Improved Distributed Expander Decomposition and Nearly Optimal Triangle Enumeration*

Yi-Jun Chang†
University of Michigan
cyijun@umich.edu

Thatchaphol Saranurak
Toyota Technological Institute at Chicago
saranurak@ttic.edu

ABSTRACT

An (ϵ, ϕ)-expander decomposition of a graph $G = (V, E)$ is a clustering of the vertices $V = V_1 \cup \cdots \cup V_x$ such that (1) each cluster V_i induces subgraph with conductance at least ϕ, and (2) the number of inter-cluster edges is at most $\epsilon|E|$. In this paper, we give an improved distributed expander decomposition, and obtain a nearly optimal distributed triangle enumeration algorithm in the CONGEST model.

Specifically, we construct an (ϵ, ϕ)-expander decomposition with $\phi = (\epsilon/\log n)^{2^{O(k)}}$ in $O(n^{2/k} \cdot \text{poly}(1/\phi, \log n))$ rounds for any $\epsilon \in (0, 1)$ and positive integer k. For example, a $(1/n^{o(1)}, 1/n^{o(1)})$-expander decomposition only requires $O(n^{o(1)})$ rounds to compute, which is optimal up to subpolynomial factors, and a $(0.01, 1/\text{poly}\log n)$-expander decomposition can be computed in $O(n^\gamma)$ rounds, for any arbitrarily small constant $\gamma > 0$. Previously, the algorithm by Chang, Pettie, and Zhang can construct a $(1/6, 1/\text{poly}\log n)$-expander decomposition using $\tilde{O}(n^{1-\delta})$ rounds for any $\delta > 0$, with a caveat that the algorithm is allowed to throw away a set of edges into an extra part which form a subgraph with arboricity at most n^δ. Our algorithm does not have this caveat.

By slightly modifying the distributed algorithm for routing on expanders by Ghaffari, Kuhn and Su [PODC'17], we obtain a triangle enumeration algorithm using $\tilde{O}(n^{1/3})$ rounds. This matches the lower bound by Izumi and Le Gall [PODC'17] and Pandurangan, Robinson and Scquizzato [SPAA'18] of $\tilde{\Omega}(n^{1/3})$ which holds even in the CONGESTED-CLIQUE model. To the best of our knowledge, this provides the first non-trivial example for a distributed problem that has essentially the same complexity (up to a polylogarithmic factor) in both CONGEST and CONGESTED-CLIQUE.

The key technique in our proof is the first distributed approximation algorithm for finding a low conductance cut that is as balanced as possible. Previous distributed sparse cut algorithms do not have this nearly most balanced guarantee.[1]

CCS CONCEPTS

• **Theory of computation → Distributed algorithms**.

KEYWORDS

expander decomposition, low diameter decomposition, triangle enumeration

ACM Reference Format:
Yi-Jun Chang and Thatchaphol Saranurak. 2019. Improved Distributed Expander Decomposition and Nearly Optimal Triangle Enumeration. In *2019 ACM Symposium on Principles of Distributed Computing (PODC '19), July 29-August 2, 2019, Toronto, ON, Canada.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3293611.3331618

1 INTRODUCTION

In this paper, we consider the task of finding an *expander decomposition* of a distributed network in the CONGEST model of distributed computing. Roughly speaking, an expander decomposition of a graph $G = (V, E)$ is a clustering of the vertices $V = V_1 \cup \cdots \cup V_x$ such that (1) each component V_i induces a high conductance subgraph, and (2) the number of inter-component edges is small. This natural bicriteria optimization problem of finding a good expander decomposition was introduced by Kannan Vempala and Vetta [22], and was further studied in many other subsequent works [3, 31, 32, 34, 37, 41, 43].[2] The expander decomposition has a wide range of applications, and it has been applied to solving linear systems [42], unique games [2, 36, 43], minimum cut [23], and dynamic algorithms [30].

Recently, Chang, Pettie, and Zhang [8] applied this technique to the field of distributed computing, and they showed that a variant of expander decomposition can be computed efficiently in CONGEST. Using this decomposition, they showed that triangle detection and enumeration can be solved in $\tilde{O}(n^{1/2})$ rounds.[3] The previous state-of-the-art bounds for triangle detection and enumeration were $\tilde{O}(n^{2/3})$ and $\tilde{O}(n^{3/4})$, respectively, due to Izumi and Le Gall [19]. Later, Daga et al. [11] exploit this decomposition and obtain the first algorithm for computing edge connectivity of a graph *exactly* using sub-linear number of rounds.

Specifically, the variant of the decomposition in [8] is as follows. If we allow one extra part that induces an n^δ-arboricity subgraph[4] in the decomposition, then in $O(n^{1-\delta})$ rounds we can construct

*The full version of the paper is available at arXiv [7].

†Supported by NSF grants CCF-1514383, CCF-1637546, and CCF-1815316.

[1] Kuhn and Molla [25] previously claimed that their approximate sparse cut algorithm also has the nearly most balanced guarantee, but this claim turns out to be incorrect [8, Footnote 3].

PODC '19, July 29-August 2, 2019, Toronto, ON, Canada

ACM ISBN 978-1-4503-6217-7/19/07…$15.00
https://doi.org/10.1145/3293611.3331618

[2] The existence of the expander decomposition is (implicitly) exploited first in the context of property testing [18].

[3] The $\tilde{O}(\cdot)$ notation hides any polylogarithmic factor.

[4] The arboricity of a graph is the minimum number α such that its edge set can be partitioned into α forests.

an expander decomposition in CONGEST such that each component has $1/O(\text{poly}\log n)$ conductance and the number of inter-component edges is at most $|E|/6$.

A major open problem left by the work [8] is to design an efficient distributed algorithm constructing an expander decomposition without the extra low-arboricity part. In this work, we show that this is possible. A consequence of our new expander decomposition algorithm is that triangle enumeration can be solved in $O(n^{1/3}\text{poly}\log n)$ rounds, nearly matching the $\Omega(n^{1/3}/\log n)$ lower bound [19, 33] by a polylogarithmic factor.

The CONGEST *Model.* In the CONGEST model of distributed computing, the underlying distributed network is represented as an undirected graph $G = (V, E)$, where each vertex corresponds to a computational device, and each edge corresponds to a bi-directional communication link. Each vertex v has a distinct $\Theta(\log n)$-bit identifier $\text{ID}(v)$. The computation proceeds according to synchronized *rounds*. In each round, each vertex v can perform unlimited local computation, and may send a distinct $O(\log n)$-bit message to each of its neighbors. Throughout the paper we only consider the randomized variant of CONGEST. Each vertex is allowed to generate unlimited local random bits, but there is no global randomness. We say that an algorithm succeeds *with high probability* (w.h.p.) if its failure probability is at most $1/\text{poly}(n)$.

The CONGESTED-CLIQUE model is a variant of CONGEST that allows all-to-all communication, and the LOCAL model is a variant of CONGEST that allows messages of unbounded length.

Terminology. Before we proceed, we review the graph terminologies related to the expander decomposition. Consider a graph $G = (V, E)$. For a vertex subset S, we write $\text{Vol}(S)$ to denote $\sum_{v\in S}\deg(v)$. Note that by default the degree is with respect to the original graph G. We write $\bar{S} = V \setminus S$, and let $\partial(S) = E(S, \bar{S})$ be the set of edges $e = \{u, v\}$ with $u \in S$ and $v \in \bar{S}$. The *sparsity* or *conductance* of a cut $(S, \bar{S})$ is defined as $\Phi(S) = |\partial(S)|/\min\{\text{Vol}(S), \text{Vol}(\bar{S})\}$. The *conductance* Φ_G of a graph G is the minimum value of $\Phi(S)$ over all vertex subsets S. Define the *balance* $\text{bal}(S)$ of a cut S by $\text{bal}(S) = \min\{\text{Vol}(S), \text{Vol}(\bar{S})\}/\text{Vol}(V)$. We say that S is a *most-balanced* cut of G of conductance at most ϕ if $\text{bal}(S)$ is maximized among all cuts of G with conductance at most ϕ. We have the following relation [20] between the mixing time $\tau_{\text{mix}}(G)$ and conductance Φ_G:

$$\Theta\left(\frac{1}{\Phi_G}\right) \le \tau_{\text{mix}}(G) \le \Theta\left(\frac{\log n}{\Phi_G^2}\right).$$

Let S be a vertex set. Denote $E(S)$ by the set of all edges whose two endpoints are both within S. We write $G[S]$ to denote the subgraph induced by S, and we write $G\{S\}$ to denote the graph resulting from adding $\deg_V(v) - \deg_S(v)$ self loops to each vertex v in $G[S]$. Note that the degree of each vertex $v \in S$ in both G and $G\{S\}$ is identical. As in [39], each self loop of v contributes 1 in the calculation of $\deg(v)$. Observe that we always have

$$\Phi(G\{S\}) \le \Phi(G[S]).$$

Let v be a vertex. Denote $N(v)$ as the set of neighbors of v. We also write $N^k(v) = \{u \in V \mid \text{dist}(u, v) \le k\}$. Note that $N^1(v) = N(v) \cup \{v\}$. These notations $\text{dist}(u, v)$, $N(v)$, and $N^k(v)$ depend on the underlying graph G. When the choice of underlying graph is not clear from the context, we use a subscript to indicate the underlying graph we refer to.

Expander Decomposition. An (ϵ, ϕ)-expander decomposition of a graph $G = (V, E)$ is defined as a partition of the vertex set $V = V_1 \cup \cdots \cup V_x$ satisfying the following conditions.

- For each component V_i, we have $\Phi(G\{V_i\}) \ge \phi$.
- The number of inter-component edges $(|\partial(V_1)| + \cdots + |\partial(V_x)|)/2$ is at most $\epsilon|E|$.

The main contribution of this paper is the following result.

Theorem 1. *Let $\epsilon \in (0, 1)$, and let k be a positive integer. An (ϵ, ϕ)-expander decomposition with $\phi = (\epsilon/\log n)^{2^{O(k)}}$ can be constructed in $O\left(n^{2/k} \cdot \text{poly}(1/\phi, \log n)\right) = O\left(n^{2/k} \cdot \left(\frac{\log n}{\epsilon}\right)^{2^{O(k)}}\right)$ rounds, w.h.p.*

The proof of Theorem 1 is in Section 2. We emphasize that the number of rounds does not depend on the diameter of G. There is a trade-off between the two parameters ϵ and ϕ. For example, an (ϵ, ϕ)-expander decomposition with $\epsilon = 2^{-\log^{1/3} n}$ and $\phi = 2^{-\log^{2/3} n}$ can be constructed in $n^{O(1/\log\log n)}$ rounds by setting $k = O(\log\log n)$ in Theorem 1. If we are allowed to have $\epsilon = 0.01$ and spend $O(n^{0.01})$ rounds, then we can achieve $\phi = 1/O(\text{poly}\log n)$.

Distributed Triangle Finding. Variants of the triangle finding problem have been studied in the literature [1, 5, 8, 12–14, 19, 33]. In the *triangle detection* problem, it is required that at least one vertex must report a triangle if the graph has at least one triangle. In the *triangle enumeration* problem, it is required that each triangle of the graph is reported by at least one vertex. Both of these problems can be solved in $O(1)$ rounds in LOCAL. It is the bandwidth constraint of CONGEST and CONGESTED-CLIQUE that makes these problems non-trivial.

It is important that a triangle $T = \{u, v, w\}$ is allowed to be reported by a vertex $x \notin T$. If it is required that a triangle $T = \{u, v, w\}$ has to be reported by a vertex $x \in T$, then there is an $\Omega(n/\log n)$ lower bound [19] for triangle enumeration, in both CONGEST and CONGESTED-CLIQUE. To achieve a round complexity of $o(n/\log n)$, it is necessary that some triangles T are reported by vertices not in T.

Dolev, Lenzen, and Peled [12] showed that triangle enumeration can be solved deterministically in $O(n^{1/3}/\log n)$ rounds in CONGESTED-CLIQUE. This algorithm is optimal, as it matches the $\Omega(n^{1/3}/\log n)$-round lower bound [19, 33] in CONGESTED-CLIQUE. Interestingly, if we only want to detect one triangle or count the number of triangles, then Censor-Hillel et al. [5] showed that the round complexity in CONGESTED-CLIQUE can be improved to $\tilde{O}(n^{1-(2/\omega)+o(1)}) = o(n^{0.158})$ time [5], where $\omega < 2.373$ is the exponent of the complexity of matrix multiplication [26].

For the CONGEST model, Izumi and Le Gall [19] showed that the triangle detection and enumeration problems can be solved in $\tilde{O}(n^{2/3})$ and $\tilde{O}(n^{3/4})$ time, respectively. These upper bounds were later improved to $\tilde{O}(n^{1/2})$ by Chang, Pettie, and Zhang using a variant of expander decomposition [8].

A consequence of Theorem 1 is that triangle enumeration (and hence detection) can be solved in $\tilde{O}(n^{1/3})$ rounds, almost matching the $\Omega(n^{1/3}/\log n)$ lower bound [19, 33] which holds even in

CONGESTED-CLIQUE. To the best of our knowledge, this provides the first non-trivial example for a distributed problem that has essentially the same complexity (up to a polylogarithmic factor) in both CONGEST and CONGESTED-CLIQUE, i.e., allowing non-local communication links does not help. In contrast, many other graph problems can be solved much more efficiently in CONGESTED-CLIQUE than in CONGEST; see e.g., [17, 21].

Theorem 2. *Triangle enumeration can be solved in $\tilde{O}(n^{1/3})$ rounds in* CONGEST*, w.h.p.*

The proof of Theorem 2 is in Section 3. Note that Theorem 2 immediately implies an algorithm for triangle detection with the same number of rounds. However, while the best known lower bounds [1, 14] for triangle detection can currently exclude only 1-round algorithms. Whether the large gap between upper and lower bounds for this problem can be closed remains an intriguing question.

1.1 Prior Work on Expander Decomposition

In the centralized setting, the first polynomial time algorithm for construction an (ϵ, ϕ)-expander decomposition is by Kannan, Vempala and Vetta [22] where $\epsilon = \tilde{O}(\phi)$. Afterward, Spielman and Teng [40, 41] significantly improved the running time to be near-linear in m, where m is the number of edges. In time $\tilde{O}(m/\text{poly}(\phi))$, they can construct a "weak" $(\text{poly}(\phi, \log n), \phi)$-expander decomposition. Their weak expander only has the following weaker guarantee that each part V_i in the partition of V might not induce an expander, and we only know that V_i is contained in some *unknown* expander. That is, there exists some $W_i \supseteq V_i$ where $\Phi_{G\{W_i\}} \geq \phi$. Although this guarantee suffices for many applications (e.g. [10, 24]), some other applications [9, 30], including the triangle enumeration algorithm of [8], crucially needs the fact that each part in the decomposition induces an expander.

Nanongkai and Saranurak [29] and, independently, Wulff-Nilsen [44] gave a fast algorithm without weakening the guarantee as the one in [40, 41]. In [29], their algorithm finds a $(\phi \log^{O(k)} n, \phi)$-expander decomposition in time $\tilde{O}(m^{1+1/k})$. Although the trade-off is worse in [44], their high-level approaches are in fact the same. They gave the same black-box reduction from constructing an expander decomposition to finding a nearly most balanced sparse cut. The difference only comes from the quality of their nearly most balanced sparse cuts algorithms. Our distributed algorithm will also follow this high-level approach.

Most recently, Saranurak and Wang [37] gave a $(\tilde{O}(\phi), \phi)$-expander decomposition algorithm with running time $\tilde{O}(m/\phi)$. This is optimal up to a polylogarithmic factor when $\phi \geq 1/\text{poly}\log(n)$. We do not use their approach, as their *trimming step* seems to be inherently sequential and very challenging to parallelize or make distributed.

The only previous expander decomposition in the distributed setting is by Chang, Pettie, and Zhang [8]. Their distributed algorithm gave an $(1/6, 1/\text{poly}\log(n))$-expander decomposition with an extra part which is an n^δ-arboricity subgraph in $O(n^{1-\delta})$ rounds in CONGEST. Our distributed algorithm significantly improved upon this work.

1.2 Technical Overview

For convenience, we call a cut with conductance at most ϕ a *ϕ-sparse cut* in this section. To give a high-level idea, the most straightforward algorithm for constructing an expander decomposition of a graph $G = (V, E)$ is as follows. Find a ϕ-sparse cut S. If such a cut S does not exist, then return V as a part in the partition. Otherwise, recurse on both sides $G\{S\}$ and $G\{V - S\}$, and so the edges in $E(S, V - S)$ become inter-cluster edges. To see the correctness, once the recursion stops at $G\{U\}$ for some U, we know that $\Phi_{G\{U\}} \geq \phi$. Also, the total number of inter-cluster edges is at most $O(m\phi \log n)$ because (1) each inter-cluster edge can be charged to edges in the smaller side of some ϕ-sparse cut, and (2) each edge can be in the smaller side of the cut for at most $O(\log n)$ times.

This straightforward approach has two efficiency issues: (1) checking whether a ϕ-sparse cut exists does not admit fast distributed algorithms (and is in fact NP-hard), and (2) a ϕ-sparse cut S can be very unbalanced and hence the recursion depth can be as large as $\Omega(n)$. Thus, even if we ignore time spent on finding cuts, the round complexity due to the recursion depth is too high. At a high-level, all previous algorithms (both centralized and distributed) handle the two issues in the same way up to some extent. First, they instead use *approximate sparse cut algorithms* which either find some ϕ'-sparse cut or certify that there is no ϕ-sparse cut where $\phi' \gg \phi$. Second, they find a cut with some guarantee about the balance of the cut, i.e., the smaller side of the cut should be sufficiently large.

Let us contrast our approach with the only previous distributed expander decomposition algorithm by Chang, Pettie, and Zhang [8]. They gave an approximate sparse cut algorithm such that the smaller side of the cut has $\Omega(n^\delta)$ vertices for some constant $\delta > 0$, so the recursion depth is $O(n^{1-\delta})$. They guarantee this property by "forcing" the graph to have minimum degree at least n^δ, so any ϕ-sparse cut must contain $\Omega(n^\delta)$ vertices (this uses the fact that the graph is simple) To force the graph to have high degree, they keep removing vertices with degree at most n^δ at any step of the algorithms. Throughout the whole algorithm, the removed part form a graph with arboricity at most n^δ. This explains why their decomposition outputs the extra part which induces a low arboricity subgraph. With some other ideas on distributed implementation, they obtained the round complexity of $\tilde{O}(n^{1-\delta})$, roughly matching the recursion depth.

In this paper, we avoid this extra low-arboricity part. The key component is the following. Instead of just guaranteeing that the smaller side of the cut has $\Omega(n^\delta)$ vertices, we give the first efficient distributed algorithm for computing a *nearly most balanced* sparse cut. Suppose there is a ϕ-sparse cut with balance b, then our sparse cut algorithm returns a ϕ'-sparse cut with balance at least $\Omega(b)$, where ϕ' is not much larger than ϕ. Intuitively, given that we can find a nearly most balanced sparse cut efficiently, the recursion depth should be made very small. This intuition can be made formal using the ideas in the centralized setting from Nanongkai and Saranurak [29] and Wullf-Nilsen [44]. Our main technical contribution is two-fold. First, we show the first distributed algorithm for computing a nearly most balanced sparse cut, which is our key algorithmic tool. Second, in order to obtain a fast distributed algorithm, we must modify the centralized approach of [29, 44] on how

to construct an expander decomposition. In particular, we need to run a *low diameter decomposition* whenever we encounter a graph with high diameter, as our distributed algorithm for finding a nearly most balanced sparse cut is fast only on graphs with low diameter.

Sparse Cut Computation. At a high level, our distributed nearly most balanced sparse cut algorithm is a distributed implementation of the sequential algorithm of Spielman and Teng [41]. The algorithm of [41] involves $\tilde{O}(m)$ *sequential* iterations of `Nibble` with a random starting vertex on the remaining subgraph. Roughly speaking, the procedure `Nibble` aims at finding a sparse cut by simulating a random walk. The idea is that if the starting vertex v belongs to some sparse cut S, then it is likely that most of the probability mass will be trapped inside S. Chang, Pettie, and Zhang [8] showed that $\tilde{O}(m)$ simultaneous iterations of an approximate version of `Nibble` with a random starting vertex can be implemented efficiently in CONGEST in $O(\text{poly}(1/\phi, \log n))$ rounds, where ϕ is the target conductance. A major difference between this work and [8] is that the expander decomposition algorithm of [8] does not need any requirement about the balance of the cut in their sparse cut computation.

Note that the $\tilde{O}(m)$ *sequential* iterations of `Nibble` in the nearly most balanced sparse cut algorithm of [41] cannot be completely parallelized. For example, it is possible that the union of all $\tilde{O}(m)$ output of `Nibble` equals the entire graph. Nonetheless, we show that this process can be partially parallelized at the cost of worsening the conductance guarantee by a polylogarithmic factor.

Theorem 3 (Nearly most balanced sparse cut)**.** *Given a parameter $\phi = O(1/\log^5 n)$, there is an $O(D \cdot \text{poly}(\log n, 1/\phi))$-round algorithm $\mathcal{A}$ that achieves the following w.h.p.*

- *In case $\Phi(G) \le \phi$, the algorithm $\mathcal{A}$ is guaranteed to return a cut C with balance $\text{bal}(C) \ge \min\{b/2, 1/48\}$ and conductance $\Phi(C) = O(\phi^{1/3} \log^{5/3} n)$, where b is defined as $b = \text{bal}(S)$, where S is a most-balanced sparse cut of G of conductance at most ϕ.*
- *In case $\Phi(G) > \phi$, the algorithm $\mathcal{A}$ either returns $C = \emptyset$ or returns a cut C with conductance $\Phi(C) = O(\phi^{1/3} \log^{5/3} n)$.*

The proof of Theorem 3 is in the full version of the paper [7]. We note again that this is the first distributed sparse cut algorithm with a nearly most balanced guarantee. The problem of finding a sparse cut the distributed setting has been studied prior to the work of [8]. Given that there is a ϕ-sparse cut and balance b, the algorithm of Das Sarma, Molla, and Pandurangan [38] finds a cut of conductance at most $\tilde{O}(\sqrt{\phi})$ in $\tilde{O}((n + (1/\phi))/b)$ rounds in CONGEST. The round complexity was later improved to $\tilde{O}(D + 1/(b\phi))$ by Kuhn and Molla [25]. These prior works have the following drawbacks: (1) their running time depends on b which can be as small as $O(1/n)$, and (2) their output cuts are not guaranteed to be nearly most balanced (see footnote 1).

Low Diameter Decomposition. The runtime of our distributed sparse cut algorithm (Theorem 3) is proportional to the diameter. To avoid running this algorithm on a high diameter graph, we employ a low diameter decomposition to decompose the current graph into components of small diameter.

The low diameter decomposition algorithm of Miller, Peng, and Xu [28] can already be implemented in CONGEST efficiently. Roughly, their algorithm is to let each vertex v sample $\delta_v \sim \text{Exponential}(\beta)$, $\beta \in (0, 1)$, and then v is assigned to the cluster of u that minimizes $\text{dist}(u, v) - \delta_u$. A similar approach has been applied to construct a *network decomposition* [4, 27].

However, there is one subtle issue that the guarantee that the number of inter-cluster edges is at most $O(\beta|E|)$ only holds *in expectation*. In sequential or parallel computation model, we can simply repeat the procedure for several times and take the best result. In CONGEST, this however takes at least diameter time, which is inefficient when the diameter is large.

We provide a technique that allows us to achieve this guarantee *with high probability* without spending diameter time, so we can ensure that the number of inter-cluster edges is small with high probability in our expander decomposition algorithm.[5]

Intuitively, the main barrier needed to be overcome is the high dependence among the $|E|$ events that an edge $\{u, v\}$ has its endpoints in different clusters. Our strategy is to compute a partition $V = V_D \cup V_S$ in such a way that V_D already induces a low diameter clustering, and the edges incident to V_S satisfy the property that if we run the the low diameter decomposition algorithm of [28], the events that they are inter-cluster have sufficiently small dependence. Then we can use a variant of Chernoff bound with bounded dependence [35] to bound the number of inter-cluster edges with high probability.

Theorem 4 (Low diameter decomposition)**.** *Let $\beta \in (0, 1)$. There is an $O(\text{poly}(\log n, 1/\beta))$-round algorithm $\mathcal{A}$ that finds a partition of the vertex set $V = V_1 \cup \cdots \cup V_x$ satisfying the following conditions w.h.p.*

- *Each component V_i has diameter $O\left(\frac{\log^2 n}{\beta^2}\right)$.*
- *The number of inter-component edges $(|\partial(V_1)| + \cdots + |\partial(V_x)|)/2$ is at most $\beta|E|$.*

Adapting the algorithm of [28] to CONGEST, in $O\left(\frac{\log n}{\beta}\right)$ rounds we can decompose the graph into components of diameter $O\left(\frac{\log n}{\beta}\right)$ such that the number of inter-component edges is $O(\beta|E|)$ *in expectation*. In the full version of the paper [7] we extend this result to obtain a high probability bound and prove Theorem 4.

Triangle Enumeration. Incorporating our expander decomposition algorithm (Theorem 1) with the triangle enumeration algorithm of [8, 15], we immediately obtain an $\tilde{O}(n^{1/3}) \cdot 2^{O(\sqrt{\log n})}$-round algorithm for triangle enumeration. This round complexity can be further improved to $\tilde{O}(n^{1/3})$ by adjusting the routing algorithm of Ghaffari, Kuhn, and Su [15] on graphs of small mixing time. The main observation is their algorithm can be viewed as a distributed data structure with a trade-off between the query time and the pre-processing time. In particular, for any given constant $\epsilon > 0$, it is possible to achieve $O(\text{poly} \log n)$ query time by spending $O(n^\epsilon)$ time on pre-processing.

[5] We remark that the triangle enumeration algorithm of [8] still works even if the guarantee on the number of inter-cluster edges in the expander decomposition only holds in expectation.

2 EXPANDER DECOMPOSITION

The goal of this section is to prove Theorem 1.

Theorem 1. *Let $\epsilon \in (0,1)$, and let k be a positive integer. An (ϵ, ϕ)-expander decomposition with $\phi = (\epsilon/\log n)^{2^{O(k)}}$ can be constructed in $O\left(n^{2/k} \cdot \mathrm{poly}(1/\phi, \log n)\right) = O\left(n^{2/k} \cdot \left(\frac{\log n}{\epsilon}\right)^{2^{O(k)}}\right)$ rounds, w.h.p.*

For the sake of convenience, we denote

$$h(\theta) = \Theta\left(\theta^{1/3} \log^{5/3} n\right)$$

as an increasing function associated with Theorem 3 such that when we run the nearly most balanced sparse cut algorithm of Theorem 3 with conductance parameter θ, if the output subset C is non-empty, then it has $\Phi(C) \le h(\theta)$. We note that

$$h^{-1}(\theta) = \Theta\left(\theta^3/\log^5 n\right).$$

Let $\epsilon \in (0,1)$ and $k \ge 1$ be the parameters specified in Theorem 1. We define the following parameters that are used in our algorithm.

Nearly Most Balanced Sparse Cut: We define $\phi_0 = O(\epsilon^2/\log^7 n)$ in such a way that when we run the nearly most balanced sparse cut algorithm with this conductance parameter, any non-empty output C must satisfy $\Phi(C) \le h(\phi_0) = \frac{\epsilon/6}{\log\binom{n}{2}}$. For each $1 \le i \le k$, we define $\phi_i = h^{-1}(\phi_{i-1})$.

Low Diameter Decomposition: The parameter $\beta = O(\epsilon^2/\log n)$ for the low diameter decomposition is chosen as follows. Set $d = O((1/\epsilon)\log n)$ as the smallest integer such that $(1-\epsilon/12)^d \cdot 2\binom{n}{2} < 1$. Then we define $\beta = (\epsilon/3)/d$.

We show that an (ϵ, ϕ)-expander decomposition can be constructed in $O\left(n^{2/k} \cdot \mathrm{poly}(1/\phi, \log n)\right)$ rounds, with conductance parameter $\phi = \phi_k = (\epsilon/\log n)^{2^{O(k)}}$. We will later see that $\phi = \phi_k$ is the smallest conductance parameter we ever use for applying the nearly most balanced sparse cut algorithm.

Algorithm. Our algorithm has two phases. In the algorithm there are three places where we remove edges from the graph, and they are tagged with Remove-j, for $1 \le j \le 3$ for convenience. Whenever we remove an edge $e = \{u, v\}$, we add a self loop at both u and v, and so the degree of a vertex never changes throughout the algorithm. We never remove self loops.

At the end of the algorithm, V is partitioned into connected components $V_1, \ldots, V_x$ induced by the remaining edges. To prove the correctness of the algorithm, we will show that the number of removed edges is at most $\epsilon|E|$, and $\Phi_{G\{V_i\}} \ge \phi$ for each component V_i.

Phase 1.

The input graph is $G = (V, E)$.

(1) Do the low diameter decomposition algorithm (Theorem 4) with parameter β on G. Remove all inter-cluster edges (Remove-1).
(2) For each connected component U of the graph, do the nearly most balanced sparse cut algorithm (Theorem 3) with parameter ϕ_0 on $G\{U\}$. Let C be the output subset.
   (a) If $C = \emptyset$, then the subgraph $G^* = G\{U\}$ quits Phase 1.
   (b) If $C \ne \emptyset$ and $\mathrm{Vol}(C) \le (\epsilon/12)\mathrm{Vol}(U)$, then the subgraph $G^* = G\{U\}$ quits Phase 1 and enters Phase 2.
   (c) Otherwise, remove the cut edges $E(C, U \setminus C)$ (Remove-2), and then we recurse on both sides $G\{C\}$ and $G\{U \setminus C\}$ of the cut.

We emphasize that we do not remove the cut edges in Step 2b of Phase 1.

Lemma 1. *The depth of the recursion of Phase 1 is at most d.*

Proof. Suppose there is still a component U entering the depth $d+1$ of the recursion of Phase 1. Then according to the threshold for $\mathrm{Vol}(C)$ specified in Step 2b, we infer that $\mathrm{Vol}(U) \le (1-\epsilon/12)^d \mathrm{Vol}(V) < 1$ by our choice of d, which is impossible. □

Phase 2.

The input graph is $G^* = G\{U\}$. Define $\tau \stackrel{\text{def}}{=} ((\epsilon/6)\cdot \mathrm{Vol}(U))^{1/k}$. Define the sequence: $m_1 \stackrel{\text{def}}{=} (\epsilon/6)\cdot\mathrm{Vol}(U)$, and $m_i \stackrel{\text{def}}{=} m_{i-1}/\tau$, for each $1 < i \le k+1$. Initialize $L \leftarrow 1$ and $U' \leftarrow U$. Repeatedly do the following procedure.

- Do the nearly most balanced sparse cut algorithm (Theorem 3) with parameter ϕ_L on $G\{U'\}$. Let C be the output subset. Note that $\Phi_{G\{U'\}}(C) \le \phi_{L-1}$.
  - If $C = \emptyset$, then the subgraph $G\{U'\}$ quits Phase 2.
  - If $C \ne \emptyset$ and $\mathrm{Vol}(C) \le m_L/(2\tau)$, then update $L \leftarrow L+1$.
  - Otherwise, update $U' \leftarrow U' \setminus C$, and remove all edges incident to C (Remove-3).

Intuitively, in Phase 2 we keep calling the nearly most balanced sparse cut algorithm to find a cut C and remove it. If we find a cut C that has volume greater than $m_L/(2\tau)$, then we make a good progress. If $\mathrm{Vol}(C) \le m_L/(2\tau)$, then we learn that the volume of the most balanced sparse cut of conductance at most ϕ_L is at most $2 \cdot m_L/(2\tau) = m_L/\tau = m_{L+1}$ by Theorem 3, and so we move on to the next level by setting $L \leftarrow L+1$.

The maximum possible level L is k. Since by definition $m_k/(2\tau) = 1/2 < 1$, there is no possibility to increase L to $k+1$. Once we reach $L = k$, we will repeatedly run the nearly most balanced sparse cut algorithm until we get $C = \emptyset$ and quit.

When we remove a cut $C \ne \emptyset$ in Phase 2, each $u \in C$ becomes an isolated vertex with $\deg(u)$ self loops, as all edges incident to u have been removed, and so in the final decomposition $V = V_1 \cup \cdots \cup V_x$ we have $V_i = \{u\}$ for some i. We emphasize that we only do the edge removal when $\mathrm{Vol}(C) > m_L/(2\tau)$. Lemma 2 bounds the volume of the cuts found during Phase 2.

Lemma 2. *For each $1 \le i \le k$, define C_i as the union of all subsets C found in Phase 2 when $L \ge i$. Then either $C_i = \emptyset$ or $\mathrm{Vol}(C_i) \le m_i$.*

Proof. We first consider the case of $i = 1$. Observe that the graph $G^* = G\{U\}$ satisfies the property that the most balanced sparse cut

of conductance at most ϕ_0 has balance at most $2(\epsilon/12) = \epsilon/6$, since otherwise it does not meet the condition for entering Phase 2. Note that all cuts we find during Phase 2 have conductance at most ϕ_0, and so the union of them C_1 is also a cut of G^* with conductance at most ϕ_0. This implies that $\mathrm{Vol}(C_1) \le (\epsilon/6)\,\mathrm{Vol}(U) = m_1$.

The proof for the case of $2 \le i \le k$ is exactly the same, as the condition for increasing L is to have $\mathrm{Vol}(C) \le m_L/(2\tau)$. Let $G' = G\{U'\}$ be the graph considered in the iteration when we increase $L = i-1$ to $L = i$. The existence of such a cut C of G' implies that the most balanced sparse cut of conductance at most ϕ_{i-1} of G' has volume at most $2\,\mathrm{Vol}(C) \le m_{i-1}/\tau = m_i$. Similarly, note that all cuts we find when $L \ge i$ have conductance at most ϕ_{i-1}, and so the union of them C_i is also a cut of G' with conductance at most ϕ_{i-1}. This implies that $\mathrm{Vol}(C_i) \le m_i$. □

Conductance of Remaining Components. For each $u \in V$, there are two possible ways for u to end the algorithm:

- During Phase 1 or Phase 2, the output of the nearly most balanced sparse cut algorithm on the component that u belongs to is $C = \emptyset$. In this case, the component that u belongs to becomes a component V_i in the final decomposition $V = V_1 \cup \cdots \cup V_x$. If ϕ' is the conductance parameter used in the nearly most balanced sparse cut algorithm, then $\Phi(G\{V_i\}) \ge \phi'$. Note that $\phi' \ge \phi_k = \phi$.
- During Phase 2, $u \in C$ for the output C of the nearly most balanced sparse cut algorithm. In this case, u itself becomes a component $V_i = \{u\}$ in the final decomposition $V = V_1 \cup \cdots \cup V_x$. Trivially, we have $\Phi(G\{V_i\}) \ge \phi$.

Therefore, we conclude that each component V_i in the final decomposition $V = V_1 \cup \cdots \cup V_x$ satisfies that $\Phi(G\{V_i\}) \ge \phi$.

Number of Removed Edges. There are three places in the algorithm where we remove edges. We show that, for each $1 \le j \le 3$, the number of edges removed due to Remove-j is at most $(\epsilon/3)|E|$, and so the total number of inter-component edges in the final decomposition $V = V_1 \cup \cdots \cup V_x$ is at most $\epsilon|E|$.

(1) By Lemma 1, the depth of recursion of Phase 1 is at most d. For each $i = 1$ to d, the number of edges removed due to the low diameter decomposition algorithm during depth i of the recursion is at most $\beta|E|$. By our choice of β, the number of edges removed due to Remove-1 is at most $d \cdot \beta|E| \le (\epsilon/3)|E|$.
(2) For each edge $e \in E(C, U \setminus C)$ removed due to the nearly most balanced sparse cut algorithm in Phase 1, we charge the cost of the edge removal to some pairs (v, e) in the following way. If $\mathrm{Vol}(C) < \mathrm{Vol}(U \setminus C)$, for each $v \in C$, and for each edge e incident to v, we charge the amount $|E(C, U \setminus C)|/\mathrm{Vol}(C)$ to (v, e); otherwise, for each $v \in U \setminus C$, and for each edge e incident to v, we charge the amount $|E(C, U \setminus C)|/\mathrm{Vol}(U \setminus C)$ to (v, e). Note that each pair (v, e) is being charged for at most $\log |E|$ times throughout the algorithm, and the amount per charging is at most $h(\phi_0)$. Therefore, the number of edges removed due to Remove-2 is at most $(\log |E|) \cdot h(\phi_0) \cdot 2|E| \le (\epsilon/3)|E|$ by our choice of ϕ_0.
(3) By Lemma 2, the summation of $\mathrm{Vol}(C)$ over all cuts C in $G^* = G\{U\}$ that are found and removed during Phase 2 due to Remove-3 is at most $m_1 = (\epsilon/6)\,\mathrm{Vol}(U) \le (\epsilon/3)|E|$.

Round Complexity. During Phase 1, each vertex participates in at most $d = O((1/\epsilon)\log n)$ times the nearly most balanced sparse cut algorithm and the low diameter decomposition algorithm. By our choice of parameters $\beta = O(\epsilon^2/\log n)$ and $\phi_0 = O(\epsilon^2/\log^7 n)$, the round complexity of both algorithms are $O(\mathrm{poly}(1/\epsilon, \log n))$, as we note that whenever we run the nearly most balanced sparse cut algorithm, the diameter of each connected component is at most $O\left(\frac{\log^2 n}{\beta^2}\right) = O\left(\frac{\log^4 n}{\epsilon^4}\right)$.

For Phase 2, Lemma 2 guarantees that for each $1 \le i \le k$ the algorithm can stay $L = i$ for at most 2τ iterations. If we neither increase L nor quit Phase 2 for 2τ iterations, then we have $\mathrm{Vol}(C_L) > m_L$, which is impossible. Therefore, the round complexity for Phase 2 can be upper bounded by

$$2\tau \sum_{i=1}^{k} O(\mathrm{poly}(1/\phi_i, \log n)) \le O\left(n^{2/k} \cdot \mathrm{poly}(1/\phi, \log n)\right).$$

During Phase 2, it is possible that the graph $G\{U'\}$ be disconnected or has a large diameter, but we are fine since we can use all edges in G^* for communication during a sparse cut computation, and the diameter of G^* is at most $O\left(\frac{\log^2 n}{\beta^2}\right) = O\left(\frac{\log^4 n}{\epsilon^4}\right)$.

3 TRIANGLE ENUMERATION

We show how to derive Theorem 2 by combining Theorem 1 with other known results in [8, 15].

Theorem 2. *Triangle enumeration can be solved in $\tilde{O}(n^{1/3})$ rounds in* CONGEST*, w.h.p.*

Chang, Pettie, and Zhang [8] showed that given an (ϵ, ϕ)-expander decomposition $V = V_1 \cup \ldots \cup V_x$ with $\epsilon \le 1/6$, there is an algorithm $\mathcal{A}$ that finds an edge subset $E^* \subseteq E$ with $|E^*| \le |E|/2$ such that each triangle in G is detected by some vertex during the execution of $\mathcal{A}$, except the triangles whose three edges are all within $|E^*|$. The algorithm $\mathcal{A}$ has to solve $\tilde{O}(n^{1/3})$ times the following routing problem in each $G[V_i]$. Given a set of routing requests where each vertex v is a source or a destination for at most $O(\deg(v))$ messages of $O(\log n)$ bits, the goal is to deliver all messages to their destinations. Ghaffari, Khun, and Su [15] showed that this routing problem can be solved in $2^{O(\sqrt{\log n \log\log n})} \cdot O(\tau_{\mathrm{mix}})$ rounds. This was later improved to $2^{O(\sqrt{\log n})} \cdot O(\tau_{\mathrm{mix}})$ by Ghaffari and Li [16].

Applying our distributed expander decomposition algorithm (Theorem 1), we can find an (ϵ, ϕ)-expander decomposition with $\epsilon \le 1/6$ and $\phi = 1/O(\mathrm{poly}\log n)$ in $o(n^{1/3})$ rounds by selecting k to be a sufficiently large constant. The mixing time τ_{mix} of each component $G[V_i]$ is at most $O\left(\frac{\log n}{\phi^2}\right) = O(\mathrm{poly}\log n)$. Then we apply the above algorithm $\mathcal{A}$, and it takes $2^{O(\sqrt{\log n})} \cdot O(\tau_{\mathrm{mix}}) = 2^{O(\sqrt{\log n})}$ rounds with the routing algorithm of Ghaffari and Li [16]. After that, we recurse on the edge set E^*, and we are done enumerating all triangles after $O(\log n)$ iterations. This concludes the $O(n^{1/3}) \cdot 2^{O(\sqrt{\log n})}$-round algorithm for triangle enumeration.

To improve the complexity to $\tilde{O}(n^{1/3})$, we make the observation that the routing algorithm of [15] can be seen as a distributed data structure with the following properties.

Parameters: The parameter k is a positive integer that specifies the depth of the hierarchical structure in the routing algorithm. Given k, define β as the number such that $k = \log_\beta m$, where m is the total number of edges.

Pre-processing Time: The algorithm for building the data structure consists of two parts. The round complexity for building the hierarchical structure is $O(k\beta)(\log n)^{O(k)} \cdot O(\tau_{\text{mix}})$ [15, Lemma 3.2]. The round complexity for adding the portals is $O(k\beta^2 \log n) \cdot O(\tau_{\text{mix}})$ [15, Lemma 3.3]

Query Time: After building the data structure, each routing task can be solved in $(\log n)^{O(k)} \cdot O(\tau_{\text{mix}})$ rounds [15, Lemma 3.4].

The parameter k can be chosen as any positive integer. In [15] they used $k = \Theta(\sqrt{\log n / \log\log n})$ to balance the pre-processing time and the query time to show that the routing task can be solved in $2^{O(\sqrt{\log n \log\log n})} \cdot O(\tau_{\text{mix}})$ rounds. This round complexity was later improved to $2^{O(\sqrt{\log n})} \cdot O(\tau_{\text{mix}})$ in [16]. We however note that the algorithm of [16] does not admit a trade-off as above. The main reason is their special treatment of the base layer G_0 of the hierarchical structure. In [16], G_0 is a random graph with degree $2^{O(\sqrt{\log n})}$, and simulating one round in G_0 already costs $2^{O(\sqrt{\log n})}$. τ_{mix} rounds in the original graph G.

In the triangle enumeration algorithm $\mathcal{A}$, we need to query this distributed data structure for $\tilde{O}(n^{1/3})$ times. It is possible to set k to be a large enough constant so that the pre-processing time costs only $o(n^{1/3})$ rounds, while the query time is still $O(\text{poly}\log n)$. This implies that the triangle enumeration problem can be solved in $\tilde{O}(n^{1/3})$ rounds.

4 OPEN PROBLEMS

In this paper, we designed a new expander decomposition algorithm that get rids of the low-arboricity part needed in [8], and this implies that triangle enumeration can be solved in $\tilde{O}(n^{1/3})$ rounds, which is optimal up to a polylogarithmic factor.

Many interesting problems are left open. In particular, the current exponent of the polylogarithmic gap between the lower and the upper bounds is enormous. The huge exponent is caused by the inefficient trade-off between the parameters in the (i) hierarchical routing structure and the (ii) expander decomposition algorithm. Improving the current state of the art of (i) and (ii) will lead to an improved upper bound for triangle enumeration, as well as several other problems [11, 15, 16].

We note that the lower bound graph underlying the $\Omega(n^{1/3}/\log n)$ lower bound [19, 33] for triangle enumeration is the Erdős-Rényi random graph $\mathcal{G}(n, p)$ with $p = 1/2$. Hence it does not rule out the possibility of an $n^{(1/3)-\Omega(1)}$-round CONGEST algorithm for the enumeration problem on sparse graphs (i.e. $m = o(n^2)$) or the detection problem. It remains an open problem to find the asymptotically optimal round complexity of these problems in CONGEST. For the case of CONGESTED-CLIQUE, efficient algorithms for these problems are already known: triangle detection can be solved in $\tilde{O}(n^{1-(2/\omega)+o(1)}) = o(n^{0.158})$ time [5], triangle enumeration on m-edge graphs can be solved in $\max\{O(m/n^{5/3}), O(1)\}$ time [6, 33].

We would also like to further investigate the power of the distributed expander decomposition. Can this tool be applied to other distributed problems than triangle detection and enumeration? It has been known that this technique can be applied to give a sublinear-time distributed algorithm for *exact* minimum cut [11]. We expect to see more applications of distributed expander decomposition in the future.

ACKNOWLEDGMENTS

We thank Seth Pettie for very useful discussion.

REFERENCES

[1] Amir Abboud, Keren Censor-Hillel, Seri Khoury, and Christoph Lenzen. 2017. Fooling views: A new lower bound technique for distributed computations under congestion. *arXiv preprint arXiv:1711.01623* (2017).

[2] Sanjeev Arora, Boaz Barak, and David Steurer. 2015. Subexponential Algorithms for Unique Games and Related Problems. *J. ACM* 62, 5, Article 42 (Nov. 2015), 25 pages.

[3] Sanjeev Arora, Satish Rao, and Umesh Vazirani. 2009. Expander Flows, Geometric Embeddings and Graph Partitioning. *J. ACM* 56, 2, Article 5 (April 2009), 37 pages. https://doi.org/10.1145/1502793.1502794

[4] Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. 1994. Low-diameter graph decomposition is in NC. *Random Structures & Algorithms* 5, 3 (1994), 441–452.

[5] Keren Censor-Hillel, Petteri Kaski, Janne H. Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. 2016. Algebraic methods in the congested clique. *Distributed Computing* (2016).

[6] Keren Censor-Hillel, Dean Leitersdorf, and Elia Turner. 2018. Sparse Matrix Multiplication and Triangle Listing in the Congested Clique Model. In *22nd International Conference on Principles of Distributed Systems (OPODIS 2018) (Leibniz International Proceedings in Informatics (LIPIcs))*, Jiannong Cao, Faith Ellen, Luis Rodrigues, and Bernardo Ferreira (Eds.), Vol. 125. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 4:1–4:17. https://doi.org/10.4230/LIPIcs.OPODIS.2018.4

[7] Yi-Jun Chang and Thatchaphol Saranurak. 2019. Improved Distributed Expander Decomposition and Nearly Optimal Triangle Enumeration. *CoRR* abs/1904.08037 (2019). arXiv:1904.08037 http://arxiv.org/abs/1904.08037

[8] Yi-Jun. Chang, Seth Pettie, and Hengjie Zhang. 2019. Distributed Triangle Detection via Expander Decomposition. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 821–840.

[9] Timothy Chu, Yu Gao, Richard Peng, Sushant Sachdeva, Saurabh Sawlani, and Junxing Wang. 2018. Graph Sparsification, Spectral Sketches, and Faster Resistance Computation, via Short Cycle Decompositions. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*. 361–372. https://doi.org/10.1109/FOCS.2018.00042

[10] Michael B. Cohen, Jonathan A. Kelner, John Peebles, Richard Peng, Anup B. Rao, Aaron Sidford, and Adrian Vladu. 2017. Almost-linear-time algorithms for Markov chains and new spectral primitives for directed graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. 410–419. https://doi.org/10.1145/3055399.3055463

[11] Mohit Daga, Monika Henzinger, Danupon Nanongkai, and Thatchaphol Saranurak. 2019. Distributed Edge Connectivity in Sublinear Time. *arXiv preprint arXiv:1904.04341* (2019). To appear at STOC'19.

[12] Danny Dolev, Christoph Lenzen, and Shir Peled. 2012. "Tri, Tri Again": Finding Triangles and Small Subgraphs in a Distributed Setting. In *Proceedings 26th International Symposium on Distributed Computing (DISC)*. 195–209.

[13] Andrew Drucker, Fabian Kuhn, and Rotem Oshman. 2014. On the power of the congested clique model. In *Proceedings 33rd ACM Symposium on Principles of Distributed Computing (PODC)*. 367–376.

[14] Orr Fischer, Tzlil Gonen, Fabian Kuhn, and Rotem Oshman. 2018. Possibilities and Impossibilities for Distributed Subgraph Detection. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures (SPAA)*. ACM, New York, NY, USA, 153–162.

[15] Mohsen Ghaffari, Fabian Kuhn, and Hsin-Hao Su. 2017. Distributed MST and Routing in Almost Mixing Time. In *Proceedings 37th ACM Symposium on Principles of Distributed Computing (PODC)*. 131–140.

[16] Mohsen Ghaffari and Jason Li. 2018. New Distributed Algorithms in Almost Mixing Time via Transformations from Parallel Algorithms. In *Proceedings 32nd International Symposium on Distributed Computing (DISC) (Leibniz International Proceedings in Informatics (LIPIcs))*, Ulrich Schmid and Josef Widder (Eds.), Vol. 121. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 31:1–31:16.

[17] Mohsen Ghaffari and Krzysztof Nowicki. 2018. Congested Clique Algorithms for the Minimum Cut Problem. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing (PODC '18)*. ACM, New York, NY, USA, 357–366. https://doi.org/10.1145/3212734.3212750

[18] Oded Goldreich and Dana Ron. 1999. A Sublinear Bipartiteness Tester for Bounded Degree Graphs. *Combinatorica* 19, 3 (01 Mar 1999), 335–373.

[19] Taisuke Izumi and François Le Gall. 2017. Triangle Finding and Listing in CONGEST Networks. In *Proceedings 37th ACM Symposium on Principles of Distributed Computing (PODC)*. 381–389. https://doi.org/10.1145/3087801.3087811

[20] Mark Jerrum and Alistair Sinclair. 1989. Approximating the Permanent. *SIAM J. Comput.* 18, 6 (1989), 1149–1178.

[21] Tomasz Jurdziński and Krzysztof Nowicki. 2018. MST in $O(1)$ Rounds of Congested Clique. In *Proceedings 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2620–2632.

[22] Ravi Kannan, Santosh Vempala, and Adrian Vetta. 2004. On Clusterings: Good, Bad and Spectral. *J. ACM* 51, 3 (May 2004), 497–515. https://doi.org/10.1145/990308.990313

[23] Ken-Ichi Kawarabayashi and Mikkel Thorup. 2018. Deterministic Edge Connectivity in Near-Linear Time. *J. ACM* 66, 1, Article 4 (Dec. 2018), 4:1–4:50 pages.

[24] Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. 2014. An Almost-Linear-Time Algorithm for Approximate Max Flow in Undirected Graphs, and its Multicommodity Generalizations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*. 217–226. https://doi.org/10.1137/1.9781611973402.16

[25] Fabian Kuhn and Anisur Rahaman Molla. 2015. Distributed Sparse Cut Approximation. In *Proceedings 19th International Conference on Principles of Distributed Systems (OPODIS)*. 10:1–10:14.

[26] François Le Gall. 2014. Powers of Tensors and Fast Matrix Multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC '14)*. ACM, New York, NY, USA, 296–303. https://doi.org/10.1145/2608628.2608664

[27] Nathan Linial and Michael Saks. 1993. Low diameter graph decompositions. *Combinatorica* 13, 4 (01 Dec 1993), 441–454.

[28] Gary L. Miller, Richard Peng, and Shen Chen Xu. 2013. Parallel graph decompositions using random shifts. In *Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures (SPAA)*. ACM, 196–203.

[29] Danupon Nanongkai and Thatchaphol Saranurak. 2017. Dynamic spanning forest with worst-case update time: adaptive, Las Vegas, and $O(n^{1/2-\epsilon})$-time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. 1122–1129. https://doi.org/10.1145/3055399.3055447

[30] Danupon Nanongkai, Thatchaphol Saranurak, and Christian Wulff-Nilsen. 2017. Dynamic minimum spanning forest with subpolynomial worst-case update time. In *Proceedings of IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 950–961.

[31] Lorenzo Orecchia and Nisheeth K. Vishnoi. 2011. Towards an SDP-based Approach to Spectral Methods: A Nearly-Linear-Time Algorithm for Graph Partitioning and Decomposition. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*. 532–545. https://doi.org/10.1137/1.9781611973082.42

[32] Lorenzo Orecchia and Zeyuan Allen Zhu. 2014. Flow-based Algorithms for Local Graph Clustering. In *Proceedings of the Twenty-fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '14)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1267–1286. http://dl.acm.org/citation.cfm?id=2634074.2634168

[33] Gopal Pandurangan, Peter Robinson, and Michele Scquizzato. 2018. On the Distributed Complexity of Large-Scale Graph Computations. In *Proceedings 30th ACM Symposium on Parallelism in Algorithms and Architecture (SPAA)*.

[34] M. Pătraşcu and M. Thorup. 2007. Planning for Fast Connectivity Updates. In *Proceedings 48th IEEE Symposium on Foundations of Computer Science (FOCS)*. 263–271.

[35] Sriram V. Pemmaraju. 2001. Equitable Coloring Extends Chernoff-Hoeffding Bounds. In *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, Michel Goemans, Klaus Jansen, José D. P. Rolim, and Luca Trevisan (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 285–296.

[36] Prasad Raghavendra and David Steurer. 2010. Graph Expansion and the Unique Games Conjecture. In *Proceedings 42nd ACM Symposium on Theory of Computing (STOC)*. 755–764.

[37] Thatchaphol Saranurak and Di Wang. 2019. Expander Decomposition and Pruning: Faster, Stronger, and Simpler. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2616–2635.

[38] Atish Das Sarma, Anisur Rahaman Molla, and Gopal Pandurangan. 2015. Distributed Computation of Sparse Cuts via Random Walks. In *Proceedings 16th International Conference on Distributed Computing and Networking (ICDCN)*. 6:1–6:10.

[39] Daniel A. Spielman and Nikhil Srivastava. 2008. Graph sparsification by effective resistances. In *Proceedings 40th ACM Symposium on Theory of Computing (STOC)*. 563–568.

[40] Daniel A. Spielman and Shang-Hua Teng. 2011. Spectral Sparsification of Graphs. *SIAM J. Comput.* 40, 4 (2011), 981–1025. https://doi.org/10.1137/08074489X

[41] Daniel A. Spielman and Shang-Hua Teng. 2013. A Local Clustering Algorithm for Massive Graphs and Its Application to Nearly Linear Time Graph Partitioning. *SIAM J. Comput.* 42, 1 (2013), 1–26.

[42] Daniel A. Spielman and Shang-Hua Teng. 2014. Nearly Linear Time Algorithms for Preconditioning and Solving Symmetric, Diagonally Dominant Linear Systems. *SIAM J. Matrix Anal. Appl.* 35, 3 (2014), 835–885.

[43] Luca Trevisan. 2008. Approximation Algorithms for Unique Games. *Theory of Computing* 4, 5 (2008), 111–128.

[44] Christian Wulff-Nilsen. 2017. Fully-dynamic minimum spanning forest with improved worst-case update time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. 1130–1143. https://doi.org/10.1145/3055399.3055415