

Expander Decomposition in Dynamic Streams

Arnold Filtser

Bar-Ilan University, Ramat-Gan, Israel

Michael Kapralov

EPFL, Lausanne, Switzerland

Mikhail Makarov

EPFL, Lausanne, Switzerland

Abstract

In this paper we initiate the study of expander decompositions of a graph $G = (V, E)$ in the streaming model of computation. The goal is to find a partitioning \mathcal{C} of vertices V such that the subgraphs of G induced by the clusters $C \in \mathcal{C}$ are good expanders, while the number of intercluster edges is small. Expander decompositions are classically constructed by a recursively applying balanced sparse cuts to the input graph. In this paper we give the first implementation of such a recursive sparsest cut process using small space in the dynamic streaming model.

Our main algorithmic tool is a new type of cut sparsifier that we refer to as a power cut sparsifier – it preserves cuts in any given vertex induced subgraph (or, any cluster in a fixed partition of V) to within a (δ, ϵ) -multiplicative/additive error with high probability. The power cut sparsifier uses $\tilde{O}(n/\epsilon\delta)$ space and edges, which we show is asymptotically tight up to polylogarithmic factors in n for constant δ .

2012 ACM Subject Classification Theory of computation → Streaming models; Theory of computation → Graph algorithms analysis; Theory of computation → Sketching and sampling

Keywords and phrases Streaming, expander decomposition, graph sparsifiers

Digital Object Identifier 10.4230/LIPIcs.ITCS.2023.50

Related Version *Full Version:* <https://arxiv.org/abs/2211.11384> [23]

Funding *Arnold Filtser:* Supported by the ISRAEL SCIENCE FOUNDATION (grant No. 1042/22).

Michael Kapralov: Supported in part by ERC Starting Grant 759471.

Mikhail Makarov: Supported by ERC Starting Grant 759471.

1 Introduction

Expanders are known to have many beneficial properties for algorithmic design. Therefore a natural divide-and-conquer approach leads to breaking a given graph into expanders. An (ϵ, ϕ) -expander decomposition (introduced by [28, 33]) of an n vertex graph $G = (V, E)$ is a partition $\mathcal{C} = \{C_1, \dots, C_k\}$ of the vertex set V such that the conductance of each induced graph $G\{C_i\}$ is at least ϕ , and such that there are at most $\epsilon \cdot |E|$ inter-cluster edges. For every $\epsilon \in (0, 1)$, one can always construct (ϵ, ϕ) -expander decomposition with $\phi = \Omega(\frac{\epsilon}{\log n})$, which is also the best possible (see, e.g., [4]). Expander decomposition have been found to be extremely useful for Laplacian solvers [46, 20], unique games [7, 49, 44], minimum cut [37], sketching/sparsification [6, 32, 19], max flow algorithms [38, 18], distributed algorithms [16, 14, 31] and dynamic algorithms [42, 50, 43, 13, 29].

Sequentially, expander decomposition can be constructed in almost linear time [45, 39]. Furthermore, expander decomposition have efficient $(\text{poly}(\epsilon^{-1}) \cdot n^{o(1)})$ rounds) constructions in distributed CONGEST model, and parallel PRAM and MPC models [27, 15, 16]. From the other hand, in the dynamic stream model (in fact even in insertions only model) it remained wide open whether it is possible to construct an expander decomposition using $\tilde{O}(n)$ space (which is the space required to store such a decomposition).



© Arnold Filtser, Michael Kapralov, and Mikhail Makarov;
licensed under Creative Commons License CC-BY 4.0

14th Innovations in Theoretical Computer Science Conference (ITCS 2023).

Editor: Yael Tauman Kalai; Article No. 50; pp. 50:1–50:13

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Graph sketching, introduced by [1] in an influential work on graph connectivity in dynamic streams, has been a de facto standard approach to constructing algorithms for dynamic streams, where the algorithm must use a small amount of space to process a stream that contains both edge insertions and deletions (while the vertex set is fixed). The main idea of [1] is to represent the input graph by its edge incident matrix, and to apply classical linear sketching primitives to the columns of this matrix. This approach seamlessly extends to dynamic streams, as by linearity of the sketch one can simply subtract the updates for deleted edges from the summary being maintained. A surprising additional benefit is the fact that such a sketching solution is trivially parallelizable: since the sketch acts on the columns of the edge incidence matrix, the neighborhood of every vertex in the input graph is compressed independently.

Sketching solutions have been constructed for many graph problems, including spanning forest computation [1], spanner construction [2, 36, 22, 24], matching and matching size approximation [9, 8], sketching the Laplacian [6, 32] and (among many other) most relevant to our paper, cut and spectral sparsifiers [3, 34, 35].

The main question we ask in this paper is:

What is the space complexity of constructing expander decompositions in the sketching model?

A classical way of constructing an expander decomposition (e.g., [47]) is to recursively apply nearly balanced sparsest cuts as long as a cut of sparsity at most ϕ exists. The recursion terminates in $O(\log n)$ depth as long as one does not recurse on large sides of unbalanced cuts. Indeed, if the graph does not include a sparse cut of balancedness more than $\frac{1}{4}$, say, it could be shown that the large part induces an $\Omega(\phi)$ -expander. Thus, one cuts $O(\phi \log n)$ fraction of edges over $O(\log n)$ levels of the recursion. Setting $\epsilon = O(\phi \log n)$, we get an $(\epsilon, \Omega(\frac{\epsilon}{\log n}))$ -expander decomposition, which is existentially optimal. The main question that our paper asks is

Can one design a streaming implementation of a **recursive** sparsest cut process?

A natural approach to emulating this process in the streaming model of computation would be to run it on a sparsifier of the input graph. Alas, graph cut sparsifiers do not preserve cuts in induced subgraphs¹, so other methods are needed. Since we would like to implement the recursive sparsest cut process in the streaming model, at the very least we need to be able to know when to stop, i.e. be able to verify that the partition that we created is already an $(\epsilon, \Omega(\frac{\epsilon}{\log n}))$ -expander decomposition. In particular, we ask

Is it possible to verify, using small space, that a partition given at the end of the stream is an expander decomposition of the input graph?

Our **first contribution** in this paper is a construction of what we call a power cut sparsifier – a random (reweighted) subgraph of the input graph that can be constructed in small space, and can be used to verify the $\Omega(\frac{\epsilon}{\log n})$ -expansion property in any fixed partition (not known before the stream) with high probability. The power cut sparsifier contains $\tilde{O}(n/\epsilon)$ edges, which we show is *optimal*.

One might hope that the construction of an ϵ -power cut sparsifier allows for direct simulation of the above mentioned recursive sparsest cut algorithm of [47]: one simply uses a fresh power cut sparsifier for every level of the recursion, taking $\tilde{O}(n/\epsilon) \cdot O(\log n) = \tilde{O}(n/\epsilon)$

¹ Preserving cuts in subgraphs of arbitrary size, for example, implies preserving them in subgraphs induced by pairs of nodes – and this requires preserving all the edges.

space overall. Surprisingly, this does not work! The issue is subtle: the termination condition that lets the [47] process not recurse on larger side of an unbalanced cut does not translate from the sparsifier to the actual graph, as it requires preservation of cuts in induced subgraphs – see Section 2.2 for a more detailed discussion. However, the more recent expander decomposition algorithm of Chang and Saranurak [15] works for our purposes. It also uses the recursive balanced sparse cut framework and the ideas introduced in [50, 42]. The termination condition is cut based, and translates from the sparsifier to the original graph. Since the depth of the recursion is $\tilde{O}(1/\epsilon)$, this gives us an $(\epsilon, \Omega(\frac{\epsilon}{\log n}))$ -expander decomposition using $\tilde{O}(n/\epsilon^2)$ space – our **second contribution** and the main result. We now state our results more formally:

► **Theorem 1.** *Given a dynamic stream containing a parameter $\epsilon \in (0, 1)$ and a graph $G = (V, E)$, there exists an algorithm that outputs a clustering \mathcal{C} of V that is a $(\epsilon, \Omega(\epsilon/\log n))$ -expander decomposition of G with high probability. This algorithm requires $\tilde{O}(\frac{n}{\epsilon^2})$ memory and takes exponential in n time.*

Notice that the decomposition quality in Theorem 1 is asymptotically optimal (see, e.g., [4]). Unfortunately, this algorithm takes exponential time due to the need to find balanced sparse cuts exactly. We also show how to make the runtime polynomial at the expense of slight losses in the quality of the expander decomposition and its space requirements:

► **Theorem 2.** *For an integer parameter $k \leq O(\log n)$, given a dynamic stream containing a parameter $\epsilon \in (0, 1)$ and a graph $G = (V, E)$, there exists an algorithm that outputs a clustering \mathcal{C} of V that is a $(\epsilon, \epsilon \cdot \Omega(\log n)^{-O(k)})$ -expander decomposition of G with high probability. This algorithm requires*

$$\tilde{O}(n) \cdot \left(\epsilon^{-2} + \epsilon^{\frac{1}{k}-1} \cdot n^{\frac{2}{k}} \cdot \log^{O(k)} n \right)$$

memory and takes polynomial in $n, 1/\epsilon, (\log n)^k$ time.

1.1 Power cut sparsifier

A natural approach for constructing expander decomposition is to find the sparsest cut (or an approximation), remove its edges and recurse on both sides. Once the sparsest cut found has conductance $\phi = \Omega(\frac{\epsilon}{\log n})$, the cluster at hand induces a ϕ -expander and the algorithm halts. One can show that in this entire process at most an ϵ -fraction of the edges is removed.

A cut sparsifier H of G (introduced by Benczur and Karger [12]) with multiplicative error $1 \pm \delta$ is a graph that preserves the values of all the cuts in G : $\forall S \subseteq V, |w_G(S, \bar{S}) - w_H(S, \bar{S})| \leq \delta \cdot w_G(S, \bar{S})$, where $w_G(S, \bar{S})$ is the sum of weights of all the edges crossing the cut S (quantity in unweighted graphs). In the dynamic stream model one can compute such a cut sparsifier using $\tilde{O}(\frac{n}{\delta^2})$ space [2, 35].

We introduce a new type of sparsifier dubbed *power cut sparsifier*. The name comes from the ambitious goal of sparsifying the entire power set of V . As this is impossible using the classical definition, we will also allow for an additive error. Denote by $\text{Vol}(S) = \sum_{v \in S} \deg(v)$ the sum of vertex degrees in S . $G\{S\}$ denotes the graph induced by S with self loops (see Section 1.4).

► **Definition 3.** *Consider a graph $G = (V, E_G)$. A graph $H = (V, E_H, w_H)$ is an (δ, ϵ) -cut sparsifier of G if for every cut $S \subseteq V$,*

$$(1 - \delta) \cdot w_G(S, \bar{S}) - \epsilon \cdot \text{Vol}(S) \leq w_H(S, \bar{S}) \leq (1 + \delta) \cdot w_G(S, \bar{S}) + \epsilon \cdot \text{Vol}(S)$$

A distribution \mathcal{D} over weighted subgraphs H is called a (δ, ϵ, p) -power cut sparsifier distribution for G , if for every partition \mathcal{C} of G , with probability at least $1 - p$, $\forall C \in \mathcal{C}$, $H\{C\}$ is a (δ, ϵ) -cut sparsifier of $G\{C\}$. A sample H from such a distribution is called a (δ, ϵ, p) -power cut sparsifier.

We show that power cut sparsifier can be constructed using very simple sampling algorithm: just add each edge $e = \{u, v\}$ to the sparsifier with probability $O\left(\frac{\log^2 n}{\epsilon \cdot \delta}\right) \cdot \left(\frac{1}{\deg(v)} + \frac{1}{\deg(u)}\right)$. Furthermore, we show that such a graph could be sampled in the dynamic stream model:

► **Theorem 4.** For every $C > 0$, and $\epsilon, \delta \in (0, 1)$, there exists an algorithm that uses $\tilde{O}(n \cdot \frac{C}{\epsilon \delta})$ space, that, given the edges of an n -vertex graph in a dynamic stream, in polynomial time produces samples from a $(\delta, \epsilon, n^{-C})$ -power cut sparsifier distribution.

Given the simplicity of the sampling algorithm, and the very robust guarantee provided by power cut sparsifiers, we believe that further application (beyond the construction of expander decomposition) will be found, and hence think that they are interesting objects of study in their own right. The dependence on ϵ in Theorem 4 is tight:

► **Theorem 5.** For every $\delta, \epsilon \in (\frac{1}{n}, \frac{1}{4})$ and even $n \in \mathbb{N}$, there is an n -vertex graph $G = (V, E)$ such that every $(\epsilon, \delta, \frac{1}{2})$ -power cut sparsifier distribution \mathcal{D} produces graphs with $\Omega(\frac{n}{\epsilon})$ edges in expectation.

In our application of Theorem 4 we will use $\frac{1}{\delta} = O(\log n)$. Thus for our purpose here, the power cut sparsifier of Theorem 4 is tight up to polylogarithmic factors. Furthermore, in the following theorem we show that Theorem 4 is also tight (up to polylogarithmic factors) for the setting of parameters $\epsilon = \delta$. In fact, this proposition holds even if one requires only a (δ, ϵ) -cut sparsifier, and not a power cut sparsifier, and even if we allow a general cut sketch (instead of a cut sparsifier).

► **Theorem 6.** For any $\epsilon, \delta \in (0, \frac{1}{4})$, any (δ, ϵ) -cut sketching scheme sk for unweighted multigraphs with n vertices at most n^2 edges must use at least $\Omega\left(\frac{n \log n}{\max\{\epsilon^2, \delta^2\}}\right)$ bits in the worst case.

1.2 Application to distance oracles

Given a graph $G = (V, E)$, a t -distance oracle DO is a data structure approximately answering distance queries, such that for every pair $u, v \in V$ one has

$$d_G(u, v) \leq \text{DO}(u, v) \leq t \cdot d_G(u, v),$$

where d_G stands for the shortest path metric of G . The parameter t is called the *stretch*. For every integer $k \geq 1$, every graph $G = (V, E)$ with n vertices admits a $(2k - 1)$ -distance oracle with $O(n^{1+1/k})$ space [17], which is optimal assuming the Erdős girth conjecture [21]. A fundamental question is what quality distance oracle one can construct in the dynamic streaming model (or alternatively using a linear sketch). This problems turned out to be very challenging, and the best known approximation using $\tilde{O}(n)$ space is only a $\tilde{O}(n^{\frac{2}{3}})$ -distance oracle by Filtser, Kapralov, and Nouri [24] (conjectured to be tight [24]). However, if $\tilde{O}(n^{1+\alpha})$ space is allowed, [24] constructed a $\tilde{O}(n^{\frac{2}{3}(1-\alpha)})$ -distance oracle. Furthermore, if the graph has at most m edges (and using $\tilde{O}(n^{1+\alpha})$ space), [24] constructed a $\tilde{O}(\sqrt{m} \cdot n^{-\alpha})$ -distance oracle.

We use our expander decomposition (Theorem 1) to construct the following distance oracle:

► **Theorem 7.** *Let $\alpha \in (0, 1)$ be a fixed constant. There is a streaming algorithm using $S = \tilde{O}(n^{1+\alpha})$ space, that given an n vertex graph with m edges (m not known in advance) in a dynamic stream fashion, returns a distance oracle with stretch $\frac{m}{S} \cdot \text{polylog}(n)$.²*

A caveat of the distance oracle of [24] is that the additional space (beyond $\tilde{O}(n)$) has very limited contribution. In particular, if the given space $\tilde{O}(n^{1+\alpha})$ is almost equal to m , the stretch is still a large polynomial. For example, let $\alpha = \frac{1}{4}$, and $m = n^{1+\alpha+\epsilon}$. Then the best stretch promised by [24] is only $\tilde{O}(\sqrt{m} \cdot n^{-\alpha}) = \tilde{O}(n^{\frac{1+\alpha+\epsilon}{2}-\alpha}) = \tilde{O}(n^{\frac{1-\alpha+\epsilon}{2}}) = \tilde{O}(n^{\frac{3}{8}+\frac{\epsilon}{2}})$. In our Theorem 7 we solve this issue, and obtain a distance oracle where the stretch goes to $\text{polylog}(n)$ as the space S goes to m . In particular, for the example above the stretch will be $\tilde{O}(n^\epsilon)$.

1.3 Related work

We refer the reader to [40, 41] for a survey of streaming algorithms. The idea of linear graph sketching was introduced in a seminal paper of Ahn, Guha, and McGregor [1]. An extension of the sketching approach to hypergraphs were presented in [30].

Graph sparsifiers with additive error were previously studied by Bansal, Svensson, and Trevisan [10]. They studied sparsifiers where all edge weights are equal. As a result, their additive error dependence also on average degree (rather than only on volume). Another crucial difference is that our analysis provides sparsification guarantees for cuts for the majority of subgraphs of G .

The distance oracle construction in [24] is only implicit, and actually they construct a graph spanner. A subgraph $H = (V, E)$ of a graph $G = (V, E)$ is a t -spanner of G if for every pair $u, v \in V$ one has $d_G(u, v) \leq d_H(u, v) \leq t \cdot d_G(u, v)$. Given a spanner H , one can construct a distance oracle DO by setting $\text{DO}(u, v) = d_H(u, v)$. The tradeoff between the stretch and the number of edges for spanners is the same as for distance oracles, where the celebrated greedy spanner [5, 25] has stretch $(2k - 1)$, and $O(n^{1+1/k})$ edges. In the dynamic streaming model, other than [24], all the previous work was concerned with spanner construction in multiple passes [11, 2, 36, 22, 24].

1.4 Preliminaries

All of the logarithms in the paper are in base 2. We use \tilde{O} notation to suppress constants and poly-logarithmic factors in n , that is $\tilde{O}(f) = f \cdot \text{polylog}(n)$ (in particular, $\tilde{O}(1) = \text{polylog}(n)$).

Given an weighted graph $G = (V, E, w_G)$, the weighted degree $\text{deg}_G(v)$ represents the sum of weights of all edges incident on v , including self-loops (simply degree for unweighted). Given a set $S \subseteq V$, $G\{S\}$ denotes the graph induced by S with self loops. That is, the vertex set of $G\{S\}$ is S , we keep all the edges where both endpoints are in S , and we add self loops to every vertex such that its degree in G and $G\{S\}$ is the same. We denote the complement set of S by $\bar{S} = V \setminus S$.

The volume of a set S is the sum of the weighted degrees of all the vertices in S : $\text{Vol}_G(S) = \sum_{v \in S} \text{deg}_G(v)$. By $E_G(A, B)$ we represent the set of edges between vertex sets A and B , and by $w_G(A, B) = \sum_{e \in E_G(A, B)} w_G(e)$ the sum of their weights (for unweighted their count). Where the graph G is clear from the context, we might abuse notation and drop the subscript (e.g. $\text{Vol}(S) := \text{Vol}_G(S)$). We say that a set of vertices $S \subset V$ is a cut if $\emptyset \subsetneq S \subsetneq V$. The value $w_G(S, \bar{S})$ is called the size of the cut S , and the sparsity of a cut S in graph G is $\Phi_G(S) = \frac{w_G(S, \bar{S})}{\min\{\text{Vol}_G(S), \text{Vol}_G(\bar{S})\}}$. We say that a cut S is ϕ -sparse if $\Phi_G(S) \leq \phi$.

² We implicitly assume here that $m \geq \frac{S}{\text{polylog}(n)}$. Otherwise, one can simply use sparse recovery to recover the entire graph G , and thus construct a distance oracle with stretch 1.

50:6 Expander Decomposition in Dynamic Streams

► **Definition 8.** For $\phi \in [0, 1]$, a graph $G = (V, E, w_G)$ is a ϕ -expander if the sparsity of every cut is at least ϕ , i.e. $\min_{\emptyset \subsetneq S \subsetneq V} \Phi_G(S) \geq \phi$.

► **Definition 9.** For $\phi, \epsilon \in [0, 1]$ and a graph $G = (V, E)$, a partition \mathcal{C} of the vertices V is called an (ϵ, ϕ) -expander decomposition of G if

- the number of inter-cluster edges is at most ϵ fraction of all edges:

$$\sum_{C \in \mathcal{C}} w(C, \bar{C}) \leq \epsilon \cdot \text{Vol}(V)$$

- for each cluster $C \in \mathcal{C}$, $G\{C\}$ is a ϕ -expander.

Dynamic streams

We say that a graph G is given to us in a dynamic stream if we are given a fixed set V of n vertices and a sequence of updates on unweighted edges, where each update either adds an edge between two vertices or removes an existing edge. We define G to be the graph at the end of the stream.

Additive Chernoff bound

The following concentration bound is used in proof of the power cut sparsifier guarantees.

► **Lemma 10 (Additive Chernoff Bound).** Let X_1, \dots, X_n be independent random variables distributed in $[0, a]$. Let $X = \sum_{i \in [n]} X_i$, $\mu = \mathbb{E}[X]$. Then for $\epsilon \in (0, 1)$ and $\alpha \geq 0$

$$\Pr[|X - \mu| > \epsilon\mu + \alpha] \leq 2 \exp\left(-\frac{\epsilon\alpha}{3a}\right).$$

2 Technical overview

2.1 Verifying expander decompositions in a dynamic stream

Before explaining our algorithm for the expander decomposition construction, we first consider a problem of *verifying* an expander decomposition. That is, given a graph G in a dynamic stream and after it a clustering \mathcal{C} of its vertices and the values ϵ, ϕ , we want to distinguish between the cases when \mathcal{C} is an (ϵ, ϕ) -expander decomposition of G and when the fraction of intercluster edges is at least $O(\epsilon)$ or some cluster contains a $O(\phi)$ -sparse cut.

Using graph sparsifiers

The core idea that allows us both to solve this problem and construct the expander decomposition using only limited memory is the use of the graph sparsifiers. According to the classical definition [26], the δ -cut sparsifier of a graph $G = (V, E)$ is a graph $H = (V, E', w)$ such that, for any cut $\emptyset \subsetneq S \subsetneq V$

$$(1 - \delta) \cdot w_G(S, \bar{S}) \leq w_H(S, \bar{S}) \leq (1 + \delta) \cdot w_G(S, \bar{S}).$$

It is well known that one can construct a cut sparsifier in the streaming setting [2] using $\tilde{O}(n/\delta^2)$ space. Having such a sparsifier, we can already distinguish between the cases when G is a ϕ -expander and when there is a $\frac{1-\delta}{1+\delta}\phi$ -sparse cut in G just by checking all possible cuts in H .

Things become more involved when we try to verify an expander decomposition. Now, we need to check that for some partitioning of V into clusters \mathcal{C} , all $G\{C\}$, $C \in \mathcal{C}$, are ϕ -expanders. Here, the classical notion of the cut sparsifier is insufficient, as the approach of checking every cut in each of the $G\{C\}$, $C \in \mathcal{C}$ would require that we have a cut sparsifier for each $G\{C\}$.

Simple sparsifier for a partition

A natural solution for this is to construct a cut sparsifier H in such a way that for any fixed partition \mathcal{C} , $H\{C\}$ will be a cut sparsifier of $G\{C\}$ for all $C \in \mathcal{C}$ with high probability. From existing literature, we can already derive how to do that for graphs G that are themselves ϕ expanders [47]. Consider the following sampling procedure: the graph $G_{\phi,\delta}$ is obtained by taking each edge $e = \{u, v\} \in E$ with probability at least $p_e \geq \min \left\{ 1, \left(\frac{1}{\deg(u)} + \frac{1}{\deg(v)} \right) \cdot O \left(\frac{\log^2 n}{\delta^2 \phi^2} \right) \right\}$ and weight $\frac{1}{p_e}$. By Spielman and Teng [47], $G_{\phi,\delta}$ is sparsifier w.h.p.:

► **Lemma 11** ([47]). *Let $G = (V, E)$ be a ϕ -expander, then with probability $1 - n^{-\Omega(1)}$, $G_{\phi,\delta}$ is $1 + \delta$ spectral sparsifier, and thus cut sparsifier of G .*

A key observation is that if we were to apply this procedure to $G\{C\}$ for some cluster C , the sampling probabilities of each non-loop edge would remain the same since the degrees of vertices are unchanged. Because of that, with some additional work, one can show that $H\{C\}$ is a $(1 + \delta)$ -cut sparsifier of $G\{C\}$ for each $C \in \mathcal{C}$ with high probability.

Intuition behind power cut sparsifiers

But how to adopt this approach to the case where G is not an expander? We observe that for our purposes we can allow a small additive error in cut size estimation in terms of the volume of the cut. Namely, suppose that we have a sparsifier H with the following guarantee: for any partition \mathcal{C} of V with high probability for any cut $\emptyset \subsetneq S \subsetneq V$,

$$(1 - \delta) \cdot w_{G\{C\}}(S, \bar{S}) - \epsilon \cdot \text{Vol}(S) \leq w_{H\{C\}}(S, \bar{S}) \leq (1 + \delta) \cdot w_{G\{C\}}(S, \bar{S}) + \epsilon \cdot \text{Vol}(S).$$

See Definition 3. This sparsifier is suitable for checking whether each $G\{C\}$ is an expander, since this guarantee implies the following guarantee for expansion of the cut S :

$$(1 - \Theta(\delta))\Phi_{G\{C\}}(S) - \Theta(\epsilon) \leq \Phi_{H\{C\}} \leq (1 + \Theta(\delta))\Phi_{G\{C\}}(S) + \Theta(\epsilon).$$

Since we only want to check whether the expansion is bigger than ϕ with constant precision, it is enough for δ to be a small constant and $\epsilon = O(\phi)$. It turns out that we can construct such a sparsifier, which we call a power cut sparsifier, by sampling each edge with probability $p_e \geq \min \left\{ 1, \left(\frac{1}{\deg(u)} + \frac{1}{\deg(v)} \right) \cdot O \left(\frac{\log^2 n}{\delta \epsilon} \right) \right\}$. This results in the sparsifier having size $O \left(\frac{n \log^2 n}{\delta \epsilon} \right)$, featuring only *linear* dependence on $1/\epsilon$. This dependence is optimal, as we show in Theorem 5.

Showing correctness

Our proof that such a sampling scheme indeed produces a power cut sparsifier is based on the idea of decomposing cuts into d -projections, introduced by Fung et al. [26]. While [26] work with sampling probabilities based on edge connectivities, in our case the probabilities only depend on the degrees of the vertices. We define the d -projection of a cut S to be the

subset of edges of $E(S, \bar{S})$ whose both endpoints have degree at least d , and show that the overall number of distinct d -projections is polynomially bounded. The rest of the proof is standard: we show that the value of each d -projection is preserved with high probability by using a multiplicative-additive version of Chernoff bound, and then take a union bound over all projections.

2.2 Constructing an expander decomposition

Naive approach

Having seen that we can verify an expander decomposition using a power cut sparsifier, a question arises: can we construct an expander decomposition provided with only power cut sparsifiers? A natural approach would be to construct a decomposition of the sparsifier using one of the existing methods and then to claim that it is also a decomposition of the original graph. Unfortunately, this plan is flawed, because the resulting decomposition would *depend* on the used power cut sparsifier, which violates the implicit assumption that the partition \mathcal{C} is chosen *independently* from the sparsifier. Hence we cannot claim that the produced clustering is an expander decomposition. One can attempt to verify that using a fresh copy of the power cut sparsifier. However, if the verification fails we are back at square one.

Recursive approach

To explain our algorithm let's first examine the classical recursive approach to expander decomposition construction [48]. The simplest version is as follows: find a sparse cut S , make this cut and then recurse on both sides $S, V \setminus S$. As we have seen, we cannot run all of those steps on one sparsifier, so we would use a fresh power cut sparsifier for each level of recursion.

A priori, one of the sides of the sparse cut might be very small, implying that the recursion depth of this algorithm might be even of linear size, resulting in a quadratic space requirement overall. Instead, we employ a classic approach of Spielman and Teng [48]. The idea is to use a balanced sparse cut procedure to find a balanced sparse cut in the sparsifier, i.e. a cut that is sparse and both sides of which have approximately the same volume. If we use this procedure instead of just finding an arbitrary sparse cut, we can guarantee that the recursion depth is only logarithmic. However, there are a couple caveats that we discuss next.

Balanced sparse cut procedure

A balanced sparse cut procedure typically takes the sparsity ϕ as an input and has three possible outcomes:

- It declares that the graph is a $O(\phi)$ -expander.
- It finds a balanced $O(\phi)$ -sparse cut. A cut is balanced if both of its sides have approximately the same size.
- It finds an unbalanced $O(\phi)$ -sparse cut, but gives some kind of guarantee on the largest side of the cut. For example, it may guarantee that it is a $O(\phi)$ -expander.

If one uses it in the recursive algorithm outlined above, first outcome means that we can stop recursing on the graph. The second means that the cut we make is balanced, so we make the cut and recurse. Finally, in the last outcome because the bigger side of the cut is $O(\phi)$ -expander, we can recurse only on the smaller side of the cut. This means that when we recurse, the volume of the part that we recurse on always decreases by a constant fraction, so the depth of the recursion is at most logarithmic.

In our case, we run it on a sparsifier H and are interested in implications for the original graph G . The first two outcomes translate from a sparsifier to the original graph with only a small error, as they both concern only the cuts in the graph. The last outcome suffers the most: it concerns the cut structure of some subgraph $H\{C\}$ of H . But because the choice of C is dependent on H , we cannot guarantee that $H\{C\}$ is a sparsifier of $G\{C\}$. Hence the guarantee that $H\{C\}$ is an expander only translates into the guarantee that $G\{C\}$ is a near-expander, which is a much weaker guarantee. This motivates us to consider an expander decomposition that uses a version of balanced sparse cut procedure with a relatively weak third guarantee, but one which survives this transition. We opt to adopt the approach of Chang and Saranurak [15].

Two phase approach

The weak third guarantee makes bounding the depth of the recursion difficult. That is why the algorithm is split into two phases. The first phase is recursive: given a vertex set C , we run the most balanced sparse cut procedure on sparsifier $H\{C\}$. If it declares that it is an expander, then the graph $G\{C\}$ is also an expander and we terminate. If there is a balanced sparse cut S , we make this cut and recurse on vertex sets S and \bar{S} with a fresh sparsifier. Otherwise, if the cut is more than $O(\phi)$ unbalanced we execute second phase on C .

The idea behind the second phase is that if $G\{C\}$ doesn't have a $O(\phi)$ -balanced sparse cut for sparsity ϕ , then if we were to successively make cuts of sparsity less than ϕ , we could not cut more than $O(\phi)$ fraction of the total volume of C . Because we have a small error in approximating the cut size and, therefore, the cut sparsity, due to the sparsifier usage, we decrease the sparsity of the cut that we are looking for by multiplying it by a close to 1 constant $c < 1$ to mitigate that error. Furthermore, we divide the required cut balancedness by a constant. We use the balanced sparse cut procedure with sparsity $c\phi$ and continue in the same manner as in phase one, except that if there is a balanced sparse cut, we split the smaller side of the cut into singletons and continue with the larger. Finally, when we find an unbalanced cut, we use the same argument and decrease ϕ and balancedness even further. Eventually, the balancedness threshold becomes so small that all cuts are trivially balanced, and thus the algorithm halts and declares the cluster to be an expander.

2.3 Distance Oracle

We construct a distance oracle for $G = (V, E)$ by constructing a t -emulator, which is a graph $H = (V, E, w_H)$ preserving all distances in G up to a t factor. Emulator is a similar notion to spanner (defined in Section 1.3) but without the requirement of being a subgraph. The key fact we use is that every ϕ -expander has diameter at most $\Delta = O(\frac{\log n}{\phi})$. Each (unweighted) graph of diameter Δ admits a 2Δ -emulator of linear size (simply take an arbitrary center x , and add edges from x to all other vertices of weight Δ). Thus, given a graph $G = (V, E)$ with m edges in a streaming fashion, we can first produce an (ϵ, ϕ) -expander decomposition \mathcal{C} , construct an $O(\frac{\log n}{\phi}) = O(\frac{\log^2 n}{\epsilon})$ -emulator H_C for each cluster $C \in \mathcal{C}$, and use sparse recovery to recover the set E_C of $\epsilon \cdot m$ inter cluster edges. The graph $H = E_C \cup \bigcup_{C \in \mathcal{C}} H_C$ will be an $O(\frac{\log^2 n}{\epsilon})$ -emulator, while the required space is $\tilde{O}(\frac{n}{\epsilon^2} + \epsilon m)$, which might be larger than our budget S (for every choice of ϵ).

To overcome this issue we construct an expander hierarchy. That is after constructing the (ϵ, ϕ) -expander decomposition \mathcal{C} , we obtain a new graph G_2 by “contracting” all the expanders in \mathcal{C} into super nodes, and keeping only the inter cluster edges. A key property is that given a linear sketch for G , we can transform it into a linear sketch for G_2 without

knowing \mathcal{C} in advance. We then create expander decomposition for G_2 , and continue in this manner in creating super graphs G_3, \dots, G_k . It holds that the number of edges in G_k is bounded by $\epsilon^{k-1} \cdot m$, while the diameter (w.r.t. G) of each cluster in the k 'th expander decomposition is bounded by $O(\frac{\log^2 n}{\epsilon})^k$. Picking the right parameters ϵ, k we can recover all the edges in G_k and obtain the desired emulator.

References

- 1 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 459–467, 2012. doi:10.1137/1.9781611973099.40.
- 2 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.
- 3 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Spectral sparsification in dynamic graph streams. In Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, volume 8096 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2013. doi:10.1007/978-3-642-40328-6_1.
- 4 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.
- 5 Ingo Althöfer, Gautam Das, David P. Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discret. Comput. Geom.*, 9:81–100, 1993. doi:10.1007/BF02189308.
- 6 Alexandr Andoni, Jiecao Chen, Robert Krauthgamer, Bo Qin, David P. Woodruff, and Qin Zhang. On sketching quadratic forms. In Madhu Sudan, editor, *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 311–319. ACM, 2016. doi:10.1145/2840728.2840753.
- 7 Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. *J. ACM*, 62(5):42:1–42:25, 2015. doi:10.1145/2775105.
- 8 Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1723–1742. SIAM, 2017. doi:10.1137/1.9781611974782.113.
- 9 Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1345–1364. SIAM, 2016. doi:10.1137/1.9781611974331.ch93.
- 10 Nikhil Bansal, Ola Svensson, and Luca Trevisan. New notions and constructions of sparsification for graphs and hypergraphs. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 910–928. IEEE, 2019.
- 11 Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Struct. Algorithms*, 30(4):532–563, 2007. doi:10.1002/rsa.20130.

- 12 András A. Benczúr and David R. Karger. Approximating s - t minimum cuts in $\tilde{O}(n^2)$ time. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 47–55, 1996. doi:10.1145/237814.237827.
- 13 Aaron Bernstein, Maximilian Probst Gutenberg, and Thatchaphol Saranurak. Deterministic decremental reachability, scc, and shortest paths via directed expanders and congestion balancing. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1123–1134. IEEE, 2020. doi:10.1109/FOCS46700.2020.00108.
- 14 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.
- 15 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.
- 16 Yi-Jun Chang and Thatchaphol Saranurak. Deterministic distributed expander decomposition and routing with applications in distributed derandomization. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 377–388. IEEE, 2020. doi:10.1109/FOCS46700.2020.00043.
- 17 Shiri Chechik. Approximate distance oracles with improved bounds. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 1–10. ACM, 2015. doi:10.1145/2746539.2746562.
- 18 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. *CoRR*, abs/2203.00671, 2022. doi:10.48550/arXiv.2203.00671.
- 19 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.
- 20 Michael B. Cohen, Jonathan A. Kelner, John Peebles, Richard Peng, Anup B. Rao, Aaron Sidford, and Adrian Vladu. Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 410–419. ACM, 2017. doi:10.1145/3055399.3055463.
- 21 P. Erdős. Extremal problems in graph theory. *Theory of Graphs and Its Applications (Proc. Sympos. Smolenice)*, pages 29–36, 1964. see here.
- 22 Manuel Fernandez, David P. Woodruff, and Taisuke Yasuda. Graph spanners in the message-passing model. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, pages 77:1–77:18, 2020. doi:10.4230/LIPIcs.ITCS.2020.77.
- 23 Arnold Filtser, Michael Kapralov, and Mikhail Makarov. Expander decomposition in dynamic streams, 2022. doi:10.48550/ARXIV.2211.11384.
- 24 Arnold Filtser, Michael Kapralov, and Navid Nouri. Graph spanners by sketching in dynamic streams and the simultaneous communication model. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 1894–1913, 2021. doi:10.1137/1.9781611976465.113.
- 25 Arnold Filtser and Shay Solomon. The greedy spanner is existentially optimal. *SIAM J. Comput.*, 49(2):429–447, 2020. doi:10.1137/18M1210678.

- 26 Wai Shing Fung, Ramesh Hariharan, Nicholas J. A. Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. *SIAM J. Comput.*, 48(4):1196–1223, 2019. doi:10.1137/16M1091666.
- 27 Mohsen Ghaffari, Fabian Kuhn, and Hsin-Hao Su. Distributed MST and routing in almost mixing time. In Elad Michael Schiller and Alexander A. Schwarzmann, editors, *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 131–140. ACM, 2017. doi:10.1145/3087801.3087827.
- 28 Oded Goldreich and Dana Ron. A sublinear bipartiteness tester for bounded degree graphs. *Comb.*, 19(3):335–373, 1999. doi:10.1007/s004930050060.
- 29 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.
- 30 Sudipto Guha, Andrew McGregor, and David Tench. Vertex and hyperedge connectivity in dynamic graph streams. In Tova Milo and Diego Calvanese, editors, *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 241–247. ACM, 2015. doi:10.1145/2745754.2745763.
- 31 Bernhard Haeupler, Harald Räcke, and Mohsen Ghaffari. Hop-constrained expander decompositions, oblivious routing, and distributed universal optimality. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1325–1338. ACM, 2022. doi:10.1145/3519935.3520026.
- 32 Arun Jambulapati and Aaron Sidford. Efficient $\tilde{O}(n/\epsilon)$ spectral sketches for the laplacian and its pseudoinverse. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2487–2503. SIAM, 2018. doi:10.1137/1.9781611975031.159.
- 33 Ravi Kannan, Santosh S. Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *J. ACM*, 51(3):497–515, 2004. doi:10.1145/990308.990313.
- 34 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.
- 35 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.
- 36 Michael Kapralov and David P. Woodruff. Spanners and sparsifiers in dynamic streams. In *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 272–281, 2014. doi:10.1145/2611462.2611497.
- 37 Ken-ichi Kawarabayashi and Mikkel Thorup. Deterministic edge connectivity in near-linear time. *J. ACM*, 66(1):4:1–4:50, 2019. doi:10.1145/3274663.
- 38 Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 217–226. SIAM, 2014. doi:10.1137/1.9781611973402.16.
- 39 Jason Li and Thatchaphol Saranurak. Deterministic weighted expander decomposition in almost-linear time. *CoRR*, abs/2106.01567, 2021. arXiv:2106.01567.
- 40 Andrew McGregor. Graph stream algorithms: A survey. *SIGMOD Rec.*, 43(1):9–20, 2014. ICALP2004. doi:10.1145/2627692.2627694.

- 41 Andrew McGregor. Graph sketching and streaming: New approaches for analyzing massive graphs. In Pascal Weil, editor, *Computer Science - Theory and Applications - 12th International Computer Science Symposium in Russia, CSR 2017, Kazan, Russia, June 8-12, 2017, Proceedings*, volume 10304 of *Lecture Notes in Computer Science*, pages 20–24. Springer, 2017. doi:10.1007/978-3-319-58747-9_4.
- 42 Danupon Nanongkai and Thatchaphol Saranurak. Dynamic spanning forest with worst-case update time: adaptive, las vegas, and $o(n^{1/2 - \epsilon})$ -time. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1122–1129. ACM, 2017. doi:10.1145/3055399.3055447.
- 43 Danupon Nanongkai, Thatchaphol Saranurak, and Christian Wulff-Nilsen. Dynamic minimum spanning forest with subpolynomial worst-case update time. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 950–961. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.92.
- 44 Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 755–764. ACM, 2010. doi:10.1145/1806689.1806792.
- 45 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.
- 46 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.
- 47 Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011. doi:10.1137/08074489X.
- 48 Daniel A. Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM J. Comput.*, 42(1):1–26, 2013. doi:10.1137/080744888.
- 49 Luca Trevisan. Approximation algorithms for unique games. *Theory Comput.*, 4(1):111–128, 2008. doi:10.4086/toc.2008.v004a005.
- 50 Christian Wulff-Nilsen. Fully-dynamic minimum spanning forest with improved worst-case update time. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1130–1143. ACM, 2017. doi:10.1145/3055399.3055415.