdot \lambda}$. Thus by applying the union bound over all $n^{\lambda+2}$ triplets, the claim follows. ◁

The proof of Lemma 18 follows by Cor. 20 and Cl. 21. Note that this algorithm can be made deterministic while keeping the same round complexity, by using the derandomization of the FT-sampling approach from Sec. 2.1 along with the deterministic neighborhood cover construction of [12].

▶ **Lemma 22.** *All edge connectivities, up to a constant λ, can be computed deterministically in* $2^{\sqrt{\log n \cdot \log\log n}} \cdot \mathsf{poly}(D)$.

Proof. The algorithm requires two main adaptations. First the randomized algorithm ApproxCycleCover is made deterministic by using the deterministic construction of neighborhood covers of [12] that uses $2^{\sqrt{\log n \cdot \log\log n}}$ rounds. In the second part, we use the derandomization of the FT-sampling, as in the minimum cut algorithm. Overall, the total round complexity is $2^{O(\sqrt{\log n \cdot \log\log n})} \cdot \mathsf{poly}(D)$. ◀

Algorithm 1 DistEdgeConnec($G = (V, E), \lambda$).

Distributed Computation of λ-edge connected cycle covers.

```
1. For $i = 1$ to $O(\lambda \cdot D)^{2\lambda}$:
     - Sample each edge $e$ into $G_i$ w.p. $p = (1 - 1/D')$.
     - $\mathcal{C}_i \leftarrow$ ApproxCycleCover($G_i, D'$).
2. $\mathcal{C} = \bigcup_i \mathcal{C}_i$.
3. For every edge $e = (u, v)$, let $G_{u,v} = \{C \in \mathcal{C} \mid e \in C\}$.
4. $\lambda(e) =$ MinCut($G_{u,v}$).
```

4 Sparse Connectivity Certificates

Finally, we consider the related problem of computing sparse connectivity certificates. We start by observing that an FT-spanner for the graph is also a connectivity certificate.

Fault Tolerant Spanners. Fault tolerant (FT) spanners [14, 4] are sparse subgraphs that preserve pairwise distances in G (up to some multiplicative stretch) even when several edges or vertices in the graph fails. These spanners have been introduced by Levcopoulos for geometric graphs [14], and later on by Chechik et al. [4] for general graphs.

▶ **Definition 23** (Fault Tolerant Spanners). *For positive integers k, f, an f edge fault-tolerant $(2k-1)$ spanner for an n-vertex graph $G = (V, E)$ is a subgraph $H \subseteq G$ satisfying that* $\mathrm{dist}(u, v, H \setminus F) \leq (2k-1)\mathrm{dist}(u, v, G \setminus F)$ *for every $u, v \in V$ and $F \subseteq E$, $|F| \leq f$. Vertex fault-tolerant spanners are defined analogously where the fault $F \subset V$.*

Chechik et al. [4] gave a generic algorithm for computing these spanners against edge failures.

▶ **Fact 24** ([4]). *Let $\mathcal{A}$ be an algorithm for computing the standard (fault-free) $(2k-1)$ spanner with $O(n^{1+1/k})$ edges and time t. Then one can compute f edge fault-tolerant $(2k-1)$ spanners with $O(f \cdot n^{1+1/k})$ edges in time $O(f \cdot t)$.*

Dinitz and Krauthgamer provided a similar transformation for vertex faults that is based on the FT-sampling technique, see Thm. 2.1 of [6].

Certificates from Fault Tolerant Spanners. The relation between FT-spanners and connectivity certificates is based on the following observation.

▶ **Observation 25.** *An f* edge *(resp., vertex) FT spanner $H \subseteq G$ is a certificate for the f* edge *(resp., vertex) connectivity of the graph.*

Proof of Observation 25. Consider a λ-edge connected graph G, and let H be an f-FT-spanner for G with $f = \lambda - 1$. We show that H is λ-edge connected, by showing that for every vertex pair s, t and a subset of F edge faults, $|F| \leq f$, there exists an s-t path in $H \setminus F$. Let P be an s-t path in $G \setminus F$. Since G is λ-connected, such a path exists. For every edge $e = (u, v) \in P \setminus H$, it holds that $\text{dist}(u, v, H \setminus F) \leq 2k - 1$, thus in particular, every neighboring pair u, v on P are connected in $H \setminus F$, the claim follows. The proof of λ-vertex connected graphs works in the same manner. ◀

Since spanners with logarithmic stretch have linear size, fault tolerant spanners with logarithmic stretch are sparse connectivity certificates. By using the existing efficient (in fact local) distributed algorithms for spanners, we get:

▶ **Observation 26.** *(1) There exists a randomized algorithm for computing a sparse λ-edge certificate with $O(\lambda n)$ edges within $O(\lambda \cdot \log^{1+o(1)} n)$ rounds, w.h.p.; (2) There exists a randomized algorithm for computing a λ-vertex certificate for the λ-vertex connectivity with $O(\lambda^2 \cdot \log n)$ edges within $O(\lambda^3 \cdot \log^{2+o(1)} n)$ rounds, w.h.p.;*

Proof. Claim (1) follows by combining the ultra-sparse spanners construction of Pettie [20] with Fact 24. To obtain sparse certificates, we use Fact 24 with algorithm $\mathcal{A}$ taken to be the ultra-sparse algorithm by Pettie [20]. This algorithm computes an $O(2^{\log^* n} \log n)$-spanner H with $O(n)$ edges within $O(\log^{1+o(1)} n)$ rounds. By taking λ disjoint copies of such spanner, constructed sequentially one after the other, we obtain a certificate with $O(\lambda \cdot n)$ edges. In the same manner, Claim (2) follows by plugging the algorithm of Pettie [20] in the meta algorithm for FT-spanners against vertex faults by Dinitz and Krauthgamer (Thm. 2.1 in [6]). ◀

While Obs. 26 gives efficient algorithm when $\lambda = O(1)$, it is less efficient for large connectivity values. In the next lemma we apply the edge sampling technique of Karger [13] to omit the dependency in λ in the round complexity. Ideas along this line appear in [10].

▶ **Lemma 27.** *For every $\lambda = \Omega(\log n)$, and $\epsilon \in [0, 1]$, given an λ-edge connected graph G, one can compute, w.h.p., an $(1 - \epsilon)\lambda$-edge sparse certificate within $O(1/\epsilon^2 \cdot \log^{2+o(1)} n)$ rounds.*

Proof. We restrict attention to $\lambda = \Omega(\log n)$, as otherwise, the lemma follows immediately from Obs. 26. The key idea for omitting the dependency in λ is by randomly decomposing the graph into spanning subgraphs each with connectivity $\min\{\lambda, \Theta(\log n/\epsilon^2)\}$ using random edge sampling, and to run the algorithm of Obs. 26 on each of the subgraphs. We randomly put each edge of G in one of μ subgraphs $G_1, \ldots, G_\mu$ for $\mu = \lceil \lambda \cdot \epsilon^2/(20 \log n) \rceil$. Karger [13] showed that each subgraph G_i has edge-connectivity in $(1 \pm \epsilon)\lambda/\mu$ with high probability. In addition, the summation of the edge-connectivities $\lambda_1, \ldots, \lambda_\mu$ of the subgraphs $G_1, \ldots, G_\mu$ is at least $\lambda(1 - \epsilon)$. The algorithm then computes a λ'-edge sparse certificate H_i for $\lambda' = (1 - \epsilon)\lambda/\mu$ in each G_i subgraph simultaneously. The output certificate H is the union of all H_i subgraphs.

We next analyze the construction, and start with round complexity. Since the G_i subgraphs are edge disjoint, applying the algorithm of Obs. 26 takes $O(\lambda' \cdot \log^{1+o(1)} n)$ rounds which is $O(\log^{2+o(1)} n)$ rounds. The edge bound follows immediately as $|E(H)| = c \cdot \mu \cdot \lambda' n = O(\lambda n)$ as required. It remains to show that H is indeed a $(1 - \epsilon)\lambda$-edge connectivity certificate. Consider a pair of vertices s, t, and a sequence of at most $(1 - \epsilon)\lambda$ edge faults F. We will show that s and t are connected in $H \setminus F$. Since the μ subgraphs are edge-disjoint, there must be a subgraph G_i containing at most $\lambda' = (1 - \epsilon)\lambda/\mu$ of the faults. Let $F_i = F \cap G_i$. Since G_i is λ'-edge connected, s and t are connected in $G_i \setminus F_i$. Since H_i is a λ'-edge certificate of G_i, it also holds that s and t are connected in $H_i \setminus F_i$ and thus also in $H \setminus F$ (i.e., as by definition, $(F \setminus F_i) \cap H_i = \emptyset$). The claim follows. ◀

References

1 Noga Alon. Explicit construction of exponential sized families of k-independent sets. *Discrete Mathematics*, 58(2):191–193, 1986.

2 Noga Alon, Shiri Chechik, and Sarel Cohen. Deterministic Combinatorial Replacement Paths and Distance Sensitivity Oracles. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece.*, pages 12:1–12:14, 2019.

3 Keren Censor-Hillel and Michal Dory. Fast Distributed Approximation for TAP and 2-Edge-Connectivity. In *21st International Conference on Principles of Distributed Systems, OPODIS 2017, Lisbon, Portugal, December 18-20, 2017*, pages 21:1–21:20, 2017.

4 Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. Fault tolerant spanners for general graphs. *SIAM Journal on Computing*, 39(7):3403–3423, 2010.

5 Mohit Daga, Monika Henzinger, Danupon Nanongkai, and Saranurak. Distributed Edge Connectivity in Sublinear Time. In *STOC*, 2019.

6 Michael Dinitz and Robert Krauthgamer. Fault-tolerant spanners: better and simpler. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 169–178. ACM, 2011.

7 Michal Dory. Distributed Approximation of Minimum k-edge-connected Spanning Subgraphs. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 149–158, 2018.

8 J Friedman. Constructing O (n log n) size monotone formulae for the k-th elementary symmetric polynomial of n Boolean variables. In *Foundations of Computer Science, 1984. 25th Annual Symposium on*, pages 506–515. IEEE, 1984.

9 Mohsen Ghaffari. *Improved Distributed Algorithms for Fundamental Graph Problems*. PhD thesis, MIT, USA, 2017. URL: `https://groups.csail.mit.edu/tds/papers/Ghaffari/PhDThesis-Ghaffari.pdf`.

10 Mohsen Ghaffari and Fabian Kuhn. Distributed minimum cut approximation. In *International Symposium on Distributed Computing*, pages 1–15. Springer, 2013.

11 Mohsen Ghaffari and Fabian Kuhn. Distributed Minimum Cut Approximation. *arXiv preprint*, 2013. `arXiv:1305.5520`.

12 Mohsen Ghaffari and Fabian Kuhn. Derandomizing Distributed Algorithms with Small Messages: Spanners and Dominating Set. In *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*, pages 29:1–29:17, 2018.

13 David R Karger. Random sampling in cut, flow, and network design problems. *Mathematics of Operations Research*, 24(2):383–413, 1999.

14 Christos Levcopoulos, Giri Narasimhan, and Michiel Smid. Efficient algorithms for constructing fault-tolerant geometric spanners. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 186–195. ACM, 1998.

15 Hiroshi Nagamochi and Toshihide Ibaraki. A linear-time algorithm for finding a sparsek-connected spanning subgraph of ak-connected graph. *Algorithmica*, 7(1-6):583–596, 1992.

16 Danupon Nanongkai and Hsin-Hao Su. Almost-tight distributed minimum cut algorithms. In *International Symposium on Distributed Computing*, pages 439–453. Springer, 2014.

17 Merav Parter and Eylon Yogev. Low Congestion Cycle Covers and Their Applications. *SODA*, 2019.

18 Merav Parter and Eylon Yogev. Optimal Short Cycle Decomposition in Almost Linear Time. *ICALP*, 2019.

19 David Peleg. *Distributed Computing: A Locality-sensitive Approach*. SIAM, 2000.

20 Seth Pettie. Distributed algorithms for ultrasparse spanners and linear size skeletons. *Distributed Computing*, 22(3):147–166, 2010. `doi:10.1007/s00446-009-0091-7`.

21 David Pritchard and Ramakrishna Thurimella. Fast computation of small cuts via cycle space sampling. *ACM Transactions on Algorithms (TALG)*, 7(4):46, 2011.

22 Václav Rozhon and Mohsen Ghaffari. Polylogarithmic-Time Deterministic Network Decomposition and Distributed Derandomization. *arXiv preprint*, 2019. `arXiv:1907.10937`.

23 Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM Journal on Computing*, 41(5):1235–1265, 2012.

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

25 Salil P. Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012. doi:10.1561/0400000010.

26 Xiao Fan Wang and Guanrong Chen. Complex networks: small-world, scale-free and beyond. *IEEE circuits and systems magazine*, 3(1):6–20, 2003.

27 Oren Weimann and Raphael Yuster. Replacement paths and distance sensitivity oracles via fast matrix multiplication. *ACM Transactions on Algorithms (TALG)*, 9(2):14, 2013.