On the Streaming Complexity of Expander Decomposition

Yu Chen
EPFL, Lausanne, Switzerland

Michael Kapralov
EPFL, Lausanne, Switzerland

Mikhail Makarov
EPFL, Lausanne, Switzerland

Davide Mazzali
EPFL, Lausanne, Switzerland

Abstract

In this paper we study the problem of finding (ϵ, ϕ)-expander decompositions of a graph in the streaming model, in particular for dynamic streams of edge insertions and deletions. The goal is to partition the vertex set so that every component induces a ϕ-expander, while the number of inter-cluster edges is only an ϵ fraction of the total volume. It was recently shown that there exists a simple algorithm to construct a $(O(\phi \log n), \phi)$-expander decomposition of an n-vertex graph using $\widetilde{O}(n/\phi^2)$ bits of space [Filtser, Kapralov, Makarov, ITCS'23]. This result calls for understanding the extent to which a dependence in space on the sparsity parameter ϕ is inherent. We move towards answering this question on two fronts.

We prove that a $(O(\phi \log n), \phi)$-expander decomposition can be found using $\widetilde{O}(n)$ space, for every ϕ. At the core of our result is the first streaming algorithm for computing boundary-linked expander decompositions, a recently introduced strengthening of the classical notion [Goranci et al., SODA'21]. The key advantage is that a classical sparsifier [Fung et al., STOC'11], with size independent of ϕ, preserves the cuts inside the clusters of a boundary-linked expander decomposition within a multiplicative error.

Notable algorithmic applications use sequences of expander decompositions, in particular one often repeatedly computes a decomposition of the subgraph induced by the inter-cluster edges (e.g., the seminal work of Spielman and Teng on spectral sparsifiers [Spielman, Teng, SIAM Journal of Computing 40(4)], or the recent maximum flow breakthrough [Chen et al., FOCS'22], among others). We prove that any streaming algorithm that computes a sequence of $(O(\phi \log n), \phi)$-expander decompositions requires $\widetilde{\Omega}(n/\phi)$ bits of space, even in insertion only streams.

2012 ACM Subject Classification Theory of computation → Streaming models; Theory of computation → Sparsification and spanners; Theory of computation → Sketching and sampling; Theory of computation → Lower bounds and information complexity

Keywords and phrases Graph Sketching, Dynamic Streaming, Expander Decomposition

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.46

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2404.16701` [12]

1 Introduction

Expander graphs are known to represent a class of easier instances for many problems. Therefore, breaking down the input into disjoint expanders can allow to conveniently solve the task on each of them separately, before combining the partial results into a global solution. This approach is enabled by (ϵ, ϕ)-expander decompositions (for short, (ϵ, ϕ)-

ED). For an undirected graph $G = (V, E)$, this is a partition $\mathcal{U}$ of the vertex set V, such that there are at most $\epsilon|E|$ inter-cluster edges, while every cluster $U \in \mathcal{U}$ induces a ϕ-expander. The list of successful applications of this framework is long, including Laplacian system solvers [31], deterministic algorithms for minimum cut [28], graph and hyper-graph sparsification [2, 14, 23, 32], dynamic algorithms for cut and connectivity problems [18, 19], fast max flow algorithms [11], distributed triangle enumeration [9], polynomial time algorithms for semirandom planted CSPs [20], and many more.

We refer to ϵ and ϕ as the *sparsity* parameters: the former controls how sparsely connected the clusters need to be, and the latter determines how expanding (i.e. non-sparse) the cuts within clusters are. The reason for using the same term for both is that they are in fact very closely related. One can show that any n-vertex graph has an (ϵ, ϕ)-ED with $\epsilon = O(\phi \log n)$. To see this, consider the following constructive argument: if the graph has no ϕ-sparse cut then $\{V\}$ is a valid ED of G, otherwise recurse on the two sides $(S, V \setminus S)$ of a ϕ-sparse cut and union the results to get an ED for G. Every cluster in this decomposition is then an expander, and a charging argument allows to bound the number of inter-cluster edges by $O(\phi|E| \log n)$ [30]. One can also observe that no better asymptotic trade-off between ϵ and ϕ is possible in general [6]. Therefore, this sets the benchmark for algorithmic constructions of EDs.

At a high level, many ED algorithms follow the approach suggested by the existential argument. As a naive implementation would take exponential time, the crux often lies in efficient algorithms that either certify that a large portion of the input is an expander, or find a balanced sparse cut. This would result in a small depth recursion, thanks to balancedness, where each level requires little computational resources. There are several successful examples of this approach. In the sequential setting, a recent algorithm constructs a $(O(\phi \log^3 n), \phi)$-ED in $\widetilde{O}(|E|)$ time [29], based on the previous best algorithm which runs in $\widetilde{O}(|E|/\phi)$ time [30]. There is also a deterministic counterpart, which outputs a $(\phi \cdot n^{o(1)}, \phi)$-ED in almost linear time [15]. For the CONGEST model of distributed computing, it is possible to obtain, for instance, a $(\phi^{1/\sqrt{\log n}} \cdot n^{o(1)}, \phi)$-ED in $n^{o(1)}/\phi$ rounds [9]. It is also possible to maintain a $(\phi \cdot n^{o(1)}, \phi)$-ED for a graph undergoing edge updates in $n^{o(1)}/\phi^2$ amortized update time [19, 21].

1.1 Previous Work

In the streaming setting, the problem of finding EDs was open until the recent work of [16]. They obtain a dynamic stream algorithm which outputs a $(O(\phi \log n), \phi)$-ED and takes $\widetilde{O}(n/\phi^2)$ space. While being optimal in the quality of the decomposition, decoding the sketch to actually output an ED takes exponential time. The authors also give a polynomial time version: one can produce a $(\phi \cdot n^{o(1)}, \phi)$-ED using space $\widetilde{O}(n/\phi^2) + n^{1+o(1)}/\phi^{1-o(1)}$, with a post-processing that takes poly(n) time (where the $o(1)$'s can be tuned, allowing for a quality-space trade-off).

These streaming algorithms also adopt the recursive approach based on finding balanced ϕ-sparse cuts, but the streaming model poses a challenge that we now illustrate. Sparsification for graph streams has been extensively studied [3, 4, 5, 24, 25], so a natural attempt would consist of maintaining a cut sparsifier as the stream comes and later run the recursive partitioning on it. However, it must be noted that these cuts are to be found in the subgraphs induced by the two sides of a previously made cut. This is not a problem in a classical computational setting, but it actually constitutes the main obstacle for the streaming model: at sketching time, the algorithm does not know which subgraphs it will need to access. Unfortunately, it is impossible to preserve cut sizes in arbitrary subgraphs with multiplicative

precision. The natural work-around is to introduce an additive error term: in [16], the authors introduce the concept of power-cut sparsifiers. For any given partition $\mathcal{U}$ of V, these sparsifiers preserve the cuts $|E(S, U \setminus S)|$ of each cluster $U \in \mathcal{U}$ in the partition up to an error of $\delta \cdot |E(S, U \setminus S)| + \psi \cdot \mathrm{vol}(S)$. They further show that maintaining $\widetilde{O}(n/\delta\psi)$ random linear measurements of the incidence matrix of the input graph is enough to obtain such sparsifiers. One can see that setting $\delta \ll 1$ and $\psi \approx \phi$ preserves the sparsity of cuts to within an additive error of roughly ϕ, using $\widetilde{O}(n/\phi)$ linear measurements. This is enough to find a balanced ϕ-sparse cut in every subgraph induced by a given partition of the vertex set. There is another caveat, though: power-cut sparsifiers give a high probability guarantee for any fixed partition, but we cannot expect them to work for all partitions simultaneously. Therefore, in order not to use the sparsifiers adaptively, we need one power-cut sparsifier for every recursion level. The authors show that the depth of the procedure cannot exceed $\widetilde{O}(1/\epsilon) = \widetilde{O}(1/\phi)$, thus obtaining the space complexity stated before. A few more details are involved in the polynomial time algorithm, but the underlying framework is the same.

1.2 Our Contribution

The work of [16] initiated the study of expander decompositions in the streaming setting, and consequently raised the question of whether a dependence in space on the sparsity parameter ϕ is inherent. In this paper, we move towards settling the streaming complexity of expander decompositions by attacking the problem on two fronts: (1) we give a nearly optimal algorithm for "one-level" expander decomposition that avoids the sparsity dependence, and (2) we show that computing a "repeated" expander decomposition, commonly used in applications, cannot avoid such dependence.

Upper Bound

We give an $\widetilde{O}(n)$ space algorithm for computing a $(O(\phi \log n), \phi)$-ED in dynamic streams. Specifically, we show that a "universal" sketch consisting of $\widetilde{O}(n)$ random linear measurements of the incidence matrix can be decoded into a $(O(\phi \log n), \phi)$-ED for any ϕ: the sketch is independent of the sparsity ϕ.

▶ **Theorem 1** (ED algorithm – exponential time decoding). *Let $G = (V, E)$ be a graph given in a dynamic stream. Then, there is an algorithm that maintains a linear sketch of G in $\widetilde{O}(n)$ space. For any $\phi \in (0, 1)$, the algorithm decodes the sketch to compute a $(O(\phi \log n), \phi)$-ED of G with high probability, in $\widetilde{O}(n)$ space and $2^{O(n)}$ time.*

We note that at least $\Omega(n \log n)$ space is needed for any small enough ϕ: for example, if the input graph is a matching of size $n/10$, say, its ED gives a $1 - O(\phi \log n)$ fraction of the matching edges.

The decoding time of our sketch can be made polynomial, at the expense of some loss in the quality of the expander decomposition (similarly to [16]), but keeping the space independent of the sparsity ϕ.

▶ **Theorem 2** (ED algorithm – polynomial time decoding). *Let $G = (V, E)$ be a graph given in a dynamic stream. Then, there is an algorithm that maintains a linear sketch of G in $n^{1+o(1)}$ space. For any $\phi \in (0, 1)$, the algorithm decodes the sketch to compute a $(\phi \cdot n^{o(1)}, \phi)$-ED of G with high probability, in $n^{1+o(1)}$ space and $\mathrm{poly}(n)$ time.*

In this case we are off by subpolynomial factors in both quality and space complexity as compared to the optimal ones. The actual theorem that we prove allows one to trade-off the loss in quality and increase in space.

Lower Bound

Most algorithmic applications of EDs, including the ones mentioned above, do not use just one ED of the input graph. Rather, they use an ED sequence obtained by repeatedly computing an ED of the inter-cluster edges from the previous level. This can be done in two natural ways: by contracting the clusters of an ED (we call this variant CED, for "contraction"), or by removing the intra-cluster edges without changing the vertex set (we call this variant RED, for "removal", see Section 1.3). These approaches lead to different results and are used for different applications (e.g., [19, 28] contract the clusters, and [32, 11] recurse on inter-cluster edges). In the sequential setting, both CEDs and REDs can be obtained straightforwardly given an ED algorithm. However, this is not so obvious in the streaming model.

One the one hand, one should be able get a sparsity-independent algorithm for computing CEDs in dynamic streams via Theorem 1 or Theorem 2: observe that contracting the vertices of a sparsifier of the input graph G gives a sparsifier of the graph obtained by contracting vertices in G, so the idea would be to maintain an independent copy of our algorithm for each of the $O(\log n)$ levels and contracting vertices in the sparsifier based on the decomposition of the previous level. On the other hand, we show a space lower bound for computing REDs, even in insertion only streams, showing that a dependence on $1/\phi$ is necessary.

▶ **Theorem 3** (RED lower bound). *Let $\epsilon, \phi \in (0,1)$ such that $1/n \ll \phi \ll 1/\operatorname{polylog} n$ and $\epsilon = \widetilde{O}(\phi)$. Any streaming algorithm that with constant probability computes at least two levels of an (ϵ, ϕ)-RED requires $\widetilde{\Omega}(n/\phi)$ bits of space.*

The result seems to challenge the intuition that EDs become weaker as ϕ and ϵ decrease (note that when ϕ is, say, $1/n^2$, an ED can simply consist of connected components). The questions of (1) whether this bound can be improved, (2) how it scales with the number of levels of RED we compute, and (3) whether there are algorithms matching such bounds, remain open.

1.3 Preliminaries

Graph Streaming

In this paper, we will be mostly working in the dynamic graph streaming model, where we know the vertex set $V = [n]$, and we receive a stream of insertions and deletions for undirected edges over V. In insertion-only graph streams, the only difference is that previously inserted edges cannot be deleted. At the end of the stream, the graph $G = (V, E)$ consists of the edges that have been inserted and not deleted, and we say that G is given in a (dynamic) stream. When we consider a graph given in a (dynamic) stream, we are implicitly assuming it to have n vertices, without introducing the parameter n explicitly. Also, when we refer to G and n without reintroducing them we are implicitly considering the graph resulting from the input stream and its number of vertices.

A powerful tool for dynamic graph streams is linear sketching, introduced in the seminal work of Ahn, Guha, and McGregor [4]. The idea is to left-multiply the $\binom{n}{2} \times n$ incidence matrix of the graph by a random $k \times \binom{n}{2}$ matrix for $k \ll \binom{n}{2}$. Since the sketch consists of linear measurements, it automatically handles the case of dynamic streams. We will not be using sketching techniques directly, but rather employ existing algorithms that do. In this paper we restrain ourselves to unweighted edge streams. One may study the same problem in general turnstile graph streams [13], but we do not do this here.

Cuts, Volumes, and Expanders

Given an unweighted graph $G = (V, E)$, possibly with multiple self-loops on the vertices, we will operate with weighted graphs that approximate G in an appropriate sense. We write $G' = (V', E', w)$ to denote a weighted graph, possibly with multiple weighted self-loops on the vertices. We describe next some notation for such weighted graph G'. The same notation carries over to the unweighted graph G by implicitly setting w to assign a weight of 1 to all edges and self-loops.

For any $A, B \subseteq V'$ we denote by $E'(A, B)$ the edges in E' with one endpoint in A and one in B, and by $w(A, B)$ the total weight of edges in $E'(A, B)$. The volume of a cut $S \subseteq V'$ is the sum of the (weighted) degrees, including self-loops, of its vertices. We denote it by $\mathrm{vol}_{G'}(S)$. The sparsity of a cut $\emptyset \neq S \subsetneq V'$ is defined as

$$\Phi_{G'}(S) = \frac{w(S, V' \setminus S)}{\min\{\mathrm{vol}_{G'}(S), \mathrm{vol}_{G'}(V' \setminus S)\}} .$$

For $\psi \in (0, 1)$, we make a distinction between cuts having sparsity less than ψ, which we call ψ-sparse, and cuts having sparsity at least ψ, which we call ψ-expanding. The sparsity of G' is then defined as

$$\Phi_{G'} = \min_{\emptyset \neq S \subsetneq V'} \Phi_{G'}(S) ,$$

and we call G' a ψ-expander if all its cuts are ψ-expanding, i.e. $\Phi_{G'} \geq \psi$.

Expander Decomposition

As we will treat expander decompositions for the input graph G only, we conveniently use the following additional notation. For a cluster $U \subseteq V$, i.e. a subset of the vertices, and a cut $S \subseteq U$, which is also a subset of the vertices, we denote the number of edges crossing S in U by $\partial_U S$, i.e. $\partial_U S = |E(S, U \setminus S)|$. This is the *local* cut of S in U. If $U = V$, we use a shorthand notation $\partial S = \partial_V S$. We call such a cut the *global* cut of S, since for $S \subseteq U$ we will be interested in both $\partial_U S$ and ∂S. Moreover, we drop the subscript from the volume and simply write $\mathrm{vol}(\cdot)$ instead of $\mathrm{vol}_G(\cdot)$. Then, EDs can be defined as follow.

▶ **Definition 4** (Expander decomposition). *Let $G = (V, E)$ and let $\epsilon, \phi \in (0, 1)$. A partition $\mathcal{U}$ of V is an (ϵ, ϕ)-expander decomposition (for short, (ϵ, ϕ)-ED) of G if*

1. $\frac{1}{2} \sum_{U \in \mathcal{U}} \partial U \leq \epsilon |E|$, *and*
2. *for every $U \in \mathcal{U}$, one has that $G[U]$ is a ϕ-expander.*

Boundary-Linked Expander Decomposition

A boundary-linked ED [19] is the same as a classical ED except that Property 2 of Definition 4 is strengthened. For a cut S in a cluster U, the boundary $\mathrm{bnd}_U(S)$ of S with respect to U is the number of edges that go from S to the outside of U, i.e. $\mathrm{bnd}_U(S) = |E(S, V \setminus U)|$. For $U \subseteq V$ and $\tau \geq 0$, the τ-boundary linked subgraph of G on U, denoted by $G[U]^\tau$, is the subgraph of G induced by U with additional $\tau \cdot \mathrm{bnd}_U(\{u\})$ self-loops attached to every $u \in U$. Then a boundary-linked ED can be defined as follows.

▶ **Definition 5** (Boundary-linked expander decomposition [19]). *Let $G = (V, E)$, let $b, \epsilon, \phi \in (0, 1)$ be parameters such that $b \geq \phi$, and let $\gamma \geq 1$ be an error parameter. A partition $\mathcal{U}$ of V is a $(b, \epsilon, \phi, \gamma)$-boundary-linked expander decomposition (for short, $(b, \epsilon, \phi, \gamma)$-BLD) of G if*

1. $\frac{1}{2} \sum_{U \in \mathcal{U}} \partial U \leq \epsilon |E|$, *and*
2. *for every $U \in \mathcal{U}$, $G[U]^{b/\phi}$ is a ϕ/γ-expander.*

Algorithm 1 DECOMPOSE: recursive procedure for computing a $(O(\phi \log n), \phi)$-ED of G.

```
// G = (V, E) is the input graph
// φ ∈ (0, 1) is the sparsity parameter
procedure DECOMPOSE(G):
    if G is a φ-expander then
        return {V}
    else
        S ← a φ-sparse cut of G
        return DECOMPOSE(G[S]) ∪ DECOMPOSE(G[V \ S])
    end
```

Expander Decomposition Sequence

For a partition $\mathcal{U}$ of V (think of $\mathcal{U}$ as an ED of G), we denote by $E \setminus \mathcal{U}$ the set of inter-cluster (or crossing) edges with respect to $\mathcal{U}$, i.e.

$$E \setminus \mathcal{U} = E \setminus \bigcup_{U \in \mathcal{U}} \binom{U}{2}.$$

Analogously, we let $G \setminus \mathcal{U} = (V, E \setminus \mathcal{U})$ be the subgraph of G obtained by removing the intra-cluster edges in $\mathcal{U}$. For a sequence of partitions $\mathcal{U}_1, \ldots, \mathcal{U}_\ell$ of V and $i \in [\ell]$, we define $G_1^{\mathrm{R}} = G$ and denote by $G_{i+1}^{\mathrm{R}} = G_i^{\mathrm{R}} \setminus \mathcal{U}_i$ the subgraph of G obtained by removing the intra-cluster edges of the first i partitions.

▶ **Definition 6** (Removal-based ED sequence). *Let $G = (V, E)$, let $\epsilon, \phi \in (0, 1)$, let $\ell \geq 1$, and let $\mathcal{U}_1, \ldots, \mathcal{U}_\ell$ be a sequence of partitions of V. The sequence $\mathcal{U}_1, \ldots, \mathcal{U}_\ell$ is an ℓ-level removal-based (ϵ, ϕ)-ED sequence (for short, ℓ-level (ϵ, ϕ)-RED or (ϵ, ϕ, ℓ)-RED) of G if, for all $i \in [\ell]$, $\mathcal{U}_i$ is an (ϵ, ϕ)-ED of the graph G_i^{R}.*

2 Sparsity-Independent One-Level Expander Decomposition

Given a graph $G = (V, E)$ in a dynamic stream, and a parameter $\phi \in (0, 1)$, we consider the problem of computing an (ϵ, ϕ)-ED of G for $\epsilon = O(\phi \log n)$. We show that one can do this in $\widetilde{O}(n)$ bits of space, without any dependence on ϕ. In this section, we sketch our approach.

Let us begin by recalling the standard recursive framework for constructing expander decompositions [22, 32, 33], concisely summarized in Algorithm 1. For any parameter $\phi \leq 10^{-1}/\log n$, this algorithm produces a $(O(\phi \log n), \phi)$-ED by recursively partitioning G along ϕ-sparse cuts until no more such cuts are found.

Many algorithmic constructions of EDs are essentially efficient implementations of Algorithm 1. Adapting Algorithm 1 to dynamic streams comes with its own set of challenges. When the input is given in a dynamic stream, one can only afford to store a limited amount of information about the input graph. Since Algorithm 1 only needs to measure the sparsity of cuts, it seems enough to have access to cut sizes and volumes. Both these quantities are preserved by cut sparsifiers. According to the classical definition [7], a δ-cut sparsifier is a weighted subgraph $H = (V, E', w)$ of the input $G = (V, E)$, where for every cut $S \subseteq V$ one has

$$(1 - \delta) \cdot \partial S \leq \partial^w S \leq (1 + \delta) \cdot \partial S.$$

It is known that such a sparsifier can be constructed in dynamic streams using $\widetilde{O}(n/\delta^2)$ bits of space [5]. With the same space requirement we can also measure the sparsity of cuts up to a $(1 \pm \delta)$ multiplicative error. Having this in mind, it is natural to consider the following algorithm: first, read the stream and construct a cut sparsifier, then run Algorithm 1 on it and output the resulting clustering as the expander decomposition. However, this approach does not work as is. As it turns out, more information is needed about the graph than what is captured by the regular notion of a cut sparsifier.

The problem with this approach becomes immediately apparent when one considers the second recursion level. As the process recurses on the two sides of a sparse cut S, it will repeat the procedure on the subgraphs $G[S]$ and $G[V \setminus S]$. Unfortunately, the cut preservation property of the sparsifier does not carry over to those subgraphs, making it inadequate for estimating the sparsity of the cuts in those graphs. In fact, the notion of expander decomposition itself already operates on the subgraphs, as it is required that each cut in each cluster of the decomposition is expanding.

2.1 Testing Expansion of Subgraphs

The reasoning from above gives rise to the following sketching problem: produce a sparsifier that can be used to check that any subgraph is a ϕ-expander. To solve it, the authors of [16] introduced the concept of a (δ, ψ)-power-cut sparsifier. Its property is that, for any cluster $U \subseteq V$, with high probability all cuts $S \subseteq U$ are preserved within an additive-multiplicative error. Recall that $\partial_U S$ is equal to the size of the cut S inside the induced subgraph $G[U]$. When talking about sparsifiers in this section, we will slightly abuse notation by assuming that there is some instance $H = (V, E', w)$ of it and denote by $\partial_U^w S$ the size of the cut S in the subgraph $H[U]$ of this sparsifier H. Then we can write the guarantee of a (δ, ψ)-power-cut sparsifier as follows[1]:

$$\forall\, S \subseteq U, \quad (1-\delta) \cdot \partial_U S - \psi \cdot \mathrm{vol}(S) \leq \partial_U^w S \leq (1+\delta) \cdot \partial_U S + \psi \cdot \mathrm{vol}(S)\,.$$

The authors also give a dynamic stream construction, which uses $\widetilde{O}(n/\delta\psi)$ bits of space by sampling edges proportionally to the degrees of their endpoints.

To check the ϕ-expansion of subgraphs up to a small constant multiplicative error, it is enough to set δ to be a small constant. However, the multiplicative error parameter ψ must be $\lesssim \phi$. This is a significant downside of this construction, as it was shown in [16] that a (δ, ϕ)-power-cut sparsifier must have at least $\Omega(n/\phi)$ edges. In fact, their lower bound is more general, and holds for general subgraphs[2]. In other words, $\Omega(n/\phi)$ space is necessary in order to test ϕ-expansion of general subgraphs. Consequently, new tools must be used to have any hope of getting an algorithm independent of $1/\phi$.

Our Contribution: Sparsification of Boundary-Linked Subgraphs

As it turns out, solving the expansion testing problem, as it was stated, is not necessary. Recently, in the breakthrough work of [19], it was shown that one could demand additional properties from the expander decomposition, and it will still exist at the price of increasing the number of inter-cluster edges by a small multiplicative factor.

[1] A similar sparsifier construction was proposed by [1]. Their construction has an additive error of $\psi|S|$, so the dependence is on the number of vertices instead of the volume.

[2] The lower bound instance is a $1/\phi$-regular graph. There, each edge by itself forms a ϕ-expander, while any pair of vertices without an edge between them is not. Being able to test the ϕ-expansion of the majority of those small subgraphs would imply being able to recover the majority of edges in the graph.

More formally, let $U \subseteq V$ and $\tau > 0$. Note that if the τ-boundary-linked subgraph $G[U]^\tau$ (see Section 1.3 for the definition) is a ϕ-expander, then so necessarily is $G[U]$, but not the other way around. Then it is possible, for a given ϕ, and $\epsilon = \widetilde{O}(\phi)$, $\tau \approx 1/\phi$, to construct an (ϵ, ϕ)-expander decomposition $\mathcal{U}$ where for each cluster $U \in \mathcal{U}$, $G[U]^\tau$ is a $\widetilde{\Omega}(\phi)$-expander [19]. Such a decomposition is called a boundary-linked expander decomposition.

A key observation is that if in our algorithm we were aiming to construct a boundary-linked expander decomposition, the testing problem would only involve checking that any given boundary-linked subgraph is ϕ-expanding. Indeed, this problem is much easier than the original one and can be solved in space $\widetilde{O}(n)$. We will show how to do it in two steps: first, we will discuss how to strengthen the power-cut sparsifier, and then prove that this strengthening is enough to resolve the problem.

Achieving Additive Error in the Global Cut

As was noted in [16], the idea behind constructing a power-cut sparsifier was to reanalyse the guarantee given by the construction of [32] for sparsifying expanders. In other words, a power-cut sparsifier results from a more rigorous analysis of an existing sparsifier. The problem with this approach is that the sparsifier of [32] is relatively weak to begin with: it only preserves cuts in expanders, while other constructions can preserve them in all graphs [7, 17]. To strengthen the guarantee, one can give the same treatment to the construction of [17] (see the full version of this paper). For the sake of simplicity and to gain an intuition for why this kind of sparsification is at all possible, we discuss here how to do that with the classical construction of [27] by closely following the original proof.

We show that, given a graph $G = (V, E)$ with a minimum cut of size k, it is possible to construct a sparsifier H of G such that every cut inside any given subgraph $G[U]$ of G is preserved with high probability with the following guarantee:

$$\forall\, S \subseteq U, \quad \partial_U S - \delta \cdot \partial S \leq \partial_U^w S \leq \partial_U S + \delta \cdot \partial S\,. \tag{1}$$

In this paper, a sparsifier with the property of Equation (1) is called a cluster sparsifier. To achieve the guarantee of Equation (1), consider using the same process as in [27]: sample each edge with the same probability $p \approx \delta^{-2}/k$.

To see why this works, fix a subgraph $U \subseteq V$, and consider any cut S in U. We wish to show that the size of the cut S inside U concentrates well after sampling. In the original proof, this is done by simply applying a Chernoff bound. In our case, this bound would look like this:

$$\Pr[|\partial_U^w S - \partial_U S| \geq \delta \cdot \partial_U S] \leq \exp\left(-\frac{1}{3}\delta^2 p \cdot \partial_U S\right).$$

However, this is insufficient, as the probability depends on the cut size inside $G[U]$. As we have no lower bound on its size, unlike with the sizes of global cuts, the second part of the argument of [27] cannot be applied. Instead, we apply an additive-multiplicative version of the Chernoff bound (see for example [16]), that allows us to compare the approximation error with a bigger value than its expectation. This gives us

$$\Pr[|\partial_U^w S - \partial_U S| \geq \delta \cdot \partial S] \leq 2\exp\left(-\frac{1}{100}\delta^2 p \cdot \partial S\right).$$

Expressing the probability in terms of the global cut allows us to use the cut counting lemma [26], which bounds the number of global cuts of size at most αk by $n^{2\alpha}$, for $\alpha \geq 1$. The proof is concluded by associating each global cut with a local cut in $G[U]$ and taking the union bound over them.

In order to get the guarantee of Equation (1) for all graphs, not only those with a minimum cut of size k, one can sample edges proportionally to the inverse of their edge connectivity (as in the work of [17]) as opposed to uniformly. Moreover, one can implement such sampling scheme in dynamic streams, using the approach of Ahn, Guha, and McGregor [5]. We defer the details of these reanalyses to the full version of this paper, where we show how to construct cluster sparsifiers with the property of Equation (1) in dynamic streams.

Benefits of Boundary-Linked Subgraphs

To see why the cluster sparsifier is enough to solve the boundary-linked ϕ-expansion testing problem, consider the following reasoning. Set the self-loop parameter τ equal to b/ϕ, for some $b \gg \phi$ and $b \ll 1$. Fix a cluster U, for which $G[U]^{b/\phi}$ is an $\widetilde{\Omega}(\phi)$-expander. The crucial fact is that the size of any cut inside $G[U]^{b/\phi}$ is lower bounded by its size in the global graph up to a small polylogarithmic factor in the following way:

$$\partial_U S \geq \widetilde{\Omega}(b) \cdot \partial S \,. \tag{2}$$

We will now explain the derivation of the above equation in detail. For a cut $\emptyset \neq S \subsetneq U$, recall that $\mathrm{bnd}_U(S)$ is the number of edges going from S to $V \setminus U$. Note that by the definition of $G[U]^{b/\phi}$, the volume of any cut S inside of it is equal to

$$\mathrm{vol}_{G[U]^{b/\phi}}(S) = \mathrm{vol}(S) + \left(\frac{b}{\phi} - 1\right) \mathrm{bnd}_U(S) \,.$$

First, because we assume that $G[U]^{b/\phi}$ is an $\widetilde{\Omega}(\phi)$-expander, we have

$$\partial_U S \geq \widetilde{\Omega}(\phi) \cdot \mathrm{vol}_{G[U]^{b/\phi}}(S) \,.$$

Then, applying the aforementioned formula for $\mathrm{vol}_{G[U]^{b/\phi}}(S)$, we have

$$\partial_U S \geq \widetilde{\Omega}(\phi)\, \mathrm{vol}(S) + \widetilde{\Omega}(\phi) \left(\frac{b}{\phi} - 1\right) \mathrm{bnd}_U(S) \,.$$

Since $b \gg \phi$ and dropping the first summand, the above simplifies to

$$\partial_U S \geq \widetilde{\Omega}(b)\, \mathrm{bnd}_U(S) \,.$$

Finally, applying $\partial_U S + \mathrm{bnd}_U(S) = \partial S$ and $b \ll 1$, we arrive back at Equation (2).

A consequence of Equation (2) is that setting $\delta \approx 1/b$ in the cluster sparsifier produces the following guarantee for $\widetilde{\Omega}(\phi)$-expander subgraphs $G[U]^{b/\phi}$:

$$\forall\, S \subseteq U, \quad \partial_U S - O(1) \cdot \partial_U S \leq \partial_U^w S \leq \partial_U S + O(1) \cdot \partial_U S \,,$$

where the $O(1)$ can be made arbitrarily small. In other words, we achieve a multiplicative approximation guarantee on subgraphs of interest, which is enough to solve the testing problem. Since b can be set to be $1/\operatorname{polylog} n$, the final sparsifier size does not depend on $1/\phi$. In this sense, such sparsifier can be thought of as a "universal" sketch of the graph that allows to solve the testing problem for all ϕ simultaneously.

Even though we now know how to solve the testing problem, an implementation of Algorithm 1 would need to actually find a sparse cut when the test fails (and in particular, it needs to find a *balanced* sparse cut, see the next section). In the full version of this paper, we show that cluster sparsifiers enable to solve this harder task too: consider an offline algorithm that either correctly determines a graph to be a boundary-linked expander or finds

a (balanced) sparse cut; one can prove that we can run such algorithm on a subgraph of the sparsifier as a black box and obtain essentially the same result as if the algorithm was run on the corresponding subgraph of the original graph. In this sense, cluster sparsifiers serve as small error proxies to the original graph for expander-vs-sparse-cut type of queries on vertex-induced subgraphs.

2.2 Low Depth Recursion

Another problem that arises when using the recursive approach is that the subgraph sparsification guarantee of both power-cut sparsifiers and cluster sparsifiers is only probabilistic, and it can be shown that it cannot be made deterministic [16]. This means that after finding a sparse cut S in a subgraph $G[U]^{b/\phi}$ of a sparsifier, we cannot claim that the same sparsifier would preserve the cuts inside the new graph $G[S]^{b/\phi}$ with high probability, as it is dependent on another cut that we have already found inside the sparsifier. This means we cannot use the same sparsifier for two different calls to Algorithm 1 inside the same execution path. However, we can share a sparsifier among all the calls at the same recursion level since these will operate on independent portions of G. Therefore, we want to have a separate sparsifier for every recursion level. This means that in order to minimize space requirements, it is crucial to have a small recursion depth.

To have small recursion depth, the algorithmic approach of [8, 9, 16] enforces the sparse cut S from line 5 of Algorithm 1 to be balanced, i.e. none of S and $V \setminus S$ is much larger than the other. In particular, they only recurse on the two sides of a cut if $\mathrm{vol}(S) \gtrsim \epsilon \,\mathrm{vol}(V \setminus S)$. When there is no balanced sparse cut and yet the input is not an expander, a lemma of Spielman and Teng [32] suggests there should be an $\Omega(\phi)$-expander $G[S']$ that accounts for a $(1 - O(\epsilon))$-fraction of the total volume. An algorithmic version of this structural result allows us to iteratively trim off a small piece of the graph until such S' is found. At this point, the algorithm of [16] can simply return $\{S'\} \cup (\cup_{u \in V \setminus S'} \{u\})$ as an ED, with at most $O(\epsilon|E|)$ inter-cluster edges between singletons. As the volume of the cluster multiplicatively decreases by $1 - O(\epsilon)$ after each call, this gives recursion depth at most $\widetilde{O}(1/\epsilon) \approx \widetilde{O}(1/\phi)$.

Our Contribution: Adaptation of Trimming

We show that a simple refinement of this approach allows us to adapt the framework of [30], which leads to an algorithm with recursion depth independent of ϕ. We run the same algorithm, but instead of separating each vertex in $V \setminus S'$ into its own singleton cluster, we recurse with Algorithm 1 on the whole set $V \setminus S'$. Conceptually, this implements an analogue of the trimming step of [30].

This means that at the end, fewer edges in $V \setminus S$ become inter-cluster edges. Because of that, we can strengthen the balancedness requirement: we recurse on the two sides of a sparse cut only if $\mathrm{vol}(S) \gtrsim \frac{1}{C} \mathrm{vol}(V \setminus S)$ for a large constant C. This allows us to trim more vertices each time, resulting in $\widetilde{O}(1)$ depth of the trimming step. On the other hand, because in each call to Algorithm 1 the volume of clusters passed to recursive calls is decreased by at least a constant factor, the total recursion depth becomes at most $\widetilde{O}(1)$.

Other than the refinement discussed above, our space efficient implementation of Algorithm 1, as well as the iterative procedure to find the large expander S', are almost the same as the ones of [16] (which in turn are inspired by the one of [9]). The details of the algorithms are given in the full version of this paper.

2.3 Putting It All Together

Combining the ideas illustrated in the two sections above, neither the sparsifier's size nor the recursion depth depend on ϕ, thus giving a sparsity-independent space algorithm for boundary-linked expander decomposition (BLD for short). This result is stated in terms of parameters $b, \epsilon, \phi \in (0,1)$, $\gamma \geq 1$: a (b,ϵ,ϕ,γ)-BLD is a partition $\mathcal{U}$ with at most an ϵ fraction of crossing edges and every $U \in \mathcal{U}$ induces a ϕ/γ-expander $G[U]^{b/\phi}$ (see Section 1.3). Using this terminology, we obtain the following result.

▶ **Theorem 7** (Exponential time decoding BLD). *Let $G = (V,E)$ be a graph given in a dynamic stream, and let $b \in (0,1)$ be a parameter such that $b \leq 1/\log^2 n$. Then, there is an algorithm that maintains a linear sketch of G in $\widetilde{O}(n/b^3)$ space. For any $\epsilon \in [n^{-2}, b\log n]$, the algorithm decodes the sketch to compute, with high probability and in $\widetilde{O}(n/b^3)$ space and $2^{O(n)}$ time, a (b,ϵ,ϕ,γ)-BLD of G for*

$$\phi = \Omega\left(\frac{\epsilon}{\log n}\right) \quad \text{and} \quad \gamma = O(1)\,.$$

From this, one can easily conclude our main result.

▶ **Theorem 8** (See Theorem 1). *Let $G = (V,E)$ be a graph given in a dynamic stream. Then, there is an algorithm that maintains a linear sketch of G in $\widetilde{O}(n)$ space. For any $\phi \in (0,1)$ such that $\phi \leq c/\log^2 n$ for a small enough constant $c > 0$, the algorithm decodes the sketch to compute a $(O(\phi \log n), \phi)$-ED of G with high probability, in $\widetilde{O}(n)$ space and $2^{O(n)}$ time.*

We remark that for $\phi \geq \Omega(1/\log^2 n)$, one can use the algorithm of [16] to still have an $\widetilde{O}(n)$ space construction of a $(O(\phi \log n), \phi)$-ED.

Proof. First note that without loss of generality we can assume $\phi \geq 1/n^2$, otherwise an ED can simply consist of the connected components of G (which can be computed in dynamic streams in $\widetilde{O}(n)$ space [4]). Then, we note that since c is small enough and $\phi \leq c/\log^2 n$, one can always define $\epsilon = C \cdot \phi \cdot \log n$ for an appropriate constant $C > 0$ while ensuring $1/n^2 \leq \epsilon \leq 1/\log n$. We can thus prove the theorem by equivalently showing that there is an algorithm that maintains a linear sketch of G in $\widetilde{O}(n)$ space, and that for all $\epsilon \in [1/n^2, 1/\log n]$ decodes the sketch to compute, with high probability, an $(\epsilon, \Omega(\epsilon/\log n)))$-ED of G in $\widetilde{O}(n)$ space and $2^{O(n)}$ time.

Let then $\epsilon \in [1/n^2, 1/\log n]$. We use the algorithm from Theorem 7 with parameter ϵ and a parameter b of our choice. We need to meet two preconditions: $b \leq 1/\log^2 n$ and $\epsilon \leq b \log n$. Since we assume $\epsilon \leq 1/\log n$, we can set $b = 1/\log^2 n$, and all the prerequisites are fulfilled. Then the algorithm from Theorem 7 runs in $2^{O(n)}$ time and takes $\widetilde{O}(n/b^3) = \widetilde{O}(n)$ bits of space. The output $\mathcal{U}$ is a (b,ϵ,ϕ,γ)-BLD of G with high probability, where $\phi = \Omega(\epsilon/\log n)$ and $\gamma = O(1)$. Since a (b,ϵ,ϕ,γ)-BLD of G is an $(\epsilon,\phi/\gamma)$-ED of G, we have obtained an $(\epsilon, \Omega(\epsilon/\log n))$-ED of G with high probability. ◀

The exponential time in the decoding is due to the subtask of finding a balanced sparse cut. As we show, one can make the decoding time polynomial by resorting to known offline approximation algorithms [29, 30]. However, we only have $\log^{\Omega(1)} n$-approximations for finding a balanced sparse cut, and in particular, we do not expect (under NP-hardness and the Unique Games Conjecture) there to be a polynomial time $O(1)$-approximation [10]. Such super-constant factor error incurs some loss in the quality of decomposition and space requirement, which, nevertheless, remains independent of the sparsity.

▶ **Theorem 9** (Polynomial time decoding BLD). *Let $G = (V, E)$ be a graph given in a dynamic stream, and let $b \in (0, 1)$ be a parameter such that $b \leq 1/\log^5 n$. Then, there is an algorithm that maintains a linear sketch of G in $n/b^3 \cdot \log^{O(\log n/\log \frac{1}{b})} n$ space. For any $\epsilon \in [n^{-2}, b \log n]$, the algorithm decodes the sketch to compute, with high probability and in $n/b^3 \cdot \log^{O(\log n/\log \frac{1}{b})} n$ space and $\mathrm{poly}(n)$ time, a $(b, \epsilon, \phi, \gamma)$-BLD of G for*

$$\phi = \Omega\left(\frac{\epsilon}{\log^4 n}\right) \quad \text{and} \quad \gamma = \log^{O\left(\frac{\log n}{\log 1/b}\right)} n\,.$$

A polynomial time version of Theorem 8 then follows from Theorem 9.

▶ **Theorem 10** (See Theorem 2). *Let $G = (V, E)$ be a graph given in a dynamic stream, and let $b \in (0, 1)$ be a parameter such that $b \leq 1/\log^5 n$. Then, there is an algorithm that maintains a linear sketch of G in $n/b^3 \cdot \log^{O(\log n/\log \frac{1}{b})} n$ space. For any $\phi \in (0, 1)$ such that $\phi \leq b/\log^{C \cdot \log n/\log \frac{1}{b}} n$ for a large enough constant $C > 0$, the algorithm decodes the sketch to compute a $(\phi \cdot \log^{O(\log n/\log \frac{1}{b})} n, \phi)$-ED of G with high probability, in $n/b^3 \cdot \log^{O(\log n/\log \frac{1}{b})} n$ space and $\mathrm{poly}(n)$ time.*

We remark that setting, say, $b = 2^{-\sqrt{\log n}}$ in the above result gives a $n^{1+o(1)}$ space algorithm for computing a $(\phi \cdot n^{o(1)}, \phi)$-ED for any $\phi \leq 2^{-2C \log\log n \sqrt{\log n}}$. For larger values of ϕ, one can use the polynomial time algorithm of [16] to still get a $n^{1+o(1)}$ space construction for a $(\phi \cdot n^{o(1)}, \phi)$-ED.

Proof. As in the proof of Theorem 8, we can assume $\phi \geq 1/n^2$. Also, by virtue of C being a large enough constant and $\phi \leq b/\log^{C \cdot \log n/\log \frac{1}{b}} n$, one can always define ϵ to be $\epsilon = \phi \cdot \log^{C \cdot \log n/\log \frac{1}{b}} n$ while ensuring $n^{-2} \leq \epsilon \leq b \log n$. Then, we equivalently prove that for any $b \in (0, 1)$ with $b \leq 1/\log^5 n$ there is an algorithm that maintains a linear sketch of G in $n/b^3 \cdot \log^{O(\log n/\log \frac{1}{b})} n$ space, and that for any $\epsilon \in (0, 1)$ such that $n^{-2} \leq \epsilon \leq b \log n$ decodes the sketch to compute, with high probability, an $(\epsilon, \epsilon/\log^{O(\log n/\log \frac{1}{b})} n)$-ED of G in $n/b^3 \cdot \log^{O(\log n/\log \frac{1}{b})} n$ space and $\mathrm{poly}(n)$ time.

Let then $b, \epsilon \in (0, 1)$ with $b \leq 1/\log^5 n$ and $n^{-2} \leq \epsilon \leq b \log n$. We use the algorithm from Theorem 9 with the same parameters b and ϵ, since every admissible pair of parameters b and ϵ fulfils the conditions of Theorem 9. The space complexity is also the same, and the running time is $\mathrm{poly}(n)$. Again, observe that a $(b, \epsilon, \phi, \gamma)$-BLD of G is an $(\epsilon, \phi/\gamma)$-ED of G. Hence the claim, since Theorem 9 gives

$$\frac{\phi}{\gamma} = \Omega\left(\frac{\epsilon}{\log^4 n}\right) \cdot \frac{1}{\log^{O\left(\frac{\log n}{\log 1/b}\right)} n} = \frac{\epsilon}{\log^{O\left(\frac{\log n}{\log 1/b}\right)} n}\,. \qquad \blacktriangleleft$$

The proofs of Theorem 7 and Theorem 9 can be found in the full version of this paper.

3 Two-Level Expander Decomposition Incurs a Sparsity Dependence

Given a graph $G = (V, E)$ in a stream and parameters $\epsilon, \phi \in (0, 1)$, we consider the problem of computing a two-level (ϵ, ϕ)-RED of G (see Section 1.3). In other words, we study the problem of computing an (ϵ, ϕ)-ED $\mathcal{U}$ of G and an (ϵ, ϕ)-ED $\mathcal{U}'$ of the graph $G' = (V, E \setminus \mathcal{U})$, where $E \setminus \mathcal{U}$ denotes the set of inter-cluster edges of $\mathcal{U}$, i.e. the edges of E that are not entirely contained in a cluster $U \in \mathcal{U}$. We remark that we wish to do so in a single pass over the stream.

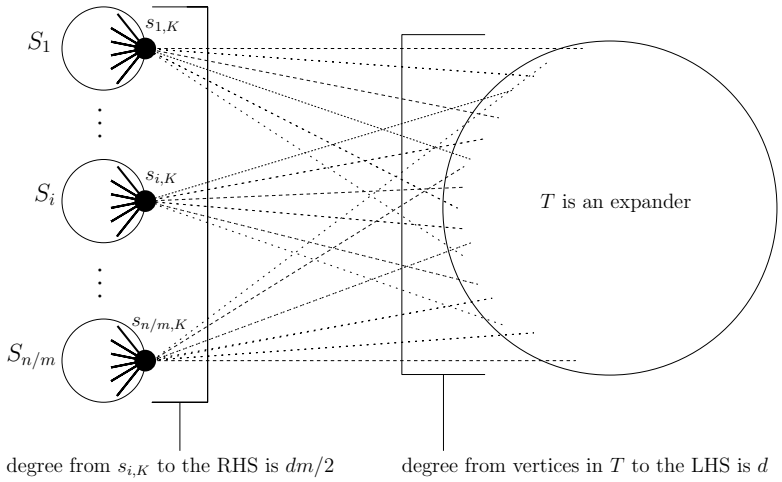

Figure 1 Illustration of the graph we use for proving the lower bound. Thick bullets represent important vertices, thick lines represent important edges, dotted lines represent edges connecting the important vertices to T.

A Natural Algorithmic Approach and Why It Fails

A naive attempt to solve this problem would be that of sketching the graph twice, for example by using our algorithm from Theorem 1. One can use the first sketch to construct the first level ED $\mathcal{U}$. After, the hope is that one can send updates to the second sketch so as to remove the intra-cluster edges and then decode this sketch into an ED of $G' = (V, E \setminus \mathcal{U})$ using again Theorem 1. However, this hope is readily dashed. Indeed, sketching algorithms break down if we send a removal update for an edge that was not there in the first place, and we do not have knowledge of which of the pairs $\binom{U}{2}$ are in E and which are not.

Our Contribution: Space Lower Bound

We show that, in sharp contrast to our algorithm for constructing a one-level expander decomposition, this problem requires $\widetilde{\Omega}(n/\phi)$ space, i.e. a dependence on $1/\phi$ is unavoidable. Formally, we obtain the following result.

▶ **Theorem 11** (See Theorem 3). *Let $\ell \geq 2$ and let $\epsilon, \phi \in (0,1)$ such that $\epsilon = 1 - \Omega(1)$, $\phi \leq \epsilon$, and $\phi \geq C \cdot \max\{\epsilon^2, 1/n\}$ for a large enough constant $C > 0$. Any streaming algorithm that with probability at least $9/10$ computes an ℓ-level (ϵ, ϕ)-RED requires $\Omega(n/\epsilon)$ bits of space.*

The above theorem gives an $\widetilde{\Omega}(n/\phi)$ space lower bound for algorithms that compute a RED with near-optimal parameters, i.e. algorithms that achieve $\epsilon = \widetilde{O}(\phi)$ for any $1/n \ll \phi \ll 1/\log n$.

Setup and Hard Instances

Throughout this section, the symbols $\ll$ and $\gg$ mean smaller or larger by a large constant factor. Let us fix the RED parameters $\epsilon, \phi \in (0,1)$, and let us restrain ourselves to the regime $\phi \gg \epsilon^2$ (and of course $\phi \ll \epsilon$). We prove the lower bound by giving a distribution over hard instances $G = (V, E)$. This distribution is parametrised by integers d and m such that $1 \ll d \ll m \ll n$, $m \gg 1/\phi$, $m \ll 1/\epsilon^2$, and $d \ll \frac{1}{\epsilon}$. With these parameters fixed, our hard distribution $\mathcal{G}$ is defined below in Definition 12. An illustration is given in Figure 1.

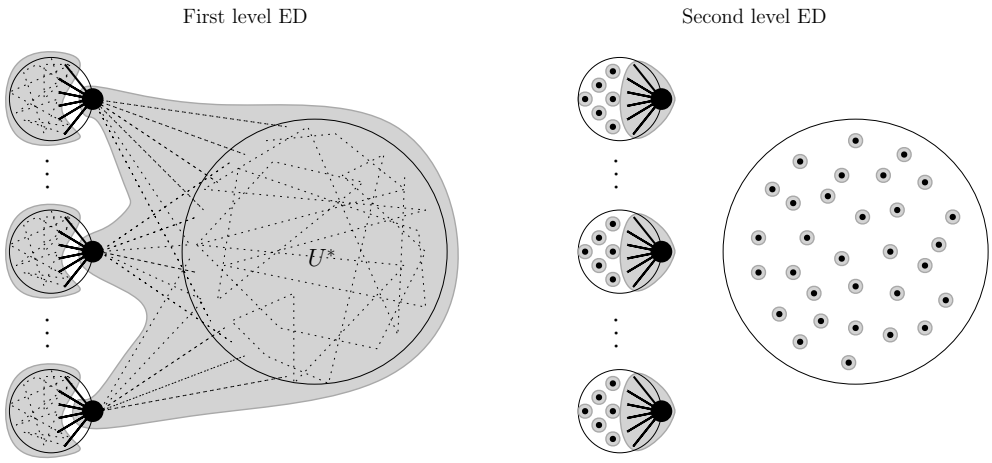

Figure 2 Illustration of the ideal expander decomposition of the graph. Thick bullets represent important vertices, smaller bullets represent ordinary vertices, thick lines represent important edges, dotted lines represent the rest of the edges. Grey areas represent the clusters in the decomposition.

▶ **Definition 12** (Distributions $\mathcal{G}$ and $\mathcal{G}'$ – Informal). *We partition V arbitrarily into two sets S and T with $n/2$ vertices each, and further partition S into n/m sets $S_1, \ldots, S_{n/m}$ with $m/2$ vertices each. The edge set of the graph $G = (V, E) \sim \mathcal{G}$ is defined as follows.*

1. *For each $i \in [n/m]$, the induced subgraph $G[S_i]$ is an Erdős-Rényi random graph with $m/2$ vertices and degree $\approx d$. We denote by $\mathcal{G}'$ the distribution of the subgraph $G[S]$.*
2. *The induced subgraph $G[T]$ is a* fixed *d-regular $\Omega(1)$-expander.*
3. *We fix for convenience an arbitrary labelling $s_{i,1}, \ldots, s_{i,m/2}$ of the vertices in each S_i, and we sample an index K uniformly from $[m/2]$. Then, for every $i \in [n/m]$, we add $dm/2$ edges from $s_{i,K}$ to T so that each $t \in T$ has d incident edges connecting to S.*

Roughly speaking, our hard instances should be composed of n/m regular expanders that are densely connected to T, which is also an expander, through a selection of "special" vertices. We will show that the hardness arises from recovering information about certain important vertices and edges, defined below and also illustrated in Figure 1.

▶ **Definition 13** (Important vertices and edges – Informal). *Let $G = (V, E) \sim \mathcal{G}$. We define the set of important vertices $V^* = \{s_{i,K} : i \in [n/m]\}$ to be the set of vertices of S that are connected to T, and define the set of important edges $E^* = \{\{s_{i,K}, v\} : i \in [n/m], v \in S\}$ to be the set of edges in the induced subgraph $G[S]$ that are incident on V^*.*

The lower bound proof has two steps: we first prove that a two-level RED leaks a non-trivial amount of information about the graph $G \sim \mathcal{G}$; then we prove that, in order to obtain such amount of information, the algorithm must use a lot of space.

3.1 Two-Level Expander Decomposition of the Hard Instance

In this section, we show that any valid two-level RED reveals a lot of informations about the important edges. The following lemma shows that a non-trivial amount of important edges are inter-cluster edges in the first level decomposition. An ideal decomposition is illustrated in Figure 2.

▶ **Lemma 14** (Informal). *Let $G = (V, E) \in \mathrm{supp}(\mathcal{G})$. Then, any (ϵ, ϕ)-ED $\mathcal{U}$ of G satisfies*

$$|E^* \setminus \mathcal{U}| \geq \frac{4}{5} \cdot |E^*| \ .$$

Proof sketch. By definition of the graph, there are $\Theta(dn)$ edges in the graph. Since there is at most an ϵ fraction of crossing edges, there are only $O(\epsilon dn)$ crossing edges. Note that there are $\Theta(dn)$ edges in $G[T]$, so only an $O(\epsilon)$ fraction of the edges in $G[T]$ are crossing edges. Furthermore, $G[T]$ is a regular expander: this implies that there is a large cluster $U^* \in \mathcal{U}$ comprising a $1 - O(\epsilon)$ fraction of T, together with a $1 - O(\epsilon)$ fraction of important vertices. The latter is true since the edges between S and T make up a constant fraction of the total volume and only a small fraction of the edges can be crossing. We refer the reader to the full version of this paper for more details.

The edges in the subgraph $G[S]$ also account for a constant fraction of the total volume. Together with the fact that each S_i induces a regular expander, for many of the S_i's we will have a cluster in $\mathcal{U}$ that contains most of S_i. Now consider a set S_i such that the important vertex of S_i is in U^* and most of the vertices in S_i are all in the same cluster. If most of the vertices in S_i are also inside U^*, then consider the cut from $(U^* \cap S_i) \setminus \{s_{i,K}\}$ to $(U^* \setminus S_i) \cup \{s_{i,K}\}$. The cut size is at most the number of important edges in S_i, which is $O(d)$. On the other hand, the volume of the cut is $\Theta(dm)$ since most of the vertices of S_i are in the cluster. Recalling that $m \gg \frac{1}{\phi}$, one concludes that the cut is sparse. Therefore, we ruled out the possibility of having many vertices of S_i in U^*. See the full version of this paper for a detailed discussion.

In summary, as illustrated in Figure 2, in any valid expander decomposition, most of the vertices in T and most of the important vertices are inside a giant cluster U^*, and for most of the S_i's, there is a cluster other than U^* that contains most of the vertices in S_i. For any such S_i, most of the important edges inside it are then crossing edges. ◀

The second level expander decomposition, i.e. an expander decomposition of the inter-cluster edges from the first level, is also quite structured, as illustrated in the ideal RED of Figure 2.

▶ **Lemma 15** (Informal). *Let $G = (V, E) \in \mathrm{supp}(\mathcal{G})$, and let $\mathcal{U}_1, \mathcal{U}_2$ be any 2-level (ϵ, ϕ)-RED of G. Then, there are at most $n/10$ vertices in S that are non-isolated vertices[3] in $\mathcal{U}_2$. Moreover, at least a $2/3$ fraction of important edges are not in $E \setminus \mathcal{U}_2$, i.e. a $2/3$ fraction of important edges are inside clusters of $\mathcal{U}_2$.*

Proof sketch. The number of crossing edges in the first level decomposition is $O(\epsilon dn)$, which is much less than n since $d \ll \frac{1}{\epsilon}$. This means that most of the vertices are isolated vertices in the second level. Moreover, by Lemma 14, most of the important edges are crossing edges in the first level decomposition. Among these $\Theta(dn/m)$ edges, at most $O(\epsilon^2 dn)$ edges can be crossing edges in the second level decomposition. Recalling that $m \ll \frac{1}{\epsilon^2}$, we see that most of the important edges are not crossing edges in the second level decomposition. ◀

3.2 Lower Bound via Communication Complexity

Our streaming lower bound will be proven in the two-player one-way communication model. In this setting, Alice gets the edges in $G[S]$ and $G[T]$, and Bob gets the edges between S and T. We prove that in order to give a two-level RED, Alice needs to send $\Omega(dn)$ bits of information to Bob.

The high level idea is the following. Note that the identity of the important vertices can be only revealed by edges given to Bob. Thus, given Alice's input, every vertex in S has the same probability to be an important vertex, which means that every edge in S has the same

[3] We call a vertex v non-isolated in a decomposition $\mathcal{U}$ if $\mathcal{U}$ puts v in a cluster with other vertices, i.e. v does not constitute a singleton cluster in $\mathcal{U}$.

probability to be an important edge. Therefore, in order to make sure that Bob recovers most of the important edges (which is morally equivalent to computing a two-level RED, as suggested by Figure 2), Alice needs to send most of the edges in S to Bob, which is $\Omega(dn)$.

To make the above idea concrete, we consider a communication problem where Alice is given a graph, and Bob is asked to output a not too large set of pairs that contains a good fraction of the edges of Alice's input. We first reduce this new problem to the two-level RED problem. Then, we will prove a communication complexity lower bound for this problem.

▶ **Definition 16** (Informal). *In the communication problem* RECOVER*, Alice's input is a graph $G' = (S, E')$ where $|S| = n/2$, and Bob's output is a set of pairs of vertices $F \subseteq \binom{S}{2}$ that must satisfy $|F| \leq nm/10$ and $|F \cap E'| \geq \Omega(|E'|)$, i.e. at least a constant fraction of the edges in G' are in F.*

Now, the idea is to plant Alice's input for RECOVER into our instance from Definition 12. The input distribution for RECOVER is sampled from the distribution $\mathcal{G}'$. In other words, the distribution of Alice's input is the same as the left-hand side part of $G \sim \mathcal{G}$ (i.e. the subgraph $G[S]$). Then, in the reduction, Alice gets the edges of $G[S]$ and $G[T]$ while Bob gets the edges between S and T. By virtue of the discussion in the previous section, we expect a RED of G to allow Bob to recover many important edges in $G[S]$. Hence, Bob could simulate the RED algorithm for all the $m/2$ possible choices of the random index $K \sim [m/2]$ that defines the important edges (see Definition 12): in this way, the k-th RED should reveal information about Alice's edges that are incident on the vertices $\{s_{i,k}\}_i$. Therefore, by varying k over $[m/2]$, Bob should obtain information about all the edges in Alice's graph. More precisely, the reduction is the following.

▶ Reduction (Informal). Let $\mathcal{A}$ be a deterministic streaming algorithm for computing a 2-level (ϵ, ϕ)-RED. Alice, given her input graph $G' = (S, E')$ generated by $\mathcal{G}'$, feeds her edges E' to $\mathcal{A}$, together with the fixed edges of $G[T]$. Then, she sends the memory state of $\mathcal{A}$ to Bob. Upon receiving the message, Bob makes $m/2$ copies of $\mathcal{A}$ and initialises them to the memory state he received from Alice. Call these copies $\mathcal{A}_1, \ldots, \mathcal{A}_{m/2}$. Next, for each $k \in [m/2]$, call $G_k = (V, E_k)$ the graph we obtain from $\mathcal{G}$ when $K = k$ and the left-hand side $G_k[S]$ is exactly Alice's input G' (so that the vertices $\{s_{i,k}\}_i$ are the important vertices in G_k, see Definition 12 and Definition 13). Then, Bob feeds the edges $E_k(S, T)$ to $\mathcal{A}_k$. Let then $\mathcal{U}_1^k, \mathcal{U}_2^k$ be the RED output by $\mathcal{A}_k$. Bob finally constructs his output set F as follows: for each $k \in [m/2]$, add the pair $\{s, s_{i,k}\}$ to F for every $i \in [n/m]$ and every vertex $s \in S_i$ that is not isolated in $\mathcal{U}_2^k$. ⌟

By Lemma 15, the number of non-isolated vertices in the second-level decomposition is at most $n/10$. Hence, the total number of pairs added to F is at most $nm/10$, thus satisfying the first requirement of Definition 16. Moreover, by Lemma 15, at least a $2/3$ fraction of the important edges are not crossing edges in the second level. This means that for each k, at least a $2/3$ fraction of the edges that are incident on $s_{i,k}$ is added to F. In turn, this implies that F contains at least a constant fraction of the edges in E', thus satisfying the second requirement of Definition 16. More precisely, one can prove the following.

▶ **Lemma 17** (Informal). *If there is a deterministic L-bit space streaming algorithm $\mathcal{A}$ that computes a 2-level (ϵ, ϕ)-RED with constant probability over inputs $G \sim \mathcal{G}$, then there is a deterministic protocol $\mathcal{R}$ that solves* RECOVER *with constant probability over inputs $G' \sim \mathcal{G}'$. The communication complexity of $\mathcal{R}$ is at most L.*

The final component of the proof is the communication complexity lower bound for RECOVER.

▶ **Lemma 18** (Informal). *The one-way communication complexity of solving* RECOVER *with constant probability over inputs sampled from* $\mathcal{G}'$ *is* $\Omega(dn)$.

Proof sketch. Roughly speaking, we show that the posterior distribution of the input conditioned on the output is shifted away from its prior distribution.

Recall that when the input $G' = (S, E')$ is sampled from $\mathcal{G}'$, the graph is a disjoint union of n/m random graphs with $m/2$ vertices each and degree $\approx d$. There are roughly

$$\left(\binom{m/2}{d}^{m/2}\right)^{n/m} \approx \left(\frac{m}{2d}\right)^{dn/2}$$

possible inputs in total and the information complexity is $\Omega(dn)$. Conditioning on the the output F of a correct protocol for RECOVER, the number of possible inputs is greatly decreased. In particular, to determine the input E', we need to select a constant fraction, say $2/3$ for example, of the pairs from F, and select the rest of the edges (a $1/3$ fraction, in our example) arbitrarily. Since $|F|$ is at most $nm/10$, $|E'| = \Theta(dn)$, and $\binom{S}{2} \approx nm/2$, the total number of possible inputs is then roughly

$$\binom{nm/10}{2/3 \cdot dn} \cdot \binom{nm/2}{1/3 \cdot dn} \approx \left(\frac{3m}{20d}\right)^{1/3 \cdot dn} \cdot \left(\frac{3m}{2d}\right)^{1/6 \cdot dn} < \left(\frac{m}{3d}\right)^{dn/2}.$$

This means the information complexity of the input is decreased by a constant factor, which means that the protocol needs to communicate $\Omega(dn)$ bits of information. ◀

Finally, one can conclude the main result Theorem 11 combining Lemma 17 and Lemma 18. The formal proofs and definitions are deferred to the full version of this paper.

References

1. Arpit Agarwal, Sanjeev Khanna, Huan Li, and Prathamesh Patil. Sublinear algorithms for hierarchical clustering. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL: `http://papers.nips.cc/paper_files/paper/2022/hash/16466b6c95c5924784486ac5a3feeb65-Abstract-Conference.html`.
2. AmirMahdi Ahmadinejad, John Peebles, Edward Pyne, Aaron Sidford, and Salil P. Vadhan. Singular value approximation and sparsifying random walks on directed graphs. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 846–854. IEEE, 2023. `doi:10.1109/FOCS57990.2023.00054`.
3. Kook Jin Ahn and Sudipto Guha. Graph sparsification in the semi-streaming model. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part II*, volume 5556 of *Lecture Notes in Computer Science*, pages 328–338. Springer, 2009. `doi:10.1007/978-3-642-02930-1_27`.
4. Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 459–467. SIAM, 2012. `doi:10.1137/1.9781611973099.40`.
5. Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In Michael Benedikt, Markus Krötzsch, and Maurizio Lenzerini, editors, *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 5–14. ACM, 2012. `doi:10.1145/2213556.2213560`.

6 Vedat Levi Alev, Nima Anari, Lap Chi Lau, and Shayan Oveis Gharan. Graph clustering using effective resistance. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 41:1–41:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICS.ITCS.2018.41.

7 András A. Benczúr and David R. Karger. Randomized approximation schemes for cuts and flows in capacitated graphs. *SIAM J. Comput.*, 44(2):290–319, 2015. doi:10.1137/070705970.

8 Yi-Jun Chang, Seth Pettie, Thatchaphol Saranurak, and Hengjie Zhang. Near-optimal distributed triangle enumeration via expander decompositions. *J. ACM*, 68(3):21:1–21:36, 2021. doi:10.1145/3446330.

9 Yi-Jun Chang and Thatchaphol Saranurak. Improved distributed expander decomposition and nearly optimal triangle enumeration. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 66–73. ACM, 2019. doi:10.1145/3293611.3331618.

10 Shuchi Chawla, Robert Krauthgamer, Ravi Kumar, Yuval Rabani, and D. Sivakumar. On the hardness of approximating multicut and sparsest-cut. *Comput. Complex.*, 15(2):94–114, 2006. doi:10.1007/S00037-006-0210-9.

11 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 612–623. IEEE, 2022. doi:10.1109/FOCS54457.2022.00064.

12 Yu Chen, Michael Kapralov, Mikhail Makarov, and Davide Mazzali. On the streaming complexity of expander decomposition, 2024. doi:10.48550/arXiv.2404.16701.

13 Yu Chen, Sanjeev Khanna, and Huan Li. On weighted graph sparsification by linear sketching. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 474–485. IEEE, 2022. doi:10.1109/FOCS54457.2022.00052.

14 Timothy Chu, Yu Gao, Richard Peng, Sushant Sachdeva, Saurabh Sawlani, and Junxing Wang. Graph sparsification, spectral sketches, and faster resistance computation, via short cycle decompositions. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 361–372. IEEE Computer Society, 2018. doi:10.1109/FOCS.2018.00042.

15 Julia Chuzhoy, Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, and Thatchaphol Saranurak. A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1158–1167. IEEE, 2020. doi:10.1109/FOCS46700.2020.00111.

16 Arnold Filtser, Michael Kapralov, and Mikhail Makarov. Expander decomposition in dynamic streams. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 50:1–50:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ITCS.2023.50.

17 Wai Shing Fung, Ramesh Hariharan, Nicholas J. A. Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 71–80. ACM, 2011. doi:10.1145/1993636.1993647.

18 Gramoz Goranci, Monika Henzinger, Danupon Nanongkai, Thatchaphol Saranurak, Mikkel Thorup, and Christian Wulff-Nilsen. Fully dynamic exact edge connectivity in sublinear time. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 70–86. SIAM, 2023. doi:10.1137/1.9781611977554.CH3.

19 Gramoz Goranci, Harald Räcke, Thatchaphol Saranurak, and Zihan Tan. The expander hierarchy and its applications to dynamic graph algorithms. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 2212–2228. SIAM, 2021. doi:10.1137/1.9781611976465.132.

20 Venkatesan Guruswami, Jun-Ting Hsieh, Pravesh K. Kothari, and Peter Manohar. Efficient algorithms for semirandom planted csps at the refutation threshold. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 307–327. IEEE, 2023. doi:10.1109/FOCS57990.2023.00026.

21 Yiding Hua, Rasmus Kyng, Maximilian Probst Gutenberg, and Zihang Wu. Maintaining expander decompositions via sparse cuts. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 48–69. SIAM, 2023. doi:10.1137/1.9781611977554.CH2.

22 Ravi Kannan, Santosh S. Vempala, and Adrian Vetta. On clusterings - good, bad and spectral. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 367–377. IEEE Computer Society, 2000. doi:10.1109/SFCS.2000.892125.

23 Michael Kapralov, Robert Krauthgamer, Jakab Tardos, and Yuichi Yoshida. Towards tight bounds for spectral sparsification of hypergraphs. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 598–611. ACM, 2021. doi:10.1145/3406325.3451061.

24 Michael Kapralov, Yin Tat Lee, Cameron Musco, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 561–570. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.66.

25 Michael Kapralov, Aida Mousavifar, Cameron Musco, Christopher Musco, Navid Nouri, Aaron Sidford, and Jakab Tardos. Fast and space efficient spectral sparsification in dynamic streams. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1814–1833. SIAM, 2020. doi:10.1137/1.9781611975994.111.

26 David R. Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In Vijaya Ramachandran, editor, *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas, USA*, pages 21–30. ACM/SIAM, 1993. URL: http://dl.acm.org/citation.cfm?id=313559.313605.

27 David R. Karger. Random sampling in cut, flow, and network design problems. In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 648–657. ACM, 1994. doi:10.1145/195058.195422.

28 Jason Li. Deterministic mincut in almost-linear time. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 384–395. ACM, 2021. doi:10.1145/3406325.3451114.

29 Jason Li, Danupon Nanongkai, Debmalya Panigrahi, and Thatchaphol Saranurak. Near-linear time approximations for cut problems via fair cuts. In Nikhil Bansal and Viswanath Nagarajan, editors, *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023, Florence, Italy, January 22-25, 2023*, pages 240–275. SIAM, 2023. doi:10.1137/1.9781611977554.CH10.

30 Thatchaphol Saranurak and Di Wang. Expander decomposition and pruning: Faster, stronger, and simpler. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2616–2635. SIAM, 2019. doi:10.1137/1.9781611975482.162.

31 Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 81–90. ACM, 2004. doi:10.1145/1007352.1007372.

32 Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011. doi:10.1137/08074489X.

33 Luca Trevisan. Approximation algorithms for unique games. *Theory Comput.*, 4(1):111–128, 2008. doi:10.4086/TOC.2008.V004A005.