Length-Constrained Directed Expander Decomposition and Length-Constrained Vertex-Capacitated Flow Shortcuts

Bernhard Haeupler
INSAIT, Sofia University "St. Kliment Ohridski", Bulgaria
ETH Zürich, Switzerland

Yaowei Long
University of Michigan, Ann Arbor, MI, USA

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

Shengzhe Wang
ETH Zürich, Switzerland

Abstract

We show the existence of *length-constrained expander decomposition* in directed graphs and undirected vertex-capacitated graphs. Previously, its existence was shown only in undirected edge-capacitated graphs [24, 21]. Along the way, we prove the multi-commodity maxflow-mincut theorems for length-constrained expansion in both directed and undirected vertex-capacitated graphs. Based on our decomposition, we build a *length-constrained flow shortcut* for undirected vertex-capacitated graphs, which roughly speaking is a set of edges and vertices added to the graph so that every multi-commodity flow demand can be routed with approximately the same vertex-congestion *and* length, but all flow paths only contain few edges. This generalizes the shortcut for undirected edge-capacitated graphs from [20]. Length-constrained expander decomposition and flow shortcuts have been crucial in the recent algorithms in undirected edge-capacitated graphs [20, 23]. Our work thus serves as a foundation to generalize these concepts to directed and vertex-capacitated graphs.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases Length-Constrained Expander, Expander Decomposition, Shortcut

Digital Object Identifier 10.4230/LIPIcs.ESA.2025.107

Related Version *Full Version*: https://arxiv.org/abs/2503.23217

Funding Part of this work was done while at INSAIT, Sofia University "St. Kliment Ohridski", Bulgaria. This work was partially funded from the Ministry of Education and Science of Bulgaria (support for INSAIT, part of the Bulgarian National Roadmap for Research Infrastructure).
Bernhard Haeupler: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (ERC grant agreement 949272).
Thatchaphol Saranurak: Supported by NSF Grant CCF-2238138.
Shengzhe Wang: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (ERC grant agreement No 949272 and No 101171685).

1 Introduction

Expander decomposition found its early applications in property testing [17], clustering [32], and approximation algorithms [10] and, for the last two decades, has been the crucial ingredient in important developments of fast graph algorithms. This includes the first almost-linear time algorithms for spectral sparsifiers and Laplacian solvers [46], approximate

max flow [33, 45, 42], deterministic global min cut [35], exact max flow [12], as well as many almost-optimal dynamic algorithms for minimum spanning trees [41], shortest paths [14, 4], sparsifiers [5], k-edge-connectivity [30], minimum cuts [31, 16], and more [18]. Significant effort [43, 8, 13, 7, 37, 36, 27, 1, 19, 11] has focused on constructing expander decomposition itself. Below, we discuss two successful orthogonal generalizations of expander decomposition.

Vertex and Directed Expander Decomposition. In 2005, Chekuri, Khanna, and Shepherd [10] showed that the construction of expander decomposition in undirected edge-capacitated graphs naturally extends to work in undirected vertex-capacitated graphs and applies them for approximating all-or-nothing vertex-capacitated flow problems. Later, this was extended to directed graphs, an even more general setting [9].[1]

Since 2020, almost-linear time expander decomposition algorithms in these generalized settings have been developed [3, 39, 27, 47] and found impressive applications. For the vertex-capacitated ones, they were crucial for the fastest deterministic vertex connectivity algorithms [44, 40] and data structures for connectivity queries under vertex failures [39, 38, 29]. For the directed ones, they were used for dynamic algorithms in directed graphs [3] and the new combinatorial approaches for exact max flow [15, 2].

Length-Constrained Expander Decomposition and Flow Shortcuts. More recently, Haeupler, Räcke, and Ghaffari [24] introduced *length-constrained expanders* (LC-expanders). At a very high level, these are graphs such that any "reasonable" demand can be routed with low congestion *and* length. In contrast, normal expanders only guarantee low congestion. [24] constructed LC-expander decomposition and applied it to show universally optimal distributed algorithms. In general, LC-expander decomposition is much more effective for problems that simultaneously concern length and congestion.

Based on the new decomposition, [20] introduced the notion of *LC-flow shortcut*[2], a new kind of graph augmentation. Roughly speaking, an LC-flow shortcut is a set of edges and vertices added to the graph so that every multi-commodity flow demand can be routed with approximately the same congestion *and* length, but all flow paths only have a few edges. We use steps, which count the number of edges, to distinguish between edge lengths. This is formalized as follows (see Section 2 for background).

▶ **Definition 1** (Length-Constrained Flow Shortcut). *Given a graph $G = (V, E)$, we say an edge set E' (possibly with endpoints outside V) is a t-step flow shortcut of G with length slack λ and congestion slack κ if*

- *(Forward Mapping) for every demand D routable in G with congestion 1 and length h, D is routable in $G \cup E'$ with congestion 1, length λh, and maximum step t, and*
- *(Backward Mapping) for every demand D on $V(G)$ routable in $G \cup E'$ with congestion 1 and length h, D is routable in G with congestion κ and length h.*

In any undirected edge-capacitated graph, [20] showed, for any $\epsilon > 0$, the existence of a $O(1/\epsilon^2)$-step LC-flow shortcut E' of size $|E'| \leq O(n^{1+O(\epsilon)}\text{polylog}(n))$ with length slack $O(1/\epsilon^3)$ and congestion slack $n^{O(\epsilon)}$.[3] Combined with newly developed close-to-linear time LC-expander decomposition [21, 22], they also obtained a close-to-linear time construction for LC-flow shortcuts albeit with worse quality.

[1] In fact, expander decomposition was only implicit in [10, 9] as their definitions were specific to their applications. The purely graph-theoretic definition was later formalized in [3].

[2] It was called a *low-step flow emulator* in [20].

[3] The shortcut E' in [20] actually has $O(1/\varepsilon^4)$ length slack and $O(1/\varepsilon^2)$ maximum step, but this is only because they tried to ensure that all endpoints of E' are in V. Allowing endpoints outside V, one can replace their *router* with a star and improve the quality to be as we stated.

LC-flow shortcuts have led to significant further progress. This includes the first close-to-linear time constant-approximation algorithm for minimum cost multi-commodity flow [20]. The dynamic but weaker version of flow shortcuts was also the key object in the first deterministic dynamic constant-approximate distance oracle with $O(n^\epsilon)$ update time [23].

However, all applications of LC-expander decomposition until now are limited to undirected edge-capacitated graphs.

1.1 Our Results

To extend the reach of the expander decomposition paradigm further, the history above suggests the following research question:

Can we construct length-constrained expander decomposition and flow shortcuts beyond undirected edge-capacitated graphs?

Indeed, we answer this question affirmatively. In this paper, we focus on the existential results, but the arguments naturally give polynomial-time algorithms. For future work, we are working towards almost-linear-time constructions, which would lead to further applications for minimum cost (multi-commodity) flow in vertex-capacitated and directed graphs. Below, we discuss our contribution in more detail.

Length-Constrained Directed and Vertex Expander Decompositions. We formalize the notions of length-constrained expanders in directed graphs and in undirected vertex-capacitated graphs. Then, we show the existence of length-constrained expander decomposition in directed graphs (Theorem 10) and in undirected vertex-capacitated graphs (Theorem 11). Along the way, we also show that the definition of length-constrained expanders based on cuts is almost equivalent to the characterization based on multi-commodity flow (Theorems 9 and 12). This can be viewed as a version of the approximate multicommodity maxflow mincut theorem [34] but for length-constrained expansion in directed and vertex-capacitated graphs. While this part does not require technical novelty, it is an important foundation for our paper and, we believe, for future work using this concept.

Length-Constrained Vertex-Capacitated Flow Shortcuts. Our main technical contribution (Theorem 14) is to show that, for any undirected *vertex-capacitated* graph and any $\epsilon > 0$, there exists a $2^{O(\frac{1}{\epsilon})}$-step flow shortcut E' of size $|E'| = O(n^{1+O(\epsilon)}\text{polylog}(n))$ with length slack $O(1/\epsilon^3)$ and congestion slack $n^{O(\epsilon)}$. This generalizes the flow shortcut of [20] in undirected edge-capacitated graphs.

Our trade-off between size, length slack, and congestion slack matches the one of [20]. However, our step-bound is $2^{O(\frac{1}{\epsilon})}$ instead of $O(1/\epsilon^2)$. This is due to technical barriers unique to vertex-capacitated graphs, which also requires us to use very different analysis. We leave as a very interesting open problem if it is possible to obtain $\text{poly}(1/\epsilon)$ steps.

We note that obtaining similar LC-flow shortcuts on directed graphs is currently out of reach because it would give the breakthrough on *reachability shortcuts*. Given a graph $G = (V, E)$, an edge set E' is a t-step reachability shortcut of G if, for every pair of vertices $u, v \in V$, u can reach v in G if and only if u can reach v in $G \cup E'$ using at most t steps. Observe that an LC-flow shortcut in a directed graph is strictly stronger than a reachability shortcut. It is a major open problem whether there exists a $n^{o(1)}$-step reachability shortcut of size $n^{1+o(1)}$.[4]

[4] When endpoints of E' must be in V, [26, 28, 6] already showed that there is no $\Omega(n^{1/4})$-step reachability shortcut of size $O(n)$. The lower bounds extend to the shortcut of size $n^{1+\epsilon}$ with a worse step bound.

2 Preliminaries

We give some brief background on length-constrained expansion. A *demand* $D : V \times V \to \mathbb{R}_{\geq 0}$ assigns value to pair of vertices (u, v) and D is *h-length* is it assigns non-zero values only to vertex pairs of distance $\text{dist}_G(u, v) \leq h$ in G. A demand D is *routable* with congestion κ and length λ if there exists a multi-commodity flow routing D with congestion κ and length λ. D *respects* a *node-weighting* $A : V \to \mathbb{R}_{\geq 0}$ if for each vertex u, $\sum_v D(u, v) \leq A(u)$. Let $|A| = \sum_{u \in V} A(u)$. For any $s \geq 1$, A is *(h, s)-length ϕ-expanding in* G if every h-length A-respecting demand is routable with length hs and congestion $O(\frac{\log n}{\phi})$.[5] A length-constrained cut C assigns to each edge an integral length increase, and $G - C$ is the graph G applied with the length increase from cut C. We informally say that G is a *length-constrained expander (LC-expander)* if a node-weighting A whose support is the whole vertex set V is expanding in G. An (h, s)-length ϕ-expander decomposition for A is a length-constrained cut C such that A is (h, s)-length ϕ-expanding in $G - C$.

We use standard graph terminology, deferring details to the full version for further (standard) preliminaries for directed graphs and vertex-capacitated graphs.

2.1 Our Techniques

Next, we give a technical overview of our LC-flow shortcut on vertex-capacitated graphs. We will explain how the strategy used in [20] fails in our setting and how we overcome the obstacle. For simplicity, here we only consider graphs with *unit capacity*. Also, we only construct a slightly weaker notion of LC-flow shortcut in the sense that, it receives an additional length parameter h and the forward mapping only guarantees that every demand routable in G with length $h' \leq h$ and congestion 1 is routable in $G \cup E'$ with length λh, congestion 1 and step t.

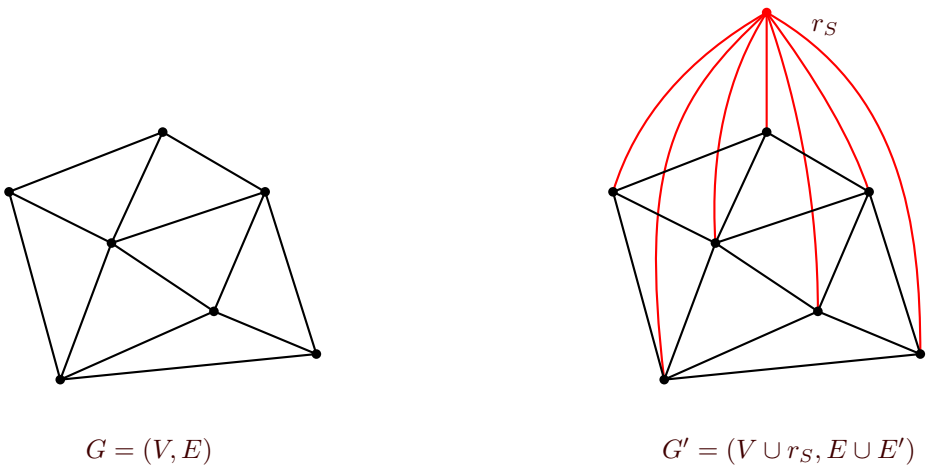

Figure 1 An LC-flow shortcut of a low-diameter LC-expander.

Warm-up: Shortcutting LC-expanders. Before explaining the obstacle, we first show how to shortcut an LC-vertex expander as a warm-up. Suppose that a node-weighting A is (h, s)-length ϕ-vertex expanding in G. Say, $A := \mathbb{1}_V$ (i.e., $A(v) = 1$ for all $v \in V$).

[5] Our definition in the paper (Definition 7) is actually cut-based. This almost-equivalent flow-based definition follows from Theorem 12 and is more convenient in this overview.

Suppose further that G has diameter at most h. In this case, our shortcut is simply a star S connecting each original vertex v to a Steiner vertex r_S with an (hs)-length $A(v)$-capacity edge. We can shortcut any feasible flow in G with 2 steps. An illustrative example is shown in Figure 1. The length slack is $O(s)$ since an h-length original feasible flow is mapped forward to a $(2hs)$-length feasible flow in the star. The congestion slack is $O(\log n/\phi)$ because any feasible flow in the star induces an h-length A-respecting demand. Since A is (h,s)-length ϕ-expanding in G, we route such demand in G with congestion $O(\log n/\phi)$ and without length increasing.

In general, the diameter can be large. Thus, we can construct a *sparse neighborhood cover* to decompose the graph into clusters with diameter h, such that (1) for each vertex v, there is a cluster containing all vertices within distance h/s from v, and (2) each vertex is inside $n^{O(1/s)}$ clusters. Then, we can construct a shortcut by adding an (hs)-edge-length star on each cluster. By a similar argument, we obtain a flow shortcut graph for (h/s)-length original flows with length slack $O(s^2)$, congestion slack $O(n^{O(1/s)}\log n/\phi)$ and step 2.

So far, when we build an LC-flow shortcut for an LC-expander, the vertex-capacitated setting presents no difficulties compared to the edge-capacitated setting, because the above simple approach works in both settings. However, the differences between the two settings arise when generalizing this approach to general graphs via expander hierarchies.

Previous Approach: Shortcutting General Graphs via Boundary-Linkedness. The key idea of [20] is to exploit a hierarchy of *boundary-linked* LC-expander decomposition, defined as follows. Let G be an edge-unit-capacity graph. Initialize the node-weight $A_0 = \deg_G$. For each level $0 \le i \le d$, compute a cut $C_{i+1} \subseteq E$ of size $|C_{i+1}| \approx \phi|A_i|$ such that $A_i + \deg_{C_{i+1}}$ is (h,s)-length ϕ-expanding in $G - C_{i+1}$ where $\deg_{C_{i+1}}(v)$ counts the number of C_{i+1}-edges incident to v.[6] The cut C_{i+1} is called the *boundary-linked* LC-expander decomposition for A_i because it gives a stronger expansion guarantee of $A_i + \deg_{C_{i+1}}$ instead of just A_i. Then, we set $A_{i+1} := \deg_{C_{i+1}}$ and continue to the next level $i+1$. By setting $\phi = 1/(n^{O(1/s)}n^{\varepsilon})$, we have that $d = O(1/\varepsilon)$ and $A_{d+1} = 0$.

From the above construction, we conclude that, for each i, $A_i + A_{i+1}$ is (h,s)-length ϕ-expanding in $G - C_{i+1}$. Therefore, as we have seen in the warm-up, we can add stars on the support of $A_i + A_{i+1}$ so that any flows routing h-length $(A_i + A_{i+1})$-respecting demands in $G - C_{i+1}$ can be shortcut.

Now consider a feasible h-length flow in G. The boundary-linkedness suggests a natural *bottom-up* shortcut scheme. For each flow path P, we can think of routing P's *head packet* and *tail packet* (initially at P's left and right endpoints, denoted by u_0 and v_0) to the same place via shortcuts. Take the head packet as an example. Start with $u_0 \in \mathrm{supp}(A_0) = V$. At each level $0 \le i \le d-1$, let u_{i+1} be the left endpoint of the first $P \cap C_{i+1}$-edge behind u_i. By definition, $u_{i+1} \in \mathrm{supp}(A_{i+1})$ and P's subpath between u_i and u_{i+1} is disjoint from C_{i+1}, so we can use star graphs at level i to route the head packet from u_i to u_{i+1} within 2 steps. In sum, each of the head and tail packets is routed from the bottom up until they reach $u_d, v_d \in \mathrm{supp}(A_d)$, and the top-level star graphs can route them together. The total number of steps is $O(d) = O(1/\epsilon)$[7].

[6] In the actual construction, C_{i+1} assigns fractional values to edges and is called a *moving cut*, defined in Section 3. Here, we assume C_{i+1} is a classic edge cut for simplicity.

[7] We note that the step bound in [20] is $O(1/\epsilon^2)$ because they used powers of expander graphs instead of star graphs to avoid creating vertices outside G, which brought another $O(1/\epsilon)$ factor.

The Obstacle from Vertex Cuts. The overall strategy of the above approach is to shortcut flow paths from a vertex of A_i to an endpoint of edges in C_{i+1}. This was possible since the boundary-linked expander decomposition guarantees that $A_i + \deg_{C_{i+1}}$ is expanding.

In the vertex-capacitated graph, however, the cut $C_{i+1} \subseteq V$ is now a vertex set. To follow the same strategy, we have two natural options. We shortcut flow from a vertex of A_i to either (1) a vertex in C_{i+1}, or (2) a neighbor of C_{i+1}.

In the first case, the strategy requires that $A_i + C_{i+1}$ is expanding in $G - C_{i+1}$. This is trivially impossible because C_{i+1} is not even in the graph $G - C_{i+1}$. In the second case, let $N(C_{i+1})$ denote the neighbors of C_{i+1} that are not in C_{i+1}. The strategy requires $A_i + N(C_{i+1})$ is expanding in $G - C_{i+1}$. However, possibly $N(C_{i+1})$ is very big and has size $|N(C_{i+1})| = \Omega(n|C_{i+1}|)$. It is unlikely that expander decomposition exists to guarantee the expansion of such a large node-weighting. Even if it exists, we would set $A_{i+1} = N(C_{i+1})$ and, hence, we cannot guarantee $|A_{i+1}| \ll |A_i|$. So the number of levels of the hierarchy is unbounded.

In either option, this overall strategy fails in the vertex-capacitated graphs. At a very high level, this is because edges have two endpoints while vertices may have an unbounded number of neighbors.

Our Approach: Top-Down Analysis without Boundary-linkedness. We construct a similar hierarchy of LC-vertex expander decomposition *without* boundary-linkedness as follows. Let $G = (V, E)$ be a vertex-unit-capacity graph. Initialize node-weighting $A_0 = \mathbb{1}_V$. At each level $0 \le i \le d$, computes a cut $C_{i+1} \subseteq V$ such that A_i is (h, s)-length ϕ-vertex-expanding in $G - C_{i+1}$, and set $A_{i+1} := \mathbb{1}_{C_{i+1}}$. In particular, the top level d has $C_{d+1} = \emptyset$. The LC-vertex-expander decomposition guarantees $|C_{i+1}| \approx \phi|A_i|$, so the number d of levels is $O(1/\epsilon)$ by choosing proper ϕ.

Next, we construct the shortcut as follows. For each i, by the expansion of A_i, we can add stars on the support of A_i into our shortcut so that any flows routing h-length A_i-respecting demands in $G - C_{i+1}$ can be shortcut. To analyze the shortcut quality, we will no longer try to route from A_i to A_{i+1} as in the edge-capacitated setting, because we no longer have boundary-linkedness guarantee.

Our analysis is instead *top-down*. At each level i, we shortcut the current flow path as much as possible, and then the prefix and suffix that have not yet been shortcut will be deferred to lower levels as subproblems. To be more concrete, say our initial goal is to shortcut a flow path P in a feasible h-length original flow. At each level $0 \le i \le d$, assume we will receive a subpath P' of P with length at most h in $G - C_{i+1}$ (note that P is a valid input to the top level d because C_{d+1} is empty). We will shortcut P' using star graphs at levels up to i as follows (see Figure 2 for an illustration when $i = 1$).

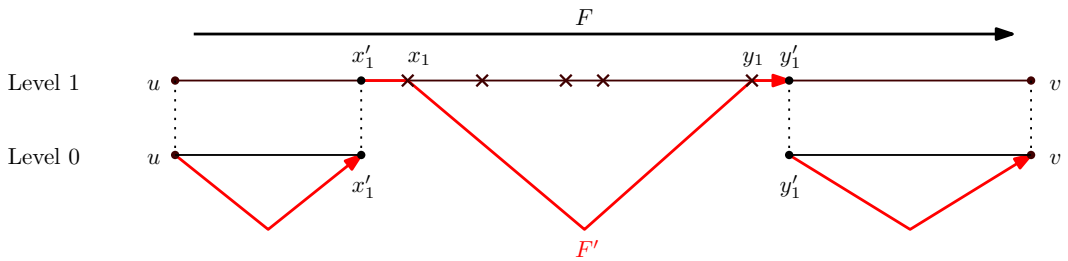

Figure 2 A toy example of forward mapping given we have 2 levels in total. Crossings represent cut vertices in C_1 along the witness path $P_{u,v}$.

Step 1. Let x_i and y_i be the first and last P'-vertices in $\text{supp}(A_i)$ respectively. We can easily shortcut the subpath $P'[x_i, y_i]$ (i.e. the subpath from x_i to y_i) within 2 steps using the star graphs at level i.

Step 2. Let x'_i be the P'-vertex right before x_i and let y'_i be the P'-vertex right after y_i. We regard shortcutting the prefix $P'[u, x'_i]$ and the suffix $P'[y'_i, v]$ as two subproblems at level $i-1$, where u, v are endpoints of P'. Note that both $P'[u, x'_i]$ and $P'[y'_i, v]$ has length at most h in $G - C_i$ because they are disjoint from C_i by definition.

Step 3. After the recursion, we obtain shortcuts for both $P'[u, x'_i]$ and $P'[y'_i, v]$. The shortcut for P' is given by concatenating shortcuts for $P'[u, x'_i], P'[x_i, y_i]$ and $P'[y'_i, v]$ using two original edges (x'_i, x_i) and (y_i, y'_i).

It is not hard to see the final step bound is $2^{O(d)} = 2^{O(1/\epsilon)}$ since the recursion has d levels and each level has two branches. We note that the actual argument is more complicated because the cuts C_i are actually moving cuts which have fractional cut values, and there is no clear partition of P' into 3 parts.

3 Length-Constrained Directed Expansion

In this section, we follow the theory of length-constrained expansion and extend it to the setting of directed graphs. We start with the generalization of notations from length-constrained expanders, which serves as the foundation for subsequent results. Next, we characterize length-constrained expansion in directed graphs with routing, and show the existence of length-constrained directed expander decomposition.

Basic Concepts of Length-Constrained Directed Expansion. The following definition of moving cuts and separation was introduced by Haeupler, Wajc and Zuzic in [25].

▶ **Definition 2** (Length-Constrained Cut). *An h-length moving cut $C : E \mapsto \{0, \frac{1}{h}, \frac{2}{h}, \ldots, 1\}$ assigns to each edge e a fractional cut value between zero and one which is a multiple of $\frac{1}{h}$. The* size *of C is defined as $|C| = \sum_e u(e) \cdot C(e)$. The length increase associated with the h-length moving cut C is denoted with $\ell_{C,h}$ and defined as assigning an edge e the length increase $\ell_{C,h}(e) = h \cdot C(e)$. Any moving cut which only assigns cut values equal to either 0 or 1 is called a pure moving cut. We define the degree of a moving cut over vertex v to be $\deg_C(v) = \sum_{e \ni v} u_G(e) \cdot C(e)$.*

▶ **Definition 3** (h-Length Separated Demand). *For any demand D and any h-length moving cut C, we define the amount of h-length separated demand as the sum of demands between vertices that are h-length separated by C. We denote this quantity with $\text{sep}_h(C, D)$, i.e.,*

$$\text{sep}_h(C, D) = \sum_{u,v:\text{dist}_{G-C}(u,v)>h} D(u, v).$$

▶ **Definition 4** (h-Length Sparsity of a Cut C for Demand D). *For any demand D and any h-length moving cut C with $\text{sep}_h(C, D) > 0$, the h-length sparsity of C with respect to D is the ratio of C's size to how much demand it h-length separates i.e.,*

$$\text{spars}_h(C, D) = \frac{|C|}{\text{sep}_h(C, D)}.$$

Above we generalize the definition of length-constrained moving cut w.r.t arbitrary directed h-length demand. However, for the definition of a directed length-constrained expander, we restrict to symmetric h-length demands.

▶ **Definition 5** ((h,s)-Length Sparsity of a Cut w.r.t. a Node-Weighting). *The (h,s)-length sparsity of any $h \cdot s$-length moving cut C with respect to a node-weighting A is defined as:*

$$\text{spars}_{(h,s)}(C, A) = \min_{A\text{-respecting } h\text{-length symmetric demand } D} \text{spars}_{h\cdot s}(C, D).$$

Intuitively, $(h \cdot s)$-length sparsity of a cut measures how much it $h \cdot s$-length separates h-length demand w.r.t its own size. Furthermore, for a given node-weighting, we associate the sparsest cut w.r.t the node-weighting with its conductance.

▶ **Definition 6** ((h,s)-Length Conductance of a Node-Weighting). *The (h,s)-length conductance of a node-weighting A in a graph G is defined as the (h,s)-length sparsity of the sparsest $h \cdot s$-length moving cut C with respect to A, i.e.,*

$$\text{cond}_{(h,s)}(A) = \min_{h\cdot s\text{-length moving cut } C} \text{spars}_{(h,s)}(C, A).$$

▶ **Definition 7** ((h,s)-Length ϕ-Expanding Node-Weightings). *We say a node-weighting A is (h,s)-length ϕ-expanding if the (h,s)-length conductance of A in G is at least ϕ.*

To see the connection, in the full version, we explain how our notion of length-constrained directed expansion generalizes the non-length-constrained version. Lastly, we give the formal definition of length-constrained directed expander decompositions as follows:

▶ **Definition 8** (Length-Constrained Directed Expander Decomposition). *Given a graph $G = (V, E)$, a directed (h,s)-length ϕ-expander decomposition for a node-weighting A with length slack s and cut slack κ is an $h \cdot s$-length cut C of size at most $\kappa \cdot \phi|A|$ such that A is (h,s)-length ϕ-expanding in $G - C$.*

Routing Characterization of Length-Constrained Directed Expansion. The definition of ϕ-expanding characterizes the sparsity of moving cuts in directed graphs. With the routing characterization, we further show that sparsity is closely related to demand routing.

▶ **Theorem 9** (Routing Characterization of Length-Constrained Directed Expanders). *Given a directed graph G and node-weighting A, for any $h \geq 1$, $\phi < 1$ and $s \geq 1$ we have:*

- *If A is (h,s)-length ϕ-expanding in G, then every A-respecting h-length symmetric demand can be routed in G with congestion at most $O(\frac{\log N}{\phi})$ and length at most $h \cdot s$.*
- *If A is not (h,s)-length ϕ-expanding in G, then some A-respecting h-length symmetric demand cannot be routed with congestion at most $\frac{1}{2\phi}$ and length at most $\frac{h\cdot s}{2}$.*

The proof idea of Theorem 9 is similar to the undirected case as shown in [24], and for completeness, we restate and adapt the proof for the directed setting in the full version.

Existence of Length-Constrained Directed Expander Decompositions. Now, we prove the existence of length-constrained directed expander decompositions. The following theorem formally states the result:

▶ **Theorem 10.** *For any $G = (V, E)$, a node-weighting A, $h > 1$, $\alpha \geq 1$, $\phi < 1$ and a length slack parameter $s = O(\log n)$, there is a directed (h,s)-length ϕ-expander decomposition for A with cut slack $\kappa = O(n^{O(\frac{1}{s})} \log n)$.*

The proof of Theorem 10 closely follows the undirected-case proof in [24], and we append it in the full version for completeness. The basic idea is that, unless the graph is already an (h, s)-length ϕ-expander for node-weighting A, we can repeatedly identify and apply cuts with (h, s)-sparsity less than ϕ. The union of these cuts ultimately renders the graph as an (h, s)-length ϕ-expander. To bound for the total size of all cuts, we sum individual cut sizes. However, since each cut size depends on the sparsity associated with different demands, analysis becomes complex. Thus, we first introduce a special base demand, called exponential demand to relate all other demands in terms of sparsity. Using this, we apply a potential argument to prove the above main theorem.

In the full version, we further discuss *boundary-linked* LC-directed expander decomposition (also called *linked* LC-directed expander decomposition). Expander decompositions with boundary-linkedness have been shown to be very useful in the length-constrained undirected setting and the classic (i.e. non-length-constrained) setting.

4 Length-Constrained Vertex Expansion

In this section we extend the theory of length-constrained expander decomposition to vertex-capacitated graphs. The basic concepts of length-constrained vertex expansion are analogous to those of length-constrained directed expansion in Section 3. The major difference is that now a moving cut C can assign cut values to both vertices and edges. See the full version for preliminaries and formal description of the basic concepts of vertex-capacitated graphs.

The main results of this section is the existence of length-constrained expander decomposition for vertex-capacitated graphs (Theorem 11) and the routing characterization of length-constrained vertex expanders (Theorem 12).

▶ **Theorem 11** (Existential (h, s)-length Expander Decomposition for Vetex-Capacitated Graphs)**.** *For any vertex-capacitated graph $G_{\text{vc}} = (V_{\text{vc}}, E_{\text{vc}})$, node-weighting A_{vc}, $h > 1$, $\phi < 1$ and a length slack parameter $s = O(\log n)$, there is an (h, s)-length ϕ-expander decomposition for A with cut slack $\kappa = O(n^{O(\frac{1}{s})} \log n)$.*

▶ **Theorem 12** (Routing Characterization of Length-Constrained Vertex Expanders)**.** *Given a vertex-capacitated graph G_{vc} and node-weighting A_{vc}, for any $h \geq 1$, $\phi < 1$ and $s \geq 1$ we have:*

- *If A_{vc} is (h, s)-length ϕ-expanding in G_{vc}, then every h-length A_{vc}-respecting demand can be routed in G_{vc} wth congestion at most $O(\frac{\log N}{\phi})$ and dilation at most $h \cdot s$.*
- *If A_{vc} is not (h, s)-length ϕ-expanding in G_{vc}, then some h-length A_{vc}-respecting demand cannot be routed with congestion at most $\frac{1}{6\phi}$ and dilation at most $\frac{h \cdot s}{2}$.*

The proofs of above are in the full version. We introduce a reduction that transforms vertex-capacitated graphs into directed edge-capacitated graphs to show the equivalence, which is crucial for the proofs of Theorem 11 and Theorem 12.

5 Length-Constrained Vertex-Capacitated Flow Shortcuts

In this section, we show the existence of LC-flow shortcuts in vertex-capacitated graphs. The Definition 13 below generalizes Definition 1 of LC-flow shortcuts in the sense that an additional length parameter h is given and the forward mapping only holds for demands routable in G with congestion 1 and length at most h.

Algorithm 1 LC-FlowShortcut(G, ϵ, h).

Initialize $A_0 = u_{V(G)}$, where $u_{V(G)}$ denotes the vertex capacity function of G.
Initialize $s = 1/\epsilon$, $\phi = 1/(n^{\epsilon}\kappa)$ ($\kappa = O(n^{O(1/s)} \log n)$ is the cut slack from Theorem 11).
Initialize $i \leftarrow 0$.
while $|A_i| > 0$ **do**
  for j from 1 to $\lceil \log_2 h \rceil$ **do**
    $h_j \leftarrow 2^j, h_{\mathrm{cov},j} = 4h_j, h_{\mathrm{diam},j} = h_{\mathrm{cov},j} \cdot s$.
    $C_{i+1,j} \leftarrow$ an $(h_{\mathrm{diam},j}, s)$-length ϕ-expander decomposition of A_i in G by Theorem 11.
    $\mathcal{N}_{i,j} \leftarrow$ a neighborhood cover with covering radius $h_{\mathrm{cov},j}$, diameter $h_{\mathrm{diam},j}$ in $G - C_{i+1,j}$ by Theorem 2.1 in the full version.
    $H_{i,j} = \bigcup_{S \in \mathcal{N}_{i,j}} H_S$, where H_S is the $h_j s$-length A_i-capacitated star graph on S.
  end for
  $A_{i+1} = \sum_j \frac{h_{\mathrm{diam},j} \cdot s}{h_j} \cdot \deg_{C_{i+1,j}} = 4s^2 \cdot \sum_j \deg_{C_{i+1,j}}$.
end while
Return $E' = \bigcup_{i,j} E(H_{i,j})$.

▶ **Definition 13** (Length-Constrained Flow Shortcut). *Given a graph $G = (V, E)$, we say an edge set E' (possibly with endpoints outside V) is an t-step h-LC-flow shortcut of G with length slack λ and congestion slack κ if*

- *(Forward Mapping) for every demand D routable in G with congestion 1 and length $h' \le h$, D is routable in $G \cup E'$ with congestion 1, length $\lambda h'$, and maximum step t, and*
- *(Backward Mapping) for every demand D on $V(G)$ routable in $G \cup E'$ with congestion 1 and length h', D is routable in G with congestion κ and length h'.*

We note that in [20] there is an analogous but weaker definition of h-LC-flow shortcut, in which the forward mapping only guarantee that D is routable in $G \cup E'$ with length λh instead of $\lambda h'$ (and the same congestion and step). That is, the length slack in Definition 13 is *competitive* in the sense that it upper bounds the ratio between the lengths of the shortcut flow and the original flow. Hence, by choosing a sufficiently large h, the total length of vertices and edges in G, an h-LC-flow shortcut is automatically an LC-flow shortcut.

▶ **Theorem 14.** *Given a vertex-capacitated graph G with parameters $\epsilon = \Omega(\frac{1}{\log n})$, there exists a t-step LC-flow shortcut E' with length slack $O(1/\epsilon^3)$, congestion slack $O(n^{O(\epsilon)} \log^3 n/\epsilon^2)$, $t = 2^{O(\frac{1}{\epsilon})}$ and size $|E'| \le O(n^{1+O(\epsilon)} \log n/\epsilon^2)$.*

Theorem 14 is the main theorem of this section. In what follows, actually we will focus on constructing h-LC-flow shortcut. Setting $h = (m+n)N$ gives the LC-flow shortcut in Theorem 14.

5.1 The Construction

The construction of the flow shortcut graph G' is given by Algorithm 1. The star graphs in Algorithm 1 are formally defined in Definition 15.

▶ **Definition 15** (Star Graphs). *Given a graph G with a node-weighting A and a length parameter h, the h-length A-capacitated star graph on some $S \subseteq V(G)$, denoted by H_S, has*

$$V(H_S) = (\mathrm{supp}(A) \cap S) \cup \{r_S\} \text{ and } E(H_S) = \{(v, r_S) \mid v \in V(H_S) \setminus \{r_S\}\},$$

where the vertex r_S is a Steiner vertex serving as the center and $V(H_S) \setminus \{r_S\}$ are original vertices. The length and capacity of each original vertex is unchanged, while r_S has length 1 and capacity $\sum_{v \in S} A(v)$. Each edge (v, r_S) has length h and capacity $A(v)$.

In short, Algorithm 1 mainly constructs a length-constrained expander hierarchy

$$\{A_i, C_{i+1,j} \mid 0 \le i \le d, 1 \le j \le \lceil \log_2 h \rceil\}$$

where d is the largest i such that $|A_i| > 0$. We point out that $C_{d+1,j}$ is a zero cut for all j. Then the shortcut graph G' is obtained by adding star graphs on neighborhoods of each LC-expander $G - C_{i+1,j}$.

We remark that we do LC-expander decompositions with different length parameters h_j at one level because we aim at a shortcut graph with length slack significantly smaller than its step bound. Intuitively, if we only use LC-expanders with length parameter around h to shortcut an original h-length flow path P, then inevitably each step will have length around h, which means the length slack cannot go far below the number of steps. Now, providing LC-expanders with different length parameters, when we want to shortcut a subpath of P with length h' far smaller than h, we can choose the appropriate LC-expander to obtain a shortcut with length around h' instead of h. Another benefit is that this automatically gives a competitive length slack (this is why in Definition 13 we define the length slack of h-LC-flow shortcut to be competitive).

We first argue the size bound of E'. Observe that, in Algorithm 1, we have $|C_{i+1,j}| \le \kappa\phi|A_i| \le |A_i|/n^\epsilon$ by Theorem 11. In Algorithm 1, the width of each neighborhood cover $\mathcal{N}_{i,j}$ is $\omega = n^{O(1/s)}s = n^{O(\epsilon)}/\epsilon$ by Theorem 2.1 in the full version. Furthermore, because in Algorithm 1 we have $|A_{i+1}| \le 4s^2 \sum_{j'} |C_{i+1,j'}| \le 4s^2 \log h |A_i|/n^\epsilon = O(\log N/(\epsilon^2 n^\epsilon))|A_i|$, we can upper bound d by $d \le O(\log |A_0|/\log(\epsilon^2 n^\epsilon/\log N)) = O(1/\epsilon)$. Finally, by the algorithm, we have

$$|E'| \le O(d \log h) \cdot \omega \cdot n = O(n^{1+O(\epsilon)} \log n/\epsilon^2).$$

Next we show the quality of the shortcut. Before that, we introduce a helper lemma Lemma 16, which shows the demands that each $H_{i,j}$ can route within small steps. We defer its proof to the full version.

▶ **Lemma 16.** *For each i, j, any demand $\hat{D}$ that is $h_{\mathrm{cov},j}$-length in $G - C_{i+1,j}$, A_i-respecting and $u_{V(G)}$-respecting can be routed in $H_{i,j}$ with length $2h_{\mathrm{diam},j}s + 1$, congestion 1 and step 2.*

Now we show that the shortcut E' constructed by Algorithm 1 has length slack $O(1/\epsilon^3)$, congestion slack $n^{O(\epsilon)}$ and step $2^{O(1/\epsilon)}$. To do this, it suffices to show the quality of the forward mapping and backward mapping, i.e. Lemma 17 and Lemma 18, whose proofs are given in Section 5.2 and the full version. Let $G' = G \cup E'$ be the shortcut graph.

▶ **Lemma 17** (Forward Mapping). *For any feasible h-length flow F in G, there is a feasible flow F' routing $\mathrm{Dem}(F)$ in G' with $\mathrm{leng}(F') \le \mathrm{leng}(F) \cdot O(1/\epsilon^3)$ and $\mathrm{step}(F') \le 2^{O(1/\epsilon)}$.*

▶ **Lemma 18** (Backward Mapping). *For any feasible flow F' in G' such that $V(\mathrm{Dem}(F')) \subseteq V(G)$, there is a flow F routing $\mathrm{Dem}(F')$ in G with $\mathrm{leng}(F) \le \mathrm{leng}(F')$ and $\mathrm{cong}(F) \le n^{O(\epsilon)}$.*

5.2 Forward Mapping: Proof of Lemma 17

We will employ a top-down argument. Start from the top level d. At the beginning of processing a level $i \geq 0$, we are given a feasible flow F_i with the following invariant: each flow path $P \in \text{path}(F_i)$ has $\text{leng}(P, G - C_{i+1,j}) \leq h_{\text{cov},j} = 4h_j$, where j is the minimum index s.t. $h_j \geq \text{leng}(P, G)$ (which means $h_j/2 \leq \text{leng}(P, G) \leq h_j$). Initially at the top level d, we set $F_d = F$. Note that F_d satisfies the invariant above because $C_{d+1,j}$ is a zero cut for any j.

First, at the bottom level $i = 0$, we can easily shortcut every flow path in F_0. For each star graph $H_{0,j}$, we assign it a demand $\hat{D}_{0,j}$ which sums over $\text{Dem}(P)$ for each flow path $P \in \text{path}(F_0)$ such that j is the minimum index with $h_j \geq \text{leng}(P, G)$. Observe that each $\hat{D}_{0,j}$ is an $h_{\text{cov},j}$-length in $G - C_{1,j}$ and A_0-respecting, so it can be routed in $H_{0,j}$ with length $2h_{\text{diam},j}s + 1$, congestion 1 and step 2 by Lemma 16.

From now on we consider levels $i \geq 1$. When processing a level i, for each flow path $P \in \text{path}(F_i)$, we may shortcut some subpaths of P using shortcut edges in H. The subpaths of P that have not been shortcut will be added to F_{i-1}, meaning that they are deferred to lower levels to get shortcut. At the end, the final F_{i-1} should ensure the invariants above, and we proceed to the lower level $i - 1$.

Now we will explain the shortcut at a level $i \geq 1$ in detail. Fix a flow path $P \in \text{path}(F_i)$. Let j be the minimum index such that $h_j \geq \text{leng}(P, G)$. We consider two cases.

Case 1: Defer. When $4s^2 \cdot \sum_{j'} C_{i,j'}(P) \leq 3$, we simply add P to F_{i-1}. We have

$$\text{leng}(P, G - C_{i,j}) = \text{leng}(P, G) + h_{\text{diam},j} \cdot s \cdot C_{i,j}(P) \leq 4h_j \leq h_{\text{cov},j},$$

where the inequality is by $\text{leng}(P, G) \leq h_j$, $h_{\text{diam},j} = 4sh_j$ and $C_{i,j}(P) \leq 3/(4s^2)$. Therefore, in this case, the flow path added to F_{i-1} satisfies the invariant.

Case 2: Shortcut. Now suppose $4s^2 \cdot \sum_{j'} C_{i,j'}(P) > 3$. Let u and v be P's endpoints. We say u is the left side and v the right side. For each $0 \leq k \leq |P|$, we refer to w_k as the P-vertex with k steps away from u. In particular, $w_0 = u$ and $w_{|P|} = v$.

We now define two functions $x : V(P) \to \mathbb{R}$ denoting the *budgets* of P-vertices from the left and the right respectively: for each vertex $w_k \in V(P)$,

$$x_P(w_k) = 4s^2 \sum_{j'} C_{i,j'}(w_{k-1}, w_k) + C_{i,j'}(w_k) + C_{i,j'}(w_k, w_{k+1}).$$

In particular,

$$x_P(w_0) = 4s^2 \sum_{j'} C_{i,j'}(w_0) + C_{i,j'}(w_0, w_1) \text{ and}$$

$$x_P(w_{|P|}) = 4s^2 \sum_{j'} C_{i,j'}(w_{|P|-1}, w_{|P|}) + C_{i,j'}(w_{|P|}).$$

Let k_L be the minimum index such that $\sum_{0 \leq k \leq k_L} x_P(w_k) \geq 1$, and symmetrically let k_R be the maximum index such that $\sum_{k_R \leq k \leq |P|} x_P(w_k) \geq 1$. To avoid clutter, we let $L = \{w_0, ..., w_{k_L}\}$ and $R = \{w_{k_R}, ..., w_{|P|}\}$. The following Claim 19 says that L and R have at most one common vertex.

▷ Claim 19. $k_L \leq k_R$.

Proof. By the definition of k_L and k_R, we have

$$\sum_{0\le k\le k_L-1} x_P(w_k) < 1 \text{ and } \sum_{k_R+1\le k\le |P|} x_P(w_k) < 1.$$

However, $\sum_{0\le k\le |P|} x_P(w_k) \ge 4s^2 \sum_{j'} C_{i,j'}(P) > 3$. This means there exists a vertex w_k with $k_L - 1 < k < k_R + 1$, which implies $k_L \le k_R$ ◁

We assign each vertex $w \in L \cup R$ a *load* $x'_P(w) \le x_P(w)$, satisfying that $\sum_{w\in L} x'_P(w) = 1$ and $\sum_{w\in R} x'_P(w) = 1$. We consider the following three-phase strategy to route $F_i(P)$ flow units from u to v. *Phase 1:* the vertex u sends $F_i(P)$ flow units and each $w \in L$ receives $F_i(P)x'_P(w)$ units. *Phase 2:* Each vertex $w \in L$ sends $F_i(P)x'_P(w)$ units and each vertex $w \in R$ receives $F_i(P)x'_P(w)$ units. *Phase 3:* Each vertex $w \in R$ sends exactly $F_i(P)x'_P(w)$ units, and the vertex v receives $F_i(P)$ units.

The First and Third Phases. Roughly speaking, for the first and third phases, we will add their corresponding original flows into F_{i-1}, meaning that they will be deferred to lower levels to get shortcut.

Regarding the first phase, recall that for each $w_k \in L$, we want to route $F_i(P)x'(w_k)$ units from u to w_k. To do this, we add into F_{i-1} a flow path $P_k = P[w_0, w_{k-1}]$ with value $F_i(P)\cdot x'(w_k)$. That is, we require the lower levels to give a shortcut that routes $F_i(P)\cdot x'(w_k)$ from u to w_{k-1} (the P-vertex one step closer to u than w_k). Then, we route $F_i(P)x'(w_k)$ units from w_{k-1} to w_k using the original edge $(w_{k-1}, w_k) \in E(G)$.

The third phase is handled in a similar way. For each $w_k \in R$, we route $F_i(P)x'(w_k)$ flow units from w_k to w_{k+1} using the original edge (w_{k-1}, w_k), and then we add into F_{i-1} a flow path $P_k = P[w_{k+1}, v]$ with value $F_i(P) \cdot x'(w_k)$.

To proceed to the lower level $i-1$, it remains to show that the flow paths added into F_{i-1} satisfy the invariant. Consider a flow path $P_k = P[u, w_{k-1}]$ added from the first phase (where $w_k \le L$, i.e. $0 \le k \le k_L$). We want to show that $\text{leng}(P_k, G - C_{i,j_k}) \le h_{\text{cov},j_k}$, where j_k is the minimum index such that $h_{j_k} \ge \text{leng}(P_k, G)$. By the definition of the budget function x, we have $\sum_{j'} C_{i,j'}(P_k) \le \frac{1}{4s^2} \sum_{0\le k'\le k-1} x_P(w_{k'}) < 1/(4s^2)$, where the second inequality is by $k \le k_L$ and $\sum_{0\le k'\le k_L-1} x_P(w_{k'}) < 1$ (from the definition of k_L). In particular, $C_{i,j_k}(P_k) \le 1/(4s^2)$. Because C_{i,j_k} is an $(h_{\text{diam},j_k} s)$-length moving cut, we have

$$\text{leng}(P_k, G - C_{i,j_k}) = \text{leng}(P_k, G) + h_{\text{diam},j_k} \cdot s \cdot C_{i,j_k}(P_k) \le 2h_{j_k} \le h_{\text{cov},j_k},$$

as desired, where the first inequality uses $h_{\text{diam},j_k} = 4sh_{j_k}$. By a similar argument, we can show that each flow path added from the third phase also satisfies the invariant, so we will not explain it in detail.

The Second Phase. The second phase is where the shortcut happens. We define an arbitrary (multi-commodity) demand $\hat{D}_{i,P}$ capturing this single-commodity demand. Namely, $\hat{D}_{i,P}$ satisfies that (1) $|\hat{D}_{i,P}| = F_i(P)$; (2) for each $w \in L$, $\hat{D}_{i,P}(w, \cdot) = F_i(P)x'_P(w)$; and (3) for each $w \in R$, $\hat{D}_{i,P}(\cdot, w) = F_i(P)x'_P(w)$. We will *assign* $\hat{D}_{i,P}$ to $H_{i,j}$, meaning that we route $\hat{D}_{i,P}$ using shortcut edges in $H_{i,j}$.

By the above assignment, for each $H_{i,j}$ at level i, its total assigned demand, denoted by $\hat{D}_{i,j}$, sums over $\hat{D}_{i,P}$ of all $P \in \text{path}(F_i)$ s.t. j is the minimum index with $h_j \ge \text{leng}(P, G)$. The following Lemma 20 showing that $\hat{D}_{i,j}$ can be routed with low steps, small length and congestion 1, and we defer the proof to the full version.

▶ **Lemma 20.** *For each $H_{i,j}$ at level i, its total assigned demand $\hat{D}_{i,j}$ can be routed in $H_{i,j}$ with length $2h_{\mathrm{diam},j}s + 1$, congestion 1 and step 2.*

Quality of the Forward Mapping. It remains to show that $\mathrm{Dem}(F)$ can be routed in $G' = G \cup \bigcup_{i,j} H_{i,j}$ with length $\mathrm{leng}(F) \cdot O(1/\epsilon^3)$, congestion 1 and step $2^{O(1/\epsilon)}$. The proof for this can be found in the full version.

References

1 Daniel Agassy, Dani Dorfman, and Haim Kaplan. Expander decomposition with fewer inter-cluster edges using a spectral cut player. In *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 9:1–9:20, 2023. doi:10.4230/LIPICS.ICALP.2023.9.

2 Aaron Bernstein, Joakim Blikstad, Thatchaphol Saranurak, and Ta-Wei Tu. Maximum flow by augmenting paths in $n^{2+o(1)}$ time, 2024. doi:10.48550/arXiv.2406.03648.

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

4 Aaron Bernstein, Maximilian Probst Gutenberg, and Thatchaphol Saranurak. Deterministic decremental sssp and approximate min-cost flow in almost-linear time. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 1000–1008, 2022. doi:10.1109/FOCS52979.2021.00100.

5 Aaron Bernstein, Jan van den Brand, Maximilian Probst Gutenberg, Danupon Nanongkai, Thatchaphol Saranurak, Aaron Sidford, and He Sun. Fully-dynamic graph sparsifiers against an adaptive adversary. In Mikolaj Bojanczyk, Emanuela Merelli, and David P. Woodruff, editors, *49th International Colloquium on Automata, Languages, and Programming, ICALP 2022, July 4-8, 2022, Paris, France*, volume 229 of *LIPIcs*, pages 20:1–20:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.20.

6 Greg Bodwin and Gary Hoppenworth. Folklore sampling is optimal for exact hopsets: Confirming the $\sqrt{\mathrm{n}}$ barrier. In *64th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2023, Santa Cruz, CA, USA, November 6-9, 2023*, pages 701–720. IEEE, 2023. doi:10.1109/FOCS57990.2023.00046.

7 Y. Chang and T. Saranurak. Deterministic distributed expander decomposition and routing with applications in distributed derandomization. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 377–388, 2020. doi:10.1109/FOCS46700.2020.00043.

8 Yi-Jun Chang and Thatchaphol Saranurak. Improved distributed expander decomposition and nearly optimal triangle enumeration. In *Proceedings of the International Symposium on Principles of Distributed Computing (PODC)*, pages 66–73, 2019. doi:10.1145/3293611.3331618.

9 Chandra Chekuri and Alina Ene. The all-or-nothing flow problem in directed graphs with symmetric demand pairs. *Mathematical Programming*, 154:249–272, 2015. doi:10.1007/S10107-014-0856-Z.

10 Chandra Chekuri, Sanjeev Khanna, and F Bruce Shepherd. Multicommodity flow, well-linked terminals, and routing problems. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 183–192, 2005. doi:10.1145/1060590.1060618.

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

12 Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 612–623, 2022. doi:10.1109/FOCS54457.2022.00064.

13 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 *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 1158–1167, 2020. doi:10.1109/FOCS46700.2020.00111.

14 Julia Chuzhoy and Sanjeev Khanna. A new algorithm for decremental single-source shortest paths with applications to vertex-capacitated flow and cut problems. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 389–400, 2019. doi:10.1145/3313276.3316320.

15 Julia Chuzhoy and Sanjeev Khanna. Maximum bipartite matching in $n^{2+o(1)}$ time via a combinatorial algorithm. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 83–94, 2024. doi:10.1145/3618260.3649725.

16 Antoine El-Hayek, Monika Henzinger, and Jason Li. Fully dynamic approximate minimum cut in subpolynomial time per operation. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 750–784, 2025. doi:10.1137/1.9781611978322.22.

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

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

19 Lars Gottesbüren, Nikos Parotsidis, and Maximilian Probst Gutenberg. Practical expander decomposition. In *Proceedings of the European Symposium on Algorithms (ESA)*, pages 61:1–61:17, 2024. doi:10.4230/LIPICS.ESA.2024.61.

20 Bernhard Haeupler, D. Ellis Hershkowitz, Jason Li, Antti Roeyskoe, and Thatchaphol Saranurak. Low-step multi-commodity flow emulators. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 71–82. ACM, 2024. doi:10.1145/3618260.3649689.

21 Bernhard Haeupler, D Ellis Hershkowitz, and Zihan Tan. New structures and algorithms for length-constrained expander decompositions. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 1634–1645, 2024. doi:10.1109/FOCS61266.2024.00102.

22 Bernhard Haeupler, Jonas Huebotter, and Mohsen Ghaffari. A cut-matching game for constant-hop expanders. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1651–1678, 2025. doi:10.1137/1.9781611978322.51.

23 Bernhard Haeupler, Yaowei Long, and Thatchaphol Saranurak. Dynamic deterministic constant-approximate distance oracles with n^{ϵ} worst-case update time. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 2033–2044, 2024. doi:10.1109/FOCS61266.2024.00121.

24 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.

25 Bernhard Haeupler, David Wajc, and Goran Zuzic. Network coding gaps for completion times of multiple unicasts. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 494–505, 2020. doi:10.1109/FOCS46700.2020.00053.

26 William Hesse. Directed graphs requiring large numbers of shortcuts. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 665–669, 2003. URL: http://dl.acm.org/citation.cfm?id=644108.644216.

27 Yiding Hua, Rasmus Kyng, Maximilian Probst Gutenberg, and Zihang Wu. Maintaining expander decompositions via sparse cuts. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 48–69, 2023. doi:10.1137/1.9781611977554.CH2.

28 Shang-En Huang and Seth Pettie. Lower bounds on sparse spanners, emulators, and diameter-reducing shortcuts. *SIAM J. Discret. Math.*, 35(3):2129–2144, 2021. doi:10.1137/19M1306154.

29 Yonggang Jiang, Merav Parter, and Asaf Petruschka. New oracles and labeling schemes for vertex cut queries, 2025. doi:10.48550/arXiv.2501.13596.

30 Wenyu Jin and Xiaorui Sun. Fully dynamic s-t edge connectivity in subpolynomial time (extended abstract). In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 861–872. IEEE, 2021. doi:10.1109/FOCS52979.2021.00088.

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

32 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.

33 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 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 217–226, 2014. doi:10.1137/1.9781611973402.16.

34 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.

35 Jason Li. Deterministic mincut in almost-linear time. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 384–395, 2021. doi:10.1145/3406325.3451114.

36 Jason Li, Danupon Nanongkai, Debmalya Panigrahi, and Thatchaphol Saranurak. Near-linear time approximations for cut problems via fair cuts. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 240–275, 2023. doi:10.1137/1.9781611977554.CH10.

37 Jason Li and Thatchaphol Saranurak. Deterministic weighted expander decomposition in almost-linear time, 2021. doi:10.48550/arXiv.2106.01567.

38 Yaowei Long, Seth Pettie, and Thatchaphol Saranurak. Connectivity labeling schemes for edge and vertex faults via expander hierarchies. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1–47, 2025. doi:10.1137/1.9781611978322.1.

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

40 Chaitanya Nalam, Thatchaphol Saranurak, and Sorrachai Yingchareonthawornchai. Deterministic k-vertex connectivity in k^2 max-flows, 2023. doi:10.48550/arXiv.2308.04695.

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

42 Harald Räcke, Chintan Shah, and Hanjo Täubig. Computing cut-based hierarchical decompositions in almost linear time. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 227–238, 2014. doi:10.1137/1.9781611973402.17.

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

44 Thatchaphol Saranurak and Sorrachai Yingchareonthawornchai. Deterministic small vertex connectivity in almost linear time. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 789–800, 2022. doi:10.1109/FOCS54457.2022.00080.

45 Jonah Sherman. Nearly maximum flows in nearly linear time. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 263–269, 2013. doi:10.1109/FOCS.2013.36.

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

47 Aurelio L. Sulser and Maximilian Probst Gutenberg. A simple and near-optimal algorithm for directed expander decompositions. *CoRR*, abs/2403.04542, 2024. doi:10.48550/arXiv.2403.04542.