Expander Decomposition with Almost Optimal Overhead

Nikhil Bansal
University of Michigan, Ann Arbor, MI, USA

Arun Jambulapati
Independent Researcher, USA

Thatchaphol Saranurak
University of Michigan, Ann Arbor, MI, USA

Abstract

We present the first polynomial-time algorithm for computing a near-optimal *flow*-expander decomposition. Given a graph G and a parameter ϕ, our algorithm removes at most a $\phi \log^{1+o(1)} n$ fraction of edges so that every remaining connected component is a ϕ-*flow*-expander (a stronger guarantee than being a ϕ-*cut*-expander). This achieves overhead $\log^{1+o(1)} n$, nearly matching the $\Omega(\log n)$ graph-theoretic lower bound that already holds for cut-expander decompositions, up to a $\log^{o(1)} n$ factor. Prior polynomial-time algorithms required removing $O(\phi \log^{1.5} n)$ and $O(\phi \log^2 n)$ fractions of edges to guarantee ϕ-cut-expander and ϕ-flow-expander components, respectively.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Approximation algorithms analysis

Keywords and phrases Graph algorithms, expander decomposition, flow expansion, sparse cuts

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.22

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2602.15015`

Funding *Nikhil Bansal*: Supported by NSF awards CCF-2327011 and CCF-2504995.
Arun Jambulapati: Supported by the NWO VICI grant 639.023.812 awarded to NB.
Thatchaphol Saranurak: Supported by NSF Grant CCF-2238138 and a Sloan Fellowship.

1 Introduction

Expander decomposition is a central structural primitive in graph algorithms – where one removes a small fraction of edges from a given graph G, so that every connected component in the resulting graph has some desired expansion. This notion first appeared implicitly in property testing [23] and was defined explicitly in [31].

As many problems can be solved better on expanders, over the last two decades, expander decomposition has been used widely in approximation algorithms [11, 20, 10], fast graph algorithms [43, 32, 42, 18, 14], dynamic data structures [37, 36, 7, 25, 24, 30, 44], parallel and distributed algorithms [8, 9, 27, 13, 26], and streaming algorithms [16, 21, 15]. A lot of these works have also focussed on speeding up constructions in various models of computation, or extending the notion of expansion in various interesting ways to obtain new applications.

Edges removed vs. Expansion

A basic and natural question is to understand how many edges must be removed to achieve a desired expansion ϕ. Equivalently, given a budget on the number of edges that can be removed, what is the best achievable expansion.

Remarkably, this question is completely understood *existentially* – for any graph on m edges, removing $O(\phi m \log n)$ edges always suffices. Moreover, removing $\Omega(\phi m \log n)$ edges is necessary in general. However, when restricted to *polynomial time* constructions of expander decompositions (which are necessary for most applications), all known results incur an additional polylogarithmic factor loss over this existential bound.

In this work, we give the *first* polynomial-time cut-expander and flow-expander decompositions with only an $\log^{o(1)} n$ factor extra loss. Before describing our results formally, we review the relevant definitions, describe the cut-and-recurse framework that achieves the existential bound, and explain why current polynomial time algorithms incur additional logarithmic losses.

Cut and Flow Expander Decomposition

Let $G = (V, E)$ be an undirected graph with n vertices and m edges. We say that G is a *ϕ-cut-expander* if for every subset $S \subseteq V$, we have

$$|E(S, V \setminus S)| \geq \phi \min\{\deg_G(S), \deg_G(V \setminus S)\},$$

where $\deg_G(S) = \sum_{u \in S} \deg(u)$.

Next, G is a *ϕ-flow-expander* if every multi-commodity demand respecting $\deg_G$ can be routed in G with congestion at most $1/\phi$. See Section 2 for detailed definitions.

A *ϕ-(cut/flow)-expander decomposition* is specified by an edge set $C \subseteq E$ such that each connected component of $G - C$ is a ϕ-(cut/flow)-expander. We say that a ϕ-(cut/flow)-expander decomposition has *overhead* γ if the number of removed edges $|C| \leq \gamma\phi m$. The overhead γ is the main quality measure.

Flow expansion is stronger requirement than cut expansion: every ϕ-flow-expander is a ϕ-cut-expander, while every ϕ-cut-expander is only guaranteed to be an $\Omega(\phi/\log n)$-flow expander by the well-known *flow–cut gap* [33]. Many flow-based applications – including all-or-nothing flow [11], edge-disjoint paths [20], flow vertex sparsifiers [17], and oblivious routing [39] – therefore require decompositions with *flow* expansion guarantees.

Benchmark: $\Omega(\log n)$ overhead is unavoidable

As mentioned above, already for *cut*-expander decompositions, there is an inherent $\Omega(\log n)$ lower bound on the overhead. In particular, for the n-vertex hypercube, standard isoperimetric bounds imply that if $|C| \leq m/2$, then some connected component of $G - C$ is not a $\Omega(1/\log n)$-cut expander [2]. Equivalently, setting $\phi = \Theta(1/\log n)$, any ϕ-cut-expander decomposition must have overhead $\gamma = \Omega(\log n)$. Since flow expansion implies cut expansion, the same lower bound applies to flow-expander decompositions as well. Thus, $\Theta(\log n)$ is the best possible target for γ.

Cut-and-recurse and polynomial-time limitations

The most basic expander decomposition procedure is to *cut and recurse*: if G is already a ϕ-cut-expander, return $C = \emptyset$. Otherwise, there exists some *ϕ-sparse* cut S satisfying

$$|E(S, V \setminus S)| \leq \phi \min\{\deg_G(S), \deg_G(V \setminus S)\},$$

and we recurse on $G[S]$ and $G[V \setminus S]$, returning $E(S, V \setminus S)$ and the cuts produced recursively.

So assuming that we can solve ϕ-sparse cut exactly, the above procedure, together with a standard charging argument,[1] gives overhead $\gamma = O(\log n)$. Together with the hypercube lower bound above, this overhead is graph-theoretically optimal (for appropriate ϕ).

In polynomial time however, one cannot find such sparse cuts exactly. In particular, when G is not a ϕ-cut-expander, using the best known $O(\sqrt{\log n})$-approximation to sparsest cut [4], we can only find an $O(\phi\sqrt{\log n})$-sparse cut. For flow expansion, the obstruction is even more fundamental: when G is not a ϕ-flow-expander, there may not even exist any $o(\phi \log n)$-sparse cut due to the flow–cut gap; and thus we can only find an $O(\phi \log n)$-sparse cut [33]. Plugging these extra losses into cut-and-recurse procedure yields, for any ϕ,

- a ϕ-cut-expander decomposition with overhead $O(\log^{1.5} n)$ in polynomial time, and
- a ϕ-flow-expander decomposition with overhead $O(\log^2 n)$ in polynomial time.

State of the art: the long-standing gap

Surprisingly, despite extensive research on expander decomposition and its extensions, no polynomial-time algorithm is known that improves upon the overhead bounds described above. In particular, it has remained open for over two decades whether one can match the $\Omega(\log n)$ existential benchmark in polynomial time, and more strongly, whether one can do so for *flow*-expansion.

Our result

Even though improving upon the $O(\sqrt{\log n})$-approximation to sparsest cut is a major open problem, and the $O(\log n)$ flow-cut gap loss may seem inherently unavoidable, we are able to circumvent these natural barriers and match the $\Omega(\log n)$ benchmark up to a $\log^{o(1)}(n)$ factor. This holds even for flow expanders. In particular, we show the following.

▶ **Theorem 1.1.** *There is a polynomial-time algorithm that, given an undirected graph $G = (V, E)$ with n vertices and m edges and parameter ϕ, returns an edge set $C \subseteq E$ such that each connected component of $G - C$ is a ϕ-flow-expander and $|C| \leq \phi\gamma m$, where $\gamma = O(\log(n) \exp(\sqrt{\log \log n}))$.*

Theorem 1.1 yields the first polynomial-time expander decomposition construction with optimal overhead up to a $\log^{o(1)} n$ factor. It improves the state-of-the-art by $\log^{0.5-o(1)} n$ and $\log^{1-o(1)} n$ factors for cut-expander decompositions and flow-expander decompositions, respectively. The result also extends to capacitated graphs, the terminal version, and more general node-weighting versions; see Theorem 3.1 for the theorem in full generality.

At a high level, we adapt the recent approach of graph clustering on top of a spreading-metric LP solution in [5] – developed to obtain improved approximations for cutwidth – to the expander decomposition setting, enabling a strictly better structural tradeoff than cut-and-recurse in polynomial time. A high-level intuition of the algorithm appears in Section 3.2.

2 Preliminaries

All logs are base 2. We work with an undirected graph $G = (V, E)$ where edges have unit capacity. For a vertex set $S \subseteq V$, let $\delta_G(S) := E(S, V \setminus S)$ be the set of edges crossing the cut $(S, V \setminus S)$.

[1] For each cut $(S, V \setminus S)$ charge $|E(S, V \setminus S)|$ to vertices on the smaller side so that each such vertex v is charged at most $\phi \deg(v)$. Each vertex can lie on the smaller side at most $\log n$ times, yielding total charge at most $2\phi(\log n)\, m$ and hence $\gamma = O(\log n)$.

Node-weighting

A *node-weighting* is a function $A : V \to \mathbb{R}_{\geq 0}$. For $S \subseteq V$ we use the shorthand

$$A(S) := \sum_{v \in S} A(v), \qquad |A| := A(V) = \sum_{v \in V} A(v).$$

We also use the restriction $A_S : V \to \mathbb{R}_{\geq 0}$ defined by $A_S(v) = A(v)$ if $v \in S$ and $A_S(v) = 0$ otherwise, so that $|A_S| = A(S)$. The degree node-weighting is $\deg_G : V \to \mathbb{Z}_{\geq 0}$ where $\deg_G(v)$ is the degree of v in G and $\deg_G(S) = \sum_{v \in S} \deg_G(v)$.

Multi-commodity Demands

A *demand* is a function $D : \binom{V}{2} \to \mathbb{R}_{\geq 0}$, where $D(u, v)$ specifies how much flow must be sent between the (unordered) pair $\{u, v\}$. We say that D *respects* A (or is A-*respecting*) if for every $x \in V$,

$$\sum_{y \in V \setminus \{x\}} D(x, y) \leq A(x).$$

We say that a demand D is *routable in G with congestion ρ* if there exists a feasible multi-commodity flow that routes each pair-demand $D(u, v)$ in G such that the total flow on every edge $e \in E$ is at most ρ (recall that edge capacities are 1). Given A, the A-*product demand* is the demand D_A defined by

$$D_A(u, v) := \frac{A(u)A(v)}{|A|} \qquad \text{for all } \{u, v\} \in \binom{V}{2}.$$

Note that D_A respects A, as $\sum_v D_A(u, v) \leq A(u)$ for each u.

Cut Expansion

We recall two notions of expansion with respect to a node-weighting A and parameter $\phi > 0$. The weighting A is ϕ-*cut-expanding in G* if for every nontrivial $\emptyset \subsetneq S \subsetneq V$,

$$|\delta_G(S)| \geq \phi \min\{A(S), A(V \setminus S)\}.$$

A set S is a ϕ-*sparse cut (with respect to A)* if it violates the above inequality.

Flow Expansion

The weighting A is ϕ-*flow-expanding in G* if every A-respecting demand is routable in G with congestion at most $1/\phi$. Fact 2.1 below shows that, upto a factor of 2, A is ϕ-flow-expanding in G iff the A-product demand D_A is routable in G with congestion $1/\phi$. That is, D_A is the "hardest" A-respecting demand. (The proof is given in Appendix A for completeness.)

▶ **Fact 2.1.** *If the A-product demand D_A is routable in G with congestion $1/\phi$, then A is $(\phi/2)$-flow-expanding in G.*

Expanders

Finally, G is a ϕ-*flow-expander* (resp., ϕ-*cut-expander*) if $\deg_G$ is ϕ-flow-expanding (resp., ϕ-cut-expanding) in G.

3 Expander Decomposition Algorithm

We now prove the following main result of the paper:

▶ **Theorem 3.1.** *For any undirected graph $G = (V, E)$, an integral node-weighting $A : V \to \mathbb{Z}_{\geq 0}$, and parameter ϕ, we can efficiently compute $C \subseteq E$ such that*

- $|C| \leq \phi |A| \log(|A|) \exp(O(\sqrt{\log \log |A|}))$, *and*
- *For each connected component U in $G - C$, $A \cap U$ is ϕ-flow-expanding in $G[U]$.*

This implies Theorem 1.1 by setting $A = \deg_G$.

Organization

In this section, we first describe the algorithm in Section 3.1 and give its high-level explanation in Section 3.2. Then, we prove that the algorithm is well-defined in Section 3.3. We formally prove the expansion guarantee and bound the cut size of the decomposition in Section 3.4.

3.1 Algorithm Description

Algorithm 1 uses two standard tools including the Concurrent Multi-Commodity Flow linear program and sparse neighborhood covers, described below.

Concurrent Flow Linear Program

Let D be the A-product demand where $D(u, v) = \frac{A(u)A(v)}{A(V)}$ for all u, v. We consider the following LPs which are dual to each other:

	(Primal)			(Dual)	
min	κ		max	$\sum_{u,v} D(u,v)\text{dist}_\ell(u,v)$	
s.t.	$\sum_{p \ni e} f_p \leq \kappa$	$\forall e \in E$	s.t.	$\sum_{e \in E} \ell_e \leq 1$	
	$\sum_{p:(u,v) \text{ paths}} f_p = D(u,v)$	$\forall u, v$		$\ell_e \geq 0$	$\forall e \in E$
	$f_p \geq 0$	$\forall \text{path } p$			

The primal LP is the problem of finding a multi-commodity flow that routes the A-product demand D with minimum congestion. The dual LP is a relaxation of finding sparse cuts w.r.t. A (see e.g., [45]). Both LPs can be solved in polynomial time (e.g., via a compact formulation for flow LP).

Sparse Neighborhood Cover

The second tool is the standard sparse neighborhood cover. We use the formulation from [5].

▶ **Lemma 3.2** (Lemma 6 of [5]). *For any weighted graph $G = (V, E)$, terminal set $T = \{v_1, \ldots, v_{|T|}\} \subseteq V$, radius parameter R, there is an efficient algorithm* CLUSTER(G, T, R) *that returns a collection $\mathcal{S} = \{S_1, \ldots, S_{|T|}\}$* of disjoint *vertex sets where*

1. ***(Cut size):*** $\sum_{S \in \mathcal{S}} |\delta(S)| \leq O(\log |T|) \frac{\sum_{e \in E} w_e}{R}$,
2. ***(Covering):*** $\bigcup_{S \in \mathcal{S}} S \supseteq \bigcup_{v \in T} B(v, R)$,
3. ***(Diameter):*** $S_i \subseteq B(v_i, 2R)$ *for each $v_i \in T$.*

The first condition bounds the total cut size of all clusters $S \in \mathcal{S}$. It is very crucial that the overhead factor is $O(\log |T|)$ and not just $O(\log n)$. The second shows that all clusters cover R-radius balls around every terminal. The third implies that the weak diameter of each cluster is at most $4R$. Now, consider the following algorithm.

Algorithm 1 $\mathrm{ED}(G = (V, E), A, \phi)$.

Define $\gamma = \exp(\sqrt{\log \log |A|})$ and $L = \left\lceil \log_\gamma \log |A| \right\rceil + 1$

1. Solve the concurrent flow LP. If $\kappa < 1/\phi$, then return. Else, let ℓ be the edge weights in G such that

$$\sum_{e \in E} \ell_e \le 1 \text{ and } \sum_{u,v} D(u,v)\mathrm{dist}_\ell(u,v) \ge 1/\phi.$$

2. Define the *radius threshold*

$$\Delta_i := 1/(4\phi|A|8^i) \text{ for each } i \ge 0.$$

Define the *mass threshold*

$$a_j := |A|/2^{\gamma^j} \text{ for each } j \ge -1.$$

3. **(Heavy-cluster case):** If there exists $x \in \mathrm{supp}(A)$ with $A(B(x, \Delta_0)) \ge |A|/2$, then perform sweep cut from $A(B(x, \Delta_0))$ and find a $O(\phi)$-sparse cut S' w.r.t. A. (See Section 4 for the precise algorithm) and return

$$E(S', V - S') \cup \mathrm{ED}(G[S'], A_{S'}, \phi) \cup \mathrm{ED}(G[V - S'], A_{V-S'}, \phi).$$

4. For each $x \in \mathrm{supp}(A)$, define the *radius scale* of x as the smallest $i_x \ge 1$ where

$$\log \frac{|A|}{A(B(x, \Delta_i))} \le \gamma \cdot \log \frac{|A|}{A(B(x, \Delta_{i-1}))}.$$

The *mass scale* j_x is such that

$$A(B(x, \Delta_{i_x})) \in (a_{j_x}, a_{j_x - 1}].$$

We have $i_x \in [1, L]$ and $j_x \in [1, L]$ by Propositions 3.4 and 3.5

5. For each radius and mass scale (i, j), let $V_{i,j} = \{x \in \mathrm{supp}(A) \mid i_x = i \text{ and } j_x = j\}$. Let

$$(i^*, j^*) = \arg\max_{(i,j)} A(V_{i,j}).$$

6. Let $N \subseteq V_{i^*,j^*}$ be a *net* obtained by a maximal packing of balls of radius Δ_{i^*} around vertices in V_{i^*,j^*}. That is, $\{B(x, \Delta_{i^*})\}_{x \in N}$ are disjoint and we have $V_{i^*,j^*} \subseteq \bigcup_{x \in N} B(x, 2\Delta_{i^*})$.
7. Compute $\mathcal{S} = \textsc{cluster}(G, T := N, R := 2\Delta_{i^*})$.
8. **(Balanced case):** Return

$$\bigcup_{S \in \mathcal{S}} \delta(S) \cup \bigcup_{S \in \mathcal{S}} \mathrm{ED}(G[S], A_S, \phi) \cup \mathrm{ED}(G[V - V(\mathcal{S})], A_{V - V(\mathcal{S})}, \phi).$$

3.2 High-level Explanation of the Algorithm

This section is informal: the goal is to explain how we adapt the spreading-metric clustering idea of [5] to beat the usual "cut-and-recurse" overhead in polynomial time.

From routability to a spreading metric

Algorithm 1 begins by solving the concurrent flow LP for the A-product demand $D(u,v) = A(u)A(v)/|A|$. If the optimum congestion $\kappa < 1/\phi$, then the A-product demand is routable with congestion $1/\phi$, and hence A is already $(\phi/2)$-flow-expanding by Fact 2.1; we can safely stop.

So the interesting case is when $\kappa \geq 1/\phi$. By LP duality, we obtain a nonnegative length function ℓ on edges such that

$$\sum_{e \in E} \ell_e \leq 1 \qquad \text{and} \qquad \sum_{u,v} D(u,v)\ \mathrm{dist}_\ell(u,v)\ \geq\ \frac{1}{\phi}.$$

Equivalently, if we sample a random pair (u,v) according to the A-product distribution, then

$$\mathbb{E}\big[\,\mathrm{dist}_\ell(u,v)\big]\ \gtrsim\ \frac{1}{\phi|A|}.$$

Thus, in the metric defined by the lengths ℓ, a typical pair of A-mass points is far apart: ℓ is a *spreading metric* for the demand.

Warm-up: why a single "good scale" would solve the problem

To see the key idea, imagine the following idealized situation. Suppose there exists a radius $\Delta^\star = \Theta(1/(\phi|A|))$ and a mass scale $a^\star$ such that for *every* vertex x,

$$a^\star\ \leq\ A(B(x,\Delta^\star))\ \leq\ A(B(x,4\Delta^\star))\ \leq\ 2a^\star, \tag{1}$$

where $B(x,r)$ denotes the ball of radius r around x in the ℓ-metric. Intuitively, (1) says that at the "right zoom level" $\Delta^\star$, every point sees about $a^\star$ units of A-mass nearby, and that increasing the radius by a constant factor does not suddenly swallow much more A-mass.

Now take a maximal packing net N of $\Delta^\star$-balls: the balls $\{B(x,\Delta^\star)\}_{x\in N}$ are disjoint, but the $2\Delta^\star$-balls cover V.[2] By disjointness and (1), the size of the net is at most

$$|N|\ \leq\ \frac{|A|}{a^\star}.$$

Following the clustering step of [5], run the neighborhood cover routine from Lemma 3.2 with terminals $T := N$ and radius $R := 2\Delta^\star$, obtaining disjoint clusters $\mathcal{S}$. By Lemma 3.2 the number of cut edges satisfies

$$\sum_{S\in\mathcal{S}} |\delta(S)|\ \leq\ O(\log|N|)\cdot\frac{\sum_e \ell_e}{2\Delta^\star}\ \lesssim\ \phi|A|\cdot\log\frac{|A|}{a^\star}.$$

Moreover, by the diameter guarantee of Lemma 3.2, each cluster $S\in\mathcal{S}$ lies inside some ball of radius $4\Delta^\star$ around its terminal, so by (1) it satisfies $A(S)\leq 2a^\star$. Therefore the total *recursive* cost on the clusters, under the standard potential $\phi\cdot A(\cdot)\log A(\cdot)$, is at most

$$\sum_{S\in\mathcal{S}} \phi\,|A_S|\,\log|A_S|\ \leq\ \sum_{S\in\mathcal{S}} \phi\,|A_S|\,\log(2a^\star)\ \lesssim\ \phi|A|\,\log a^\star.$$

[2] The factor 2 for covering the whole V here is the crucial reason why our end result suffers the $\exp(\sqrt{\log\log|A|})$ factor and does not achieve $O(\log\log|A|)$. The factor 2 is tight; there exists a metric (the projective plane) where, for every $R' < 2\Delta^\star$, the R'-balls can cover only a square root fraction of the metric.

Putting the two contributions together yields the telescoping expression

$$\phi|A|\log\frac{|A|}{a^\star} \;+\; \phi|A|\log a^\star \;=\; \phi|A|\log|A|.$$

This is the guiding principle of the whole algorithm:

We want one clustering step whose boundary cost is $\log(\#\textit{clusters}) \approx \log(|A|/a)$*, while ensuring each cluster has mass* $\approx a$*, so that recursion only pays* $\log a$*.*

Reality: different vertices have different good scales, so we "vote" over scales

As in [5], the main difficulty is that, in general, there is no single radius $\Delta^\star$ that satisfies (1) for all x. Instead, Algorithm 1 discretizes radii and masses into L scales: for $0 \le i \le L$ and $-1 \le j \le L$,

$$\Delta_i := \frac{1}{4\phi|A|\,8^i}, \qquad a_j := \frac{|A|}{2^{\gamma^j}}, \tag{2}$$

where $\gamma = \exp(\sqrt{\log\log|A|})$ and $L = \lceil \log_\gamma \log|A| \rceil + 1 = O(\sqrt{\log\log|A|})$. [3]

For a fixed vertex x, consider the growth curve of ball masses as we zoom in:

$$A(B(x,\Delta_0)),\; A(B(x,\Delta_1)),\; \ldots,\; A(B(x,\Delta_L)).$$

Since A is integral, $A(B(x,\Delta_L)) \ge 1$ always. On the other hand, if already $A(B(x,\Delta_0)) \ge |A|/2$, then x lies inside a very large "heavy" ball, which is handled separately in Step 3 (we discuss this case below). So in the balanced regime we may assume $A(B(x,\Delta_0)) \le |A|/2$ for all $x \in \mathrm{supp}(A)$.

Now define the *radius scale* i_x of x to be the *first* index where the ball stops shrinking "too fast" in a logarithmic sense:

$$\log\Big(\frac{|A|}{A(B(x,\Delta_{i_x}))}\Big) \;\le\; \gamma\cdot\log\Big(\frac{|A|}{A(B(x,\Delta_{i_x-1}))}\Big).$$

Intuitively, we keep zooming in until the "difficulty measure" $\log(|A|/A(B(x,\Delta_i)))$ fails to increase by a factor γ in one step. Such an index must exist (and indeed $i_x \le L$), because if the measure increased by a factor γ at *every* step, then after L steps it would exceed $\log|A|$, forcing $A(B(x,\Delta_L)) < 1$, which is impossible.

▶ **Example 3.3.** Suppose that as we zoom in by a factor 8 in radius, the A-mass near x shrinks like

$$A(B(x,\Delta_0)) \approx |A|/2, \qquad A(B(x,\Delta_1)) \approx |A|/2^{\gamma^{10}}, \qquad A(B(x,\Delta_2)) \approx |A|/2^{\gamma^{11}}.$$

Then $\log(|A|/A(B(x,\Delta_1)))$ is much larger than $\log(|A|/A(B(x,\Delta_0)))$ by a γ^{10} factor, but the increase from Δ_1 to Δ_2 is mild (only a γ factor). Then, $i_x = 2$ is the first such "stabilization" scale.

Next define the *mass scale* j_x by bucketing the mass at the stabilization radius:

$$A(B(x,\Delta_{i_x})) \in (a_{j_x},\, a_{j_x-1}].$$

Observe that indeed $1 \le j_x \le L$ as calculated in Proposition 3.5.

[3] These parameters are chosen to optimize the various tradeoffs that arise in the analysis later.

So each $x \in \mathrm{supp}(A)$ chooses one pair $(i_x, j_x) \in [L] \times [L]$. We now let the vertices "vote" for their chosen pair: for each (i,j), define $V_{i,j} := \{x \in \mathrm{supp}(A) \mid i_x = i,\ j_x = j\}$, and pick

$$(i^*, j^*) := \arg\max_{(i,j)} A(V_{i,j}).$$

Since there are only L^2 choices, this guarantees a large consensus class

$$A(V_{i^*,j^*}) \ \geq\ \frac{|A|}{L^2}. \tag{3}$$

You should think of V_{i^*,j^*} as a large set of vertices that agree on (1) the same "good zoom level" Δ_{i^*}, and (2) the same local mass scale $\approx a_{j^*}$.

The clustering step at the consensus scale

We now rerun the warm-up argument, but only using terminals from the consensus class. Let $N \subseteq V_{i^*,j^*}$ be a maximal packing net of Δ_{i^*}-balls. Disjointness and the definition of j^* imply

$$|N| \leq |A|/a_{j^*}.$$

We call $\textsc{cluster}(G, T := N, R := 2\Delta_{i^*})$ and obtain disjoint clusters $\mathcal{S}$.

Two properties make this step useful.

(i) We cut around a nontrivial amount of A-mass. Because N is maximal, the $2\Delta_{i^*}$-balls around N cover V_{i^*,j^*}, and the covering guarantee of Lemma 3.2 then implies that the union of clusters $V(\mathcal{S})$ contains all of V_{i^*,j^*}. By (3), this means the recursion peels off at least $|A|/L^2$ mass in one shot.

(ii) Each cluster is "small" in A-mass. By the diameter guarantee of Lemma 3.2, each cluster $S \in \mathcal{S}$ lies inside a ball of radius $4\Delta_{i^*}$ around its terminal $v \in N$. Since v has radius scale i^*, the mass of a $(\Theta(\Delta_{i^*}))$-ball around v cannot jump too much when we expand the radius by a constant factor; combined with the fact that $A(B(v, \Delta_{i^*})) \approx a_{j^*}$ (by the definition of j^*), this yields an upper bound of the form

$$A(S) \ \lesssim\ a_{j^*-O(1)}.$$

(Section 3.4 makes this precise; the point here is that the stabilization rule defining i_x is exactly what prevents a cluster of weak diameter $O(\Delta_{i^*})$ from capturing much more than the local scale a_{j^*}.)

Boundary accounting. Finally, Lemma 3.2 bounds the total boundary of the clusters by

$$\sum_{S \in \mathcal{S}} |\delta(S)| \ \leq\ O(\log |N|) \cdot \frac{\sum_e \ell_e}{2\Delta_{i^*}} \ \approx\ \phi|A| \cdot \log |N| \leq \phi|A| \cdot \log \frac{|A|}{a_{j^*}}.$$

This is exactly the "log(#clusters)" term from the warm-up. Meanwhile, since each cluster has A-mass at most $\approx a_{j^*-O(1)}$, the recursion on these clusters only pays $\log a_{j^*}$ in the potential. Thus the same telescoping intuition from the ideal case survives:

$$\text{(boundary)} \ \approx\ \log \frac{|A|}{a} \qquad \text{and} \qquad \text{(recursion)} \ \approx\ \log a \quad \Longrightarrow \quad \text{total} \ \approx\ \log |A|.$$

The technical work in Sections 3.3 and 3.4 is to make the words "≈" precise and to keep the losses within $\log^{o(1)} |A|$ factors (coming from the L and γ discretization).

Why we need the heavy-cluster case

The only way the "zoom until stabilization" story can fail is if we start with a vertex x whose coarsest ball already contains a constant fraction of the total mass, $A(B(x, \Delta_0)) \geq |A|/2$. In this situation there is a dense core $K := B(x, \Delta_0)$ of small ℓ-diameter containing a large fraction of A. The dual constraint $\sum_{u,v} D(u, v) \operatorname{dist}_\ell(u, v) \geq 1/\phi$ then forces a significant amount of demand to cross any sweep cut that moves away from K. A Leighton-Rao sweep algorithm (formalized in Section 4) finds a cut $S' \supseteq K$ that is $O(\phi)$-sparse with respect to A. Since we only pay $O(1)$ overhead factor in this level, we can revert to the standard cut-and-recurse step on $G[S']$ and $G[V \setminus S']$. This is exactly Step 3 of Algorithm 1.

Why Seymour's telescoping trick does not directly apply

Readers familiar with telescoping-volume arguments may wonder whether one can do even better, for example by charging each level by $\log(\mathrm{vol}_\ell(\text{parent})/\mathrm{vol}_\ell(\text{child}))$. The obstruction is that our metric ℓ is *not fixed across recursion*. When we recurse on an induced subgraph $G[S]$, we re-solve the flow LP inside $G[S]$, producing a new dual metric ℓ_S that can be essentially unrelated to the restriction of ℓ. Therefore, any "ℓ-volume" potential need not be consistent: a set that looks small in the parent metric can become large again under the new metric, preventing a clean telescoping argument.

Our approach avoids this instability by telescoping with a quantity that is invariant under recursion: the true A-mass. We explicitly choose clusters so that the boundary cost depends on $\log(|A|/a)$ (how many clusters we create), while the recursive cost depends on $\log a$ (how large clusters are in true mass). Because $A(\cdot)$ does not change when we recurse, these logs add up cleanly to $\log |A|$.

3.3 Validity of the Algorithm

First, we show that indeed the radius scale of each x is $i_x < L$. The idea is, otherwise, $A(B(x, \Delta_{i+1}))$ would shrink rapidly compared to $A(B(x, \Delta_i))$. But even the smallest ball $B(x, \Delta_L)$ has $A(B(x, \Delta_L)) \geq 1$. So, this can happen only when $A(B(x, \Delta_0)) > |A|/2$. But this contradicts that there is no heavy cluster from Step 3.

▶ **Proposition 3.4.** *For any $x \in \mathrm{supp}(A)$, $1 \leq i_x \leq L$.*

Proof. Suppose that $i_x > L$. Then, for all $1 \leq i \leq L$, $\log \frac{|A|}{A(B(x,\Delta_i))} > \gamma \cdot \log \frac{|A|}{A(B(x,\Delta_{i-1}))}$. Thus,

$$\log \frac{|A|}{A(B(x, \Delta_L))} \geq \gamma^L \log \frac{|A|}{A(B(x, \Delta_0))}.$$

We have $A(B(x, \Delta_L)) \geq 1$ as A is integral and $A(B(x, \Delta_0)) \leq |A|/2$, otherwise the algorithm would be in the heavy-cluster case. So,

$$\log |A| \geq \log \frac{|A|}{A(B(x, \Delta_L))} \text{ and } \log \frac{|A|}{A(B(x, \Delta_0))} \geq 1$$

and, hence,

$$\log |A| \geq \gamma^L.$$

But, the choice of γ and L, we have $\gamma^L > \log |A|$ which is a contradiction. ◀

▶ **Proposition 3.5.** *For any $x \in \mathrm{supp}(A)$, $1 \le j_x \le L$.*

Proof. We have

$$a_L < 1 \le A(B(x, \Delta_{L-1})) \le A(B(x, \Delta_0)) \le |A|/2 = a_0$$

because, again, A is integral and there is no heavy cluster in Step 3. Since we assign the mass scale j_x to x if $A(B(x, \Delta_{i_x})) \in (a_{j_x}, a_{j_x-1}]$, the claim follows. ◀

3.4 Correctness

We verify that the expansion guarantee in each component of $G - C$.

▶ **Proposition 3.6.** *For each connected component U in $G-C$, $A \cap U$ is $(\phi/2)$-flow-expanding in $G[U]$.*

Proof. When the optimal solution of the LP is $\kappa < 1/\phi$. This means that the A-product demand is routable in G with congestion less than $1/\phi$. So, A is a $\phi/2$-flow-expanding in G. The claim follows from applying this argument on each induced subgraph in the recursion. ◀

The next two lemmas bound the cut size (excluding the recursion). The heavy-cluster case is easy and we obtain an $O(1)$-approximate sparsest cut by closely following the technique by Leighton and Rao [33]. Thus, we defer this standard proof to Section 4.

▶ **Lemma 3.7.** *$|\delta(S')| \le 12\phi \min\{A(S'), A(V - S')\}$ and S' can be found efficiently.*

In the balanced case, we bound the cut size as follows.

▶ **Lemma 3.8.** *$\sum_{S \in \mathcal{S}} |\delta(S)| \le c_0 \phi 8^L \gamma^2 \cdot |A| \log \frac{|A|}{a_{j^*-2}}$ for some constant c_0.*

Proof. For each $x \in N$, we have $A(B(x, \Delta_{i^*})) > a_{j^*}$ since x has radius scale i^* and mass scale j^*. But the Δ_{i^*}-radius balls around each $x \in N$ are disjoint. So $|N| \le |A|/a_{j^*}$. As from definition $\log(|A|/a_j) = \gamma^j$ for all j, we have the bound

$$\log |N| \le \log \frac{|A|}{a_{j^*}} = \gamma^2 \log \frac{|A|}{a_{j^*-2}}. \tag{4}$$

By the cut size guarantee of CLUSTER from Lemma 3.2, we have that

$$\begin{aligned} \sum_{S \in \mathcal{S}_{(i^*, j^*)}} |\delta(S)| &= O(\log |N|) \frac{\sum_{e \in E} \ell_e}{2\Delta_{i^*}} \\ &\le c_0 \phi 8^L \gamma^2 \cdot |A| \log \frac{|A|}{a_{j^*-2}} \end{aligned}$$

for some large enough constant c_0. Here, in the second line we used that $\sum_{e \in E} \ell_e \le 1$, the bound in (4) and that $\Delta_{i^*} = 1/(4\phi |A| 8^{i^*}) \ge 1/(4\phi |A| 8^L)$. ◀

The next two lemmas are needed for our the induction proof.

▶ **Lemma 3.9.** *For each cluster $S \in \mathcal{S}$, $|A_S| \le a_{j^*-2}$*

Proof. By the diameter bound of CLUSTER from Lemma 3.2, $S \subseteq B(x, 4\Delta_{i^*}) \subseteq B(x, \Delta_{i^*-1})$ for some $x \in N$. By the definition of radius scale, we have

$$\frac{|A|}{A(B(x, \Delta_{i^*}))} \le \left(\frac{|A|}{A(B(x, \Delta_{i^*-1}))} \right)^{\gamma}.$$

Thus,

$$\begin{aligned} A(B(x,\Delta_{i^*-1})) &\le |A|^{1-1/\gamma} A(B(x,\Delta_{i^*}))^{1/\gamma} \\ &\le |A|^{1-1/\gamma} a_{j^*-1}^{1/\gamma} \\ &= a_{j^*-2}. \end{aligned}$$

where the second line is because x has mass scale j^*. To see the last line, write $a = |A|$. Observe that

$$|A|^{1-1/\gamma} a_{j^*-1}^{1/\gamma} = a^{1-1/\gamma} \frac{a^{1/\gamma}}{2^{(\gamma^{j^*-1})/\gamma}} = \frac{a}{2^{\gamma^{j^*-2}}} = a_{j^*-2}. \qquad \blacktriangleleft$$

▶ **Lemma 3.10.** *$A(V(\mathcal{S})) \ge |A|/L^2$.*

Proof. By the covering property of CLUSTER from Lemma 3.2, and as the Algorithm invokes Lemma 3.2 with $R = 2\Delta_{i^*}$, we have that

$$V(\mathcal{S}) = \bigcup_{S\in\mathcal{S}} S \supseteq \bigcup_{v\in N} B(v, 2\Delta_{i^*}) \supseteq V_{i^*,j^*}$$

where the final inclusion follows as N is a maximal packing of balls of radius Δ_{i^*} centered at vertices in V_{i^*,j^*}. The claim now follows because $A(V_{i^*,j^*}) \ge |A|/L^2$ by the choice of (i^*, j^*). $\blacktriangleleft$

Now, we are ready to conclude the bound on $|C|$. This would complete the proof.

▶ **Lemma 3.11.** *Let $C = ED(G, A, \phi)$. We have $|C| \le \phi\beta(|A|)|A|\log|A|$ where $\beta(|A|) = c_1 8^L L^2\gamma^2 = \exp(O(\sqrt{\log\log|A|})$ and c_1 is some constant.*

Proof. There are two cases.

Heavy-component case.
Assume w.l.o.g. that $A(S') \le |A|/2$. By Lemma 3.7, we have

$$\begin{aligned} |C| &\le 12\phi A(S') + |\mathrm{ED}(G[S'], A_{S'}, \phi)| + |\mathrm{ED}(G[V-S'], A_{V-S'}, \phi)| \\ &\le 12\phi|A_{S'}| + \phi\beta(|A|)|A_{S'}|(\log|A| - 1) + \phi\beta(|A|)|A_{V-S'}|\log|A| \\ &\le \phi\beta(|A|)|A_{S'}|\log|A| + \phi\beta(|A|)|A_{V-S'}|\log|A| \\ &= \phi\beta(|A|)|A|\log|A|. \end{aligned}$$

The second inequality is because $A(S') \le |A|/2$ and the third is because $12 \le \beta(|A|)$.

Balanced case.
We have

$$\begin{aligned} |C| &\le c_0\phi 8^L\gamma^2 \cdot |A|\log\frac{|A|}{a_{j^*-2}} + \sum_{S\in\mathcal{S}} |\mathrm{ED}(G[S], A_S, \phi)| + |\mathrm{ED}(G[V-V(\mathcal{S})], A_{V-V(\mathcal{S})}, \phi)| \\ &\le c_0\phi L^2 8^L\gamma^2 \cdot |A_{V(\mathcal{S})}|\log\frac{|A|}{a_{j^*-2}} + \sum_{S\in\mathcal{S}} \phi\beta(|A_S|)|A_S|\log|A_S| + \phi\beta(|A_{V-V(\mathcal{S})}|)|A_{V-V(\mathcal{S})}|\log|A_{V-V(\mathcal{S})}| \\ &\le \phi\beta(|A|)|A_{V(\mathcal{S})}|\log\frac{|A|}{a_{j^*-2}} + \phi\beta(|A|)|A_{V(\mathcal{S})}|\log a_{j^*-2} + \phi\beta(|A|)|A_{V-V(\mathcal{S})}|\log|A| \\ &= \phi\beta(|A|)|A|\log|A|. \end{aligned}$$

The first line is by Lemma 3.8. The second line is by Lemma 3.10 and the induction hypothesis. The third line is because $\beta(|A|) = c_1 8^L L^2\gamma^2 \ge c_0 L^2 8^L\gamma^2$ and Lemma 3.9. $\blacktriangleleft$

4 Heavy-Component Case

Throughout this section, $\mathrm{diam}_\ell(K) := \max_{u,v\in K} \mathrm{dist}_\ell(u,v)$ denotes the (strong) diameter of K in the shortest-path metric induced by ℓ.

Observe that the lemma below implies Lemma 3.7.

▶ **Lemma 4.1.** *Let $G=(V,E)$ be a graph with edge length ℓ and A be a node-weighting. Suppose that $\sum_{e\in E}\ell_e \le 1$ and $\sum_{u,v} D(u,v)dist_\ell(u,v) \ge 1/\phi$. Let $K\subseteq V$ be a set where $A(K)\ge A(V)/3$ and diameter $\mathrm{diam}_\ell(K)\le \frac{1}{4\phi|A|}$. Then, there is an efficient algorithm that finds a set $S'\supseteq K$ where*

$$|\delta(S')| \le 12\phi \min\{A(S'), A(V-S')\}.$$

Algorithm Sweep Cut

Define $\pi(v) := \mathrm{dist}_\ell(v,K)$ and sort vertices so that $\pi(v_1)\ge\pi(v_2)\ge\cdots\ge\pi(v_n)$. Let $S_k=\{v_1,\dots,v_k\}$ Return

$$S' \leftarrow \arg\min_{1\le k<n} \frac{|\delta(S_k)|}{D(S_k, V-S_k)}$$

where $D(S,T)=\sum_{s\in S,t\in T} D(s,t)$.

Analysis

Since D respects A, we have

$$\frac{|\delta(S_k)|}{\min\{A(S_k),A(V-S_k)\}} \le \frac{|\delta(S_k)|}{D(S_k,V-S_k)}.$$

So, it suffices to show

$$\min_{1\le k<n} \frac{|\delta(S_k)|}{D(S_k,V-S_k)} \le 12\phi.$$

We have

$$\begin{aligned}
\min_{1\le k<n} \frac{|\delta(S_k)|}{D(S_k,V-S_k)} &= \min_{1\le k<n} \frac{|\delta(S_k)|\cdot|\pi(v_k)-\pi(v_{k+1})|}{D(S_k,V-S_k)\cdot|\pi(v_k)-\pi(v_{k+1})|} \\
&\le \frac{\sum_{k<n}|\delta(S_k)|\cdot|\pi(v_k)-\pi(v_{k+1})|}{\sum_{k<n} D(S_k,V-S_k)\cdot|\pi(v_k)-\pi(v_{k+1})|} \\
&= \frac{\sum_{(u,v)\in E}|\pi(u)-\pi(v)|}{\sum_{u,v} D(u,v)\cdot|\pi(u)-\pi(v)|}. \qquad (5)
\end{aligned}$$

To see the last equality, for each $(u,v)=(v_i,v_j)$, its total contribution to $\sum_{k<n}|\delta(S_k)|\cdot|\pi(v_k)-\pi(v_{k+1})|$ is exactly

$$|\pi(v_i)-\pi(v_{i+1})|+\cdots+|\pi(v_{j-1})-\pi(v_j)| = |\pi(v_i)-\pi(v_j)|.$$

For each demand pair $D(v_i,v_j)$, its total contribution to $\sum_{k<n} D(S_k,V-S_k)\cdot|\pi(v_k)-\pi(v_{k+1})|$ is exactly

$$D(v_i,v_j)|\pi(v_i)-\pi(v_{i+1})|+\cdots+D(v_i,v_j)|\pi(v_{j-1})-\pi(v_j)| = D(v_i,v_j)|\pi(v_i)-\pi(v_j)|.$$

Numerator is ≤ 1

We bound each term in the sum. For each edge (u, v), we have

$$|\pi(u) - \pi(v)| = |\mathrm{dist}_\ell(u, K) - \mathrm{dist}_\ell(v, K)| \leq \mathrm{dist}_\ell(u, v) \leq \ell_{uv}$$

So

$$\sum_{e=(u,v)\in E} |\pi(u) - \pi(v)| \leq \sum_{e\in E} \ell_e \leq 1.$$

Denominator $\geq 1/12\phi$

Since $\pi(v) = 0$ for all $v \in K$,

$$\sum_{u,v} D(u, v) \cdot |\pi(u) - \pi(v)| \geq \sum_u \pi(u) D(u, K) \geq \frac{1}{3} \sum_u \pi(u) A(u). \tag{6}$$

where the last inequality is by the product structure of D: $D(u, K) = A(u)\frac{A(K)}{|A|} \geq \frac{1}{3}A(u)$. Next, we have

$$\begin{aligned}
1/\phi &\leq \sum_{u,v} D(u, v)\mathrm{dist}_\ell(u, v) \\
&\leq \sum_{u,v} D(u, v)(\mathrm{dist}_\ell(u, K) + \mathrm{diam}_\ell(K) + \mathrm{dist}_\ell(K, v)) \\
&\leq 2\sum_u \pi(u)A(u) + \sum_{u,v} D(u, v) \cdot \frac{1}{4\phi|A|}. \qquad (\mathrm{diam}_\ell(K) \leq \tfrac{1}{4\phi|A|}, \textstyle\sum_v D(u, v) \leq A(u))
\end{aligned}$$

As $\sum_{u,v} D(u, v) \cdot \frac{1}{4\phi|A|} = 1/4\phi$, plugging this above and rearranging gives

$$\frac{1}{4\phi} \leq \sum_u \pi(u)A(u).$$

Together with (6) this implies that denominator in (5) is at least $1/12\phi$. Thus we have that

$$\min_{1\leq k<n} \frac{|\delta(S_k)|}{D(S_k, V - S_k)} \leq 12\phi,$$

which completes the proof of the heavy-component case.

5 Open Questions

All-or-nothing Flow: From $\log^2 k$ to $\log^{1+o(1)} k$?

In the *all-or-nothing flow* problem [12], we are given an undirected graph $G = (V, E)$ with unit edge capacities and a set of k demand pairs $P = \{(s_1, t_1), \ldots, (s_k, t_k)\}$. The goal is to choose a subset $P' \subseteq P$ and, for each $(s_i, t_i) \in P'$, route one unit of (splittable) flow from s_i to t_i so that the total load on every edge is at most 1. Equivalently, we seek a maximum-cardinality subset of pairs that can be routed simultaneously with congestion 1.

This problem can be viewed as a relaxation of the *edge-disjoint paths* (EDP) problem, where each selected commodity must be routed on a *single path*. All-or-nothing flow was introduced as a "nice" intermediate model that admits polylogarithmic approximation guarantees, whereas EDP with congestion 1 appears substantially harder even in very restricted graph classes [19].

The current best approximation ratio for all-or-nothing flow in general graphs is $O(\log^2 k)$ [11]. A key bottleneck is the $O(\log^2 k)$ *overhead* incurred by the *well-linked decomposition* for all-or-nothing flow. Informally, it repeatedly finds sparse cuts and recurses (in the spirit of expander decomposition), and the loss can be viewed as the product of (i) an $O(\log k)$ flow–cut gap (when there are $\Theta(k)$ terminals) and (ii) an $O(\log k)$ recursion depth.

Our new flow-expander decomposition suggests that one might be able to *telescope* the loss across levels of recursion and reduce the total overhead to $\log^{1+o(1)} k$. However, the connection is not black-box: the well-linked decomposition used for all-or-nothing flow interleaves *concurrent* multicommodity flow computations with *maximum-throughput* multicommodity flow steps, whereas flow-expander decomposition mostly only needs concurrent-flow computation at each recursive call.

Can we construct the well-linked decomposition of [11] with $\log^{1+o(1)} k$ overhead using our technique? If so, it would immediately imply the following.

▶ **Conjecture 5.1.** *There is a polynomial-time* $(\log^{1+o(1)} k)$*-approximation algorithm for the all-or-nothing flow problem.*

Tree sparsifiers

Tree cut/flow sparsifiers can be viewed as a hierarchical analogue of expander decomposition and underlie many routing and graph-optimization applications, including oblivious routing [39], online multicut [3], near-linear-time approximate maximum flow [41, 38], almost-linear-time minimum-cost flow [44], and dynamic graph algorithms for connectivity and more [25].

For disjoint sets $S, T \subseteq V$, let $\mathrm{mincut}_G(S,T)$ be the value of a minimum cut separating S from T in G. A *tree cut sparsifier* for G with quality γ is a tree T (with edge capacities and $V(T) \supseteq V$) such that for all disjoint $S, T \subseteq V$,

$$\mathrm{mincut}_G(S,T) \;\le\; \mathrm{mincut}_T(S,T) \;\le\; \gamma \cdot \mathrm{mincut}_G(S,T).$$

A *tree flow sparsifier* for G with quality γ is a tree T such that for every $\deg_G$-respecting demand matrix D (i.e., $\sum_u D(v,u) \le \deg_G(v)$ for all v),

- if D is routable in G with congestion 1, then D is routable in T with congestion 1, and
- if D is routable in T with congestion 1, then D is routable in G with congestion at most γ.

As with cut vs. flow expansion, tree flow sparsifiers are strictly stronger: every tree flow sparsifier of quality γ is also a tree cut sparsifier of quality γ, whereas a tree cut sparsifier of quality γ only implies a tree flow sparsifier with an additional $O(\log n)$ loss in general.

The qualitative picture here mirrors expander decomposition. There is an $\Omega(\log n)$ graph-theoretic lower bound (already for tree cut sparsifiers, e.g. on grids) [35, 6]. Existentially, this barrier can be matched up to lower-order factors: tree cut sparsifiers of quality $O(\log n \log\log n)$ exist for every graph [40].

However, all known *polynomial-time* constructions lose an additional polylogarithmic factor beyond the existential bound: the best current algorithms achieve roughly $O(\log^{1.5} n \log\log n)$ for tree cut sparsifiers [40] and $O(\log^2 n \log\log n)$ for tree flow sparsifiers [28]. Recent work [41, 29, 1] focused on faster construction time but did *not* lead to improved quality.

Given that our work essentially removes the analogous extra loss for flow-expander decompositions, it is natural to ask whether the same is possible for tree flow sparsifiers.

▶ **Conjecture 5.2.** *There is a polynomial-time algorithm that constructs a tree flow sparsifier with quality* $\log^{1+o(1)} n$ *for every* n*-vertex undirected graph.*

Vertex and directed expander decompositions

Our results concern edge-based decompositions in undirected graphs. A very natural next step is to ask whether the same near-optimal overhead guarantees extend to (i) *vertex*-expander decompositions as defined in [34] and (ii) *directed*-expander decompositions as defined in [7, 22]. Can we obtain $\log^{1+o(1)} n$-overhead analogues in these settings as well?

References

1 Daniel Agassy, Dani Dorfman, and Haim Kaplan. Improved tree sparsifiers in near-linear time. *arXiv preprint arXiv:2511.06574*, 2025. doi:10.48550/arXiv.2511.06574.

2 Vedat Levi Alev, Nima Anari, Lap Chi Lau, and Shayan Oveis Gharan. Graph Clustering using Effective Resistance. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 41:1–41:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2018.41.

3 Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. A general approach to online network optimization problems. *ACM Transactions on Algorithms (TALG)*, 2(4):640–660, 2006. doi:10.1145/1198513.1198522.

4 Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):1–37, 2009. doi:10.1145/1502793.1502794.

5 Nikhil Bansal, Dor Katzelnick, and Roy Schwartz. On approximating cutwidth and pathwidth. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 713–729. IEEE, 2024. doi:10.1109/FOCS61266.2024.00051.

6 Yair Bartal and Stefano Leonardi. On-line routing in all-optical networks. In *International Colloquium on Automata, Languages, and Programming*, pages 516–526. Springer, 1997. doi:10.1007/3-540-63165-8_207.

7 Aaron Bernstein, Maximilian Probst Gutenberg, and Thatchaphol Saranurak. Deterministic decremental reachability, scc, and shortest paths via directed expanders and congestion balancing. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1123–1134. IEEE, 2020. doi:10.1109/FOCS46700.2020.00108.

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

9 Yi-Jun Chang and Thatchaphol Saranurak. Deterministic distributed expander decomposition and routing with applications in distributed derandomization. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 377–388. IEEE, 2020. doi:10.1109/FOCS46700.2020.00043.

10 Chandra Chekuri and Julia Chuzhoy. Polynomial bounds for the grid-minor theorem. *Journal of the ACM (JACM)*, 63(5):1–65, 2016. doi:10.1145/2820609.

11 Chandra Chekuri, Sanjeev Khanna, and F Bruce Shepherd. Multicommodity flow, well-linked terminals, and routing problems. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 183–192, 2005. doi:10.1145/1060590.1060618.

12 Chandra Chekuri, Sanjeev Khanna, and F Bruce Shepherd. The all-or-nothing multicommodity flow problem. *SIAM Journal on Computing*, 42(4):1467–1493, 2013. doi:10.1137/100796820.

13 Daoyuan Chen, Simon Meierhans, Maximilian Probst Gutenberg, and Thatchaphol Saranurak. Parallel and distributed expander decomposition: Simple, fast, and near-optimal. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1705–1719. SIAM, 2025. doi:10.1137/1.9781611978322.53.

14 Li Chen, Rasmus Kyng, Yang Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. *Journal of the ACM*, 72(3):1–103, 2025. doi:10.1145/3728631.

15 Yu Chen, Michael Kapralov, Mikhail Makarov, and Davide Mazzali. On the Streaming Complexity of Expander Decomposition. In *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, volume 297 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 46:1–46:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.ICALP.2024.46.

16 Yu Chen, Sanjeev Khanna, and Huan Li. On weighted graph sparsification by linear sketching. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 474–485. IEEE, 2022. doi:10.1109/FOCS54457.2022.00052.

17 Julia Chuzhoy. On vertex sparsifiers with steiner nodes. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 673–688, 2012. doi:10.1145/2213977.2214039.

18 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 *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1158–1167. IEEE, 2020. doi:10.1109/FOCS46700.2020.00111.

19 Julia Chuzhoy, David HK Kim, and Rachit Nimavat. Almost polynomial hardness of node-disjoint paths in grids. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1220–1233, 2018.

20 Julia Chuzhoy and Shi Li. A polylogarithmic approximation algorithm for edge-disjoint paths with congestion 2. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 233–242. IEEE, 2012. doi:10.1109/FOCS.2012.54.

21 Arnold Filtser, Michael Kapralov, and Mikhail Makarov. Expander Decomposition in Dynamic Streams. In *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*, volume 251 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 50:1–50:13. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ITCS.2023.50.

22 Henry Fleischmann, George Z Li, and Jason Li. Improved directed expander decompositions. *arXiv preprint arXiv:2507.09729*, 2025. doi:10.48550/arXiv.2507.09729.

23 Oded Goldreich and Dana Ron. A sublinear bipartiteness tester for bounded degree graphs. In *Proceedings of the thirtieth annual ACM Symposium on Theory of Computing*, pages 289–298, 1998. doi:10.1145/276698.276767.

24 Gramoz Goranci, Monika Henzinger, Danupon Nanongkai, Thatchaphol Saranurak, Mikkel Thorup, and Christian Wulff-Nilsen. Fully dynamic exact edge connectivity in sublinear time. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 70–86. SIAM, 2023. doi:10.1137/1.9781611977554.CH3.

25 Gramoz Goranci, Harald Räcke, Thatchaphol Saranurak, and Zihan Tan. The expander hierarchy and its applications to dynamic graph algorithms. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2212–2228. SIAM, 2021. doi:10.1137/1.9781611976465.132.

26 Bernhard Haeupler, Yonggang Jiang, Yaowei Long, Thatchaphol Saranurak, and Shengzhe Wang. Parallel $(1+\epsilon)$-approximate multi-commodity mincost flow in almost optimal depth and work. *arXiv preprint arXiv:2510.20456*, 2025. doi:10.48550/arXiv.2510.20456.

27 Bernhard Haeupler, Harald Räcke, and Mohsen Ghaffari. Hop-constrained expander decompositions, oblivious routing, and distributed universal optimality. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1325–1338, 2022. doi:10.1145/3519935.3520026.

28 Chris Harrelson, Kirsten Hildrum, and Satish Rao. A polynomial-time tree decomposition to minimize congestion. In *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, pages 34–43, 2003. doi:10.1145/777412.777419.

29 Monika Henzinger, Robin Münk, and Harald Räcke. An improved quality hierarchical congestion approximator in near-linear time. *arXiv preprint arXiv:2511.03716*, 2025. doi:10.48550/arXiv.2511.03716.

30 Wenyu Jin, Xiaorui Sun, and Mikkel Thorup. Fully dynamic min-cut of superconstant size in subpolynomial time. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2999–3026. SIAM, 2024. doi:10.1137/1.9781611977912.107.

31 Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings: Good, bad and spectral. *Journal of the ACM (JACM)*, 51(3):497–515, 2004. doi:10.1145/990308.990313.

32 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 *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 217–226. SIAM, 2014. doi:10.1137/1.9781611973402.16.

33 Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM)*, 46(6):787–832, 1999. doi:10.1145/331524.331526.

34 Yaowei Long and Thatchaphol Saranurak. Near-optimal deterministic vertex-failure connectivity oracles. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1002–1010. IEEE, 2022. doi:10.1109/FOCS54457.2022.00098.

35 Bruce M Maggs, F Meyer auf der Heide, Berthold Vocking, and Matthias Westermann. Exploiting locality for data management in systems of limited bandwidth. In *Proceedings 38th Annual Symposium on Foundations of Computer Science*, pages 284–293. IEEE, 1997.

36 Danupon Nanongkai, Thatchaphol Saranurak, and Christian Wulff-Nilsen. Dynamic minimum spanning forest with subpolynomial worst-case update time. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 950–961. IEEE, 2017. doi:10.1109/FOCS.2017.92.

37 Mihai Patrascu and Mikkel Thorup. Planning for fast connectivity updates. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 263–271. IEEE, 2007. doi:10.1109/FOCS.2007.59.

38 Richard Peng. Approximate undirected maximum flows in $o(m\text{polylog}(n))$ time. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1862–1867. SIAM, 2016. doi:10.1137/1.9781611974331.CH130.

39 Harald Racke. Minimizing congestion in general networks. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pages 43–52. IEEE, 2002.

40 Harald Räcke and Chintan Shah. Improved guarantees for tree cut sparsifiers. In *European Symposium on Algorithms*, pages 774–785. Springer, 2014. doi:10.1007/978-3-662-44777-2_64.

41 Harald Räcke, Chintan Shah, and Hanjo Täubig. Computing cut-based hierarchical decompositions in almost linear time. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 227–238. SIAM, 2014. doi:10.1137/1.9781611973402.17.

42 Thatchaphol Saranurak and Di Wang. Expander decomposition and pruning: Faster, stronger, and simpler. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2616–2635. SIAM, 2019. doi:10.1137/1.9781611975482.162.

43 Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pages 81–90. ACM, 2004. doi:10.1145/1007352.1007372.

44 Jan Van Den Brand, Li Chen, Rasmus Kyng, Yang P Liu, Simon Meierhans, Maximilian Probst Gutenberg, and Sushant Sachdeva. Almost-linear time algorithms for decremental graphs: Min-cost flow and more via duality. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2010–2032. IEEE, 2024. doi:10.1109/FOCS61266.2024.00120.

45 Vijay V. Vazirani. *Approximation Algorithms*, volume 1 of *Algorithms and Combinatorics*. Springer, 2001. doi:10.1007/978-3-662-04565-7.

A Proof of Fact 2.1

Let D' be any A-respecting demand. Consider routing D' in the following complete graph G_A with capacities given by D_A as follows: for each unordered pair $\{x, y\}$ and each intermediate vertex $z \in V$, send an amount $D'(x, y) \cdot \frac{A(z)}{|A|}$ from x to z and an equal amount from z to y. This defines a feasible (fractional) two-hop routing.

Fix an (undirected) edge $\{x, z\}$ in G_A. The total flow sent on $\{x, z\}$ is at most

$$\sum_{y \in V \setminus \{x\}} D'(x, y) \cdot \frac{A(z)}{|A|} \leq A(x) \cdot \frac{A(z)}{|A|} = D_A(x, z),$$

and the same bound holds for the load contributed when x appears as the second endpoint. Thus every edge $\{u, v\}$ is used with congestion at most 2 compared to its capacity $D_A(u, v)$.

Therefore, D' is routable in G_A with congestion 2. Composing this routing with a congestion-$1/\phi$ routing of D_A in G yields a routing of D' in G with congestion $2/\phi$, proving that A is $(\phi/2)$-flow-expanding in G.