# Stronger Directed Low-Diameter Decompositions with Sub-Logarithmic Diameter and Separation

Bernhard Haeupler* Richard Hladík† Shengzhe Wang‡ Zhijun Zhang§

**Abstract**

This paper significantly strengthens directed low-diameter decompositions in several ways.

We define and give the first results for separated low-diameter decompositions in directed graphs, tighten and generalize probabilistic guarantees, and prove new independence results between (far away) edges. Our results are the first to give meaningful guarantees for decompositions with small diameters $D = \Omega(\log \log n)$ in contrast to the state of the art that only applies to super-logarithmic diameters $D = \omega(\log n)$.

These results transfer several important and widely used aspects of undirected low-diameter decompositions to the directed setting. All our results are algorithmic – small modifications to two existing directed low-diameter decompositions [Bri+25; Li25] can be used to sample decompositions with our new guarantees in near-linear time $\tilde{O}(m)$.

*INSAIT, Sofia University "St. Kliment Ohridski" and ETH Zurich. bernhard.haeupler@inf.ethz.ch. This research was partially funded by the Ministry of Education and Science of Bulgaria (support for INSAIT, part of the Bulgarian National Roadmap for Research Infrastructure) and by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (ERC grant agreement 949272).

†ETH Zurich. richard.hladik@inf.ethz.ch. Partially funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (ERC grant agreement 949272).

‡ETH Zurich. shengzhe.wang@inf.ethz.ch Partially funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (ERC grant agreement 949272).

§INSAIT, Sofia University "St. Kliment Ohridski". zhijun.zhang@insait.ai. This research was partially done while at Princeton University and was partially funded by the Ministry of Education and Science of Bulgaria (support for INSAIT, part of the Bulgarian National Roadmap for Research Infrastructure).

# Contents

# 1 Introduction

This paper proves stronger and new guarantees for low-diameter decompositions (LDDs) in directed graphs and shows how state-of-the-art algorithms [Bri+25; Li25] can be adapted to sample such decompositions in near-linear time.

Low-diameter decompositions aim to randomly cut a graph into clusters of bounded diameter as uniformly and unbiased as possible. A sample from a low-diameter decomposition should result in a decomposition where every cluster behaves, as much as possible, like a uniformly random, representative low-diameter neighborhood. For example, edges should be cut with probability inversely proportional to the target diameter $D$ by a random decomposition and no edge should be much more likely to be cut than any other edge. More generally, one should expect that every short edge, short path, or small neighborhood has a good probability of being clustered (i.e., contained within a single cluster), where short and small are measured in terms of $D$. Moreover, in a random decomposition, far away parts of a graph should be clustered essentially independently, ensuring for example that any shortest path much longer than $D$ gets cut roughly equally often and not too many clusters just barely graze any given long path.

Since their introduction in the 80's [Awe85], many variants/notions of uniformity, and algorithms for LDDs have been developed (network decompositions [Awe+89], neighborhood covers [Awe+92a], and sparse covers [AP90] are some of the names used). LDDs were motivated from the start by applications and have led to countless applications in a wide variety of topics, areas, and fields over the last 40 years. A small subsample of examples includes applications to graph structures (spanners [Coh98; HMO23; FGN24], hopsets [Coh00; Hae+24], etc.), data structures (distance oracle [Awe+98; TZ01; HLS24], distance labeling [Pel00], etc.), network communications (oblivious routings [PU89; BLT14; Zuz+22], synchronizers [Awe85; Awe+92b; BLT14], etc.), metric embeddings [Bar96; Kra+04; Fil21], and efficient algorithms for shortest-paths [KS97; Awe+98; Coh00; BGS21], low-stretch trees [Alo+95; ABN08; Elk+08; KMP11; AN19; EN19], and related problems in a wide variety of models (parallel [MPX13; Ble+14], distributed [Awe+89; Awe+92a], (semi-)streaming [AGM12; FKM23], derandomization [Awe+89; RG20; Roz+22], etc.).

More recently, research on fast graph algorithms has pushed more into the much harder setting of directed graphs, and unsurprisingly, directed LDDs have started to show their impact there as well. In particular, Bernstein, Nanongkai, and Wulff-Nilsen [BNW22] showed how LDDs can be generalized to the directed setting and employed their probabilistic guarantees in a breakthrough result, giving a simple near-linear time combinatorial algorithm for negative-length single-source shortest paths in directed graphs. Follow-up works have refined both the decomposition [Bri+25; Li25] and its application in this context [BCF23; Fis+25]. These are just the beginning of our understanding of directed LDDs and we expect that in the (near) future, many different versions and refinements of directed LDDs will be developed and find a wide variety of applications, as their undirected versions have over the last 40 years and very much continue to this day.

This paper makes several steps in this direction by giving positive and negative results on how key aspects of undirected LDDs transfer and work in the directed setting. The rest of this paper is organized as follows: We first formally state the simplest undirected LDD guarantee in Section 1.1, which also maps directly to the current state-of-the-art for the directed setting. In Section 1.2, we then state many more general and stronger LDD properties that have been developed in the undirected setting. These properties are key and crucially used in many applications. Section 2.1 states our results, which demonstrate that many of these guarantees have equivalents in the directed setting. In Section 2.2, we summarize our techniques. In Sections 4 and 5, respectively, we present and analyze algorithms adapted from [Bri+25; Li25] to prove and provide efficient sampling algorithms for these new and improved directed LDD guarantees.

### 1.1 Simplest Low-Diameter Decompositions – Undirected and Directed

In this section, we first give some minimal notation and key definitions. Then, the simplest version of undirected LDDs is stated. We will also explain how this corresponds to the current state-of-the-art in the directed setting.

Consider a graph $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges. Throughout, we assume edges have positive integral lengths, which naturally induce shortest-path distances $\text{dist}_G(\cdot, \cdot)$. The *diameter* of a vertex subset $U$ is defined as $\text{diam}(U) = \max_{u,v \in U} \text{dist}_G(u, v)$. A *clustering* is a collection of disjoint vertex subsets $C_1, C_2, \ldots$ called *clusters*. We say a clustering has diameter $D$ if for every cluster $C_i$, its diameter $\text{diam}(C_i)$ is at most $D$. In the *undirected* setting, an edge $\{u, v\}$ is *clustered* by a clustering if there exists an $i$ such that cluster $C_i$ contains both endpoints, otherwise we say the clustering *cuts* the edge $\{u, v\}$.

With this minimal setup, we can now formally state the simplest undirected LDDs:

**Theorem 1.1** (Simplest Undirected LDDs)**.** *For any undirected graph $G$ and diameter $D$, there exists a distribution over $D$-diameter clusterings such that the following holds for every edge $e$ of length $d_e$:*

$$\Pr(e \text{ is cut}) \leq \frac{d_e}{D} \cdot \mathcal{O}(\log n).$$

While it is immediately obvious, by looking at a path of unit-length edges, that at least a $\frac{1}{D}$ fraction of edges needs to be cut, implying the multiplicative factor is at least 1, it is also known [Bar96] that a loss of $\Theta(\log n)$ is necessary and the best possible up to a constant.

*Low-diameter decompositions for directed graphs*, as proposed by Bernstein, Nanongkai, and Wulff-Nilsen [BNW22], are similar in that one cuts edges as uniformly as possible, such that the remaining *strongly connected components* have small diameters. Notably, this allows for edges to go between clusters as long as they do not form cycles or, equivalently, as long as they follow some topological order. The only difference in the formal setup is that we use the ordering of the clusters as a (topological) order and edges are considered cut only if they are against this order.

The best known directed LDDs, stated in the following, are due to Bringmann, Fischer, Haeupler, and Latypov [Bri+25] and Li [Li25].

**Theorem 1.2** (State-of-the-art Directed LDDs)**.** *For any directed graph $G$ and diameter $D$, there exists a distribution over ordered $D$-diameter clusterings such that the following holds for every edge $e$ of length $d_e$:*

$$\Pr(e \text{ is cut}) \leq \frac{d_e}{D} \cdot \mathcal{O}(\log n \log\log n).$$

In fact, a factor of $\mathcal{O}(\log^2 n)$ loss was initially proved by Bernstein, Nanongkai, and Wulff-Nilsen [BNW22], which was later improved by Bringmann, Fischer, Haeupler, and Latypov [Bri+25] to the state-of-the-art loss factor $\mathcal{O}(\log n \log\log n)$. Li [Li25] subsequently gave a different proof for the same loss factor. Given that the directed setting is a strict generalization of the undirected setting (one can replace an undirected edge by two directed edges), the loss factor of Theorem 1.2 is optimal up to the extra $\log\log n$ factor. Furthermore, this additional $\log\log n$ factor appears in many other different contexts (see for example [Bar98; AN19; Che+20]). Essentially, the same overhead can all be attributed to a general approach by Seymour [Sey95] in analyzing recursive structures. It seems very unlikely that current techniques can avoid this. It is quite possible that the $\log\log n$ loss is necessary and tight in the directed setting.

We remark that all results discussed above are known to have near-linear time implementations.

## 1.2 General Undirected Low-Diameter Decompositions

As mentioned above, much stronger guarantees than Theorem 1.1 are known for undirected LDDs. For any path, subset of edges, or ball, it is considered cut if any of its edges is cut. Next, we state the more general version, part or all of which can be obtained for example from [LS93; Awe+92a; Bar96; Ble+14; MPX13]:

**Theorem 1.3** (General Undirected LDDs)**.** *For any undirected graph $G$ and diameter $D$, there exists a distribution over $D$-diameter clusterings such that:*

1. *For every edge $e$ of length $d_e$:*
$$\Pr(e \text{ is not cut}) \geq \exp\left(-\frac{d_e}{D} \cdot \mathcal{O}(\log n)\right).$$
2. *For every path $P$ of length $d_P$:*
$$\Pr(P \text{ is not cut}) \geq \exp\left(-\frac{d_P}{D} \cdot \mathcal{O}(\log n)\right).$$
3. *For every subset of edges $\Gamma$ with $d_\Gamma = \sum_{e\in\Gamma} d_e$:*
$$\Pr(\Gamma \text{ is not cut}) \geq \exp\left(-\frac{d_\Gamma}{D} \cdot \mathcal{O}(\log n)\right).$$
4. *For every vertex $v$ and ball $B(v, d_B)$ centered at $v$ with radius $d_B$:*
$$\Pr(B(v, d_B) \text{ is not cut}) \geq \exp\left(-\frac{d_B}{D} \cdot \mathcal{O}(\log n)\right).$$
5. *For any two edges, paths, subsets of edges, or balls that are distance $\Omega(D)$ apart, the above probabilities are as if independent.*

*Furthermore, there is a near-linear time algorithm that samples such a low-diameter decomposition.*

Note that guarantees for edges and paths are direct corollaries of guarantees for subsets of edges and balls. Theorem 1.3 strengthens Theorem 1.1 in multiple ways as we discuss next:

**Stronger Probability Bounds and Clustering with Constant Diameters.** The first notable difference already occurs for the cutting probabilities of edges. Indeed, the $\exp(-\frac{d}{D}\cdot\mathcal{O}(\log n))$ guarantee for edges being not cut in Theorem 1.3 is strictly stronger than the $\frac{d}{D}\cdot\mathcal{O}(\log n)$ guarantee for an edge being cut in Theorem 1.1, because $\exp(-\frac{d}{D}\cdot\mathcal{O}(\log n)) > 1-\frac{d}{D}\cdot\mathcal{O}(\log n)$. In particular, for $d = \mathcal{O}(\frac{D}{\log n})$, both give at least a constant probability for an edge not to be cut. However, $1-\frac{d}{D}\cdot\mathcal{O}(\log n)$ becomes vacuous for even larger $d$ while $\exp(-\frac{d}{D}\cdot\mathcal{O}(\log n))$ remains meaningful for $d$ as large as $\epsilon D$ for any sufficiently small (constant) $\epsilon$, in which case with probability $n^{-\mathcal{O}(\epsilon)}$, the edge is not cut. This enables meaningful clustering results with constant-diameter clusters.

**Results for Subsets of Edges and Their (In)dependence.** The guarantees for edges can be generalized to hold for paths and even arbitrary subsets of edges. From Theorem 1.1, one can get a similar bound for subsets of edges, with a linear dependency on $d/D$, for free by union bound. However, the stronger bounds in Theorem 1.3 cannot be derived this way. Instead, note that intuitively, if only balls with independent and bounded radii are cut, the events of being cut

for two far away edges are as if independent, i.e., the probability bound still holds for one edge being cut, when we condition on the other far away edge. We give the formal definition for the independence property later in Theorem 4.1. Indeed, such property also holds for far away paths, subsets of edges, and balls as stated in Theorem 1.3. Furthermore, the probability guarantees for them essentially implies that even if they are not far away, these events are never worse than independent (i.e., could either be as if independent or positively dependent).

**Results for Neighborhoods.** Motivated by applications, especially in non-sequential models, there has been a line of work, tracing back to [AP90; Awe+92a; Awe+98], devoted to decompositions preserving not only individual edges but also small neighborhoods. Formally, Theorem 1.3 shows that all (potentially exponentially many) paths starting from a node and all vertices in a $d$-neighborhood are all clustered together with similar probabilities. To see why this is useful, note that it is basically equivalent to the following Theorem 1.4 about separated (partial) clusterings. A partial clustering is a clustering in which the union of $C_i$ is not all of $V$. We say a node $v$ is clustered if it is contained in some cluster and unclustered if $v \in V \setminus \bigcup_i C_i$. We remark that Theorems 1.2 and 1.3 do not require clusterings to be a partition. Nevertheless, since edges of any unclustered vertex are by definition always cut, it is without loss of generality to assume all vertices are clustered. Theorem 1.4 below examines clustering probabilities of vertices instead of cutting probabilities of edges and cares about separation between clusters.

**Theorem 1.4** (Undirected LDDs with Separation)**.** *For any undirected graph $G$, diameter $D$, and separation $d$, there exists a distribution over partial $D$-diameter clusterings such that:*

1. *For every vertex $v$:*
$$\Pr(v \text{ is clustered}) \geq \exp\left(-\frac{d}{D} \cdot \mathcal{O}(\log n)\right).$$
2. *For any two clusters $C, C'$, and any two vertices $u \in C, v \in C'$, it holds that $\text{dist}_G(u, v) > d$.*

Up to constants, Theorem 1.4 is actually equivalent to Theorem 1.3. To see this, fix the value of $d$. Given LDDs where neighborhoods are preserved by Theorem 1.3, we consider each vertex $v$ as clustered if its ball $B(v, d/2)$ is not cut. Then, for any two clustered vertices $u, v$ from different clusters, $B(u, d/2)$ and $B(v, d/2)$ are disjoint. Consequently, clusters are at least distance $d$ apart from each other. Conversely, suppose we are given separated clusterings by Theorem 1.4. We assign each unclustered vertex $v$ to the nearest cluster. Due to the separation, $v$ can only be in the ball $B(u, d/2)$ for originally clustered vertices $u$ in at most one cluster. Moreover, if such a cluster exists, $v$ is assigned to this cluster. As a result, the ball $B(u, d/2)$ is never cut for any originally clustered vertex $u$. For both directions, neighborhood preservation probabilities translate to vertex clustering probabilities and vice versa.

### 1.3 Importance of Decompositions with Small Diameter and Separation

It is important to note that in many applications of (undirected) low-diameter decompositions, the stronger guarantees of Theorem 1.4 and Theorem 1.3 compared to the simple per edge $1-\frac{d}{D}\cdot\mathcal{O}(\log n)$ cutting guarantee of Theorem 1.1 are absolutely crucial.

Indeed, in many cases $\frac{d}{D}$ corresponds to a stretch, length-slack, or approximation ratio while the clustering or cutting probability goes into the running time or size and being able to achieve sub-logarithmic, typically even constant, slack is key.

Examples among the many listed above include: Distance oracles and distance labeling schemes, which are data structures that provide $\mathcal{O}(k)$-approximate distances while storing only $\tilde{\mathcal{O}}(n^{1+1/k})$

bits of information about a graph. They are most interesting for $k$ being a small constant or at least sub-logarithmic and they crucially rely on clusterings with diameter proportional to $k$. Also intimately connected to LDDs are graph spanners. A $(2k+1)$-stretch spanner for a given graph $G$ is a sparse subgraph in which all distances are approximated up to the stretch-factor. For every graph there exists such a subgraph with only $\mathcal{O}(n^{1+1/k})$ edges. LDDs with diameter $D = k$ (for unweighted graphs with $d = 1$) have been key from the start in both proving the existence and giving fast (parallel and distributed) constructions of spanners and closely-related synchronizers [Awe85] (see for example the paper "Improved parallel algorithms for spanners and hopsets" [Mil+15]).

Separation and independence are also used in many applications. They lead for example to neighborhood covers with constant length-approximation or length-slack $\frac{d}{D}$. Together they for example guarantee that running an LDD with diameter $D$ and contracting the components shrinks the diameter of the graph by at least $d$ with high probability (see for example [HW16; CD21]). Even the optimal [Abb+22] way to compute $t$-pair approximate shortest paths that runs in $(m+t)\cdot n^{\mathcal{O}(1/k)}$ time simply uses $n^{\mathcal{O}(1/k)}$ LDDs for every scale $D = 2^i$, checks at what scale a pair appears in a joint cluster, and uses the fact that paths of length $d = D/k$ are clustered with good enough probability to obtain an $\mathcal{O}(k)$-approximation.

As a last of many examples, all recent works on length-constrained expanders (see for example [HHG25]) and their applications (such as $\mathcal{O}_\varepsilon(1)$-approximate dynamic distance oracles [HLS24] with $n^{1/\varepsilon}$ update time or $\mathcal{O}_\varepsilon(1)$-approximate $t$-commodity flows in $(m+t)^{1+1/\varepsilon}$ time [Hae+24] crucially rely on low-diameter decompositions with sub-logarithmic stretch.

# 2 Results and Technical Overview

## 2.1 Our Results

Our main contribution is the following result on directed LDDs that have the same strong guarantees as in Theorem 1.3 for undirected LDDs, except for the hard-to-avoid $\log\log n$ factor.

**Theorem 2.1** (General Directed LDDs)**.** *For any directed graph $G$ and diameter $D$, there exists a distribution over ordered $D$-diameter clusterings such that:*

1. *For every edge $e$ of length $d_e$:*
$$\Pr(e \text{ is not cut}) \geq \exp\left(-\frac{d_e}{D}\cdot\mathcal{O}(\log n\log\log n)\right).$$
2. *For every path $P$ of length $d_P$:*
$$\Pr(P \text{ is not cut}) \geq \exp\left(-\frac{d_P}{D}\cdot\mathcal{O}(\log n\log\log n)\right).$$
3. *For every subset of edges $\Gamma$ with $d_\Gamma = \sum_{e\in\Gamma} d_e$:*
$$\Pr(\Gamma \text{ is not cut}) \geq \exp\left(-\frac{d_\Gamma}{D}\cdot\mathcal{O}(\log n\log\log n)\right).$$
4. *For any two edges, paths, or subsets of edges that are distance $\Omega(D)$ apart* in the underlying undirected graph *(i.e., the graph obtained by replacing each directed edge with an undirected edge of the same length), the above probabilities are as if independent.*

*Furthermore, there is a near-linear time algorithm that samples such a low-diameter decomposition.*

The strong guarantees of Theorem 2.1 enable meaningful discussion about directed LDDs with substantially sub-logarithmic diameters as small as $\Theta(\log \log n)$.

We remark that compared to Theorem 1.3, Theorem 2.1 lacks the property for neighborhoods. As will be seen in Section 2.2, it turns out that it is impossible to preserve out-balls/in-balls for each vertex with nonzero probability in the directed setting. This also implies that we no longer obtain LDDs with separation for free. Indeed, it is, a priori, even unclear if directed LDDs with any notion of separation should exist at all. The recursive approaches taken by all known directed LDD constructions may increase distances between vertices when recursing into induced subgraphs. It thus demands a careful selection of clustered vertices to ensure that separation in induced subgraphs imply separation in the original graph.

Nevertheless, while there does not seem to be a truly directed analog for neighborhoods, we do prove the following meaningful generalization of Theorem 1.4 to the directed setting:

**Theorem 2.2** (Directed LDDs with Separation)**.** *For any directed graph $G$, diameter $D$, and separation $d$, there exists a distribution over ordered partial $D$-diameter clusterings such that:*

1. *For every vertex $v$:*

$$\Pr(v \text{ is clustered}) \geq \exp\left(-\frac{d}{D} \cdot \mathcal{O}(\log n \log \log n)\right).$$

2. *For any two clusters $C, C'$ such that $C$ is ordered after $C'$, and any two vertices $u \in C, v \in C'$, it holds that $\mathrm{dist}_G(u, v) > d$.*

*Furthermore, there is a near-linear time algorithm that samples such a low-diameter decomposition.*

This allows to show meaningful separation properties for partial clusterings with diameters as small as $\Theta(\log \log n)$.

## 2.2 Our Techniques

Next we give a technical overview of our directed LDDs.

**General Directed LDDs.** Theorem 2.1 is proved by adapting [Bri+25]'s algorithm. We first give a brief overview of [Bri+25]'s algorithm and its analysis, before introducing our modifications and new proof ideas.

At a high level, the algorithm of [Bri+25] repeatedly cuts balls of radii between $r_0$ and $r_1$, both of which are initially set to be $\Theta(D)$. Each time, it picks a random center vertex $x$ and samples a radius $r \sim r_0 + \mathsf{Geom}(p)$ for some $p$ such that $r \geq r_1$ (implying the algorithm fails) only occurs with probability $\delta = 1/\operatorname{poly}\log(n)$ across all cutting steps. If $r \geq r_1$ does occur, the entire algorithm restarts from the very beginning. Otherwise, by cutting the radius-$r$ ball around $x$, the algorithm recurses on the subgraph induced by the ball while continuing the cutting procedure on the remaining graph. Once there are no such balls, it gradually decreases $r_0, r_1$ (so cutting smaller balls) and adjusts $p$ accordingly. The analysis of (single edge) cutting probabilities crucially relies on the following arguments.

1. Since $\delta$ is very small, restarting the algorithm can at most double the cutting probability.
2. When no restarting happens, the overall cutting probability is upper bounded by a union bound over all cutting steps of individual balls.

3. For any edge $e = (u, v)$ of length $d_e$ and any ball $B$, by the memoryless property of geometric distributions, it is shown that conditioned on $u$ being included in $B$, $v$ is not included in $B$ (so $e$ is cut by $B$) with probability at most $\mathcal{O}(d_e p)$.

However, none of these arguments is suitable for obtaining our stronger exponential-type bounds.

1. A restarting probability of $\delta = 1/\operatorname{poly}\log(n)$ is no longer affordable as it can ruin any exponential-type bound we get, which can be as small as $n^{-\mathcal{O}(\epsilon)}$ for some $\epsilon$.

2. Even if exponential-type bounds can be shown for individual cutting steps using prior analysis, they cannot be straightforwardly combined using union bounds. It will not produce exponential-type bounds for overall cutting probabilities.

3. It turns out that proving exponential-type bounds for individual cutting steps is also problematic. Indeed, consider some radius-$r$ ball $B$ around some vertex $x$ and some edge $e = (u, v)$ such that $\operatorname{dist}(x, u) \in [r_1 - d_e, r_1)$. Conditioned on $u$ included in $B$, $v$ is actually never included in $B$ as $r \in [\operatorname{dist}(x, u), \operatorname{dist}(x, u) + d_e)$.[1] That is, $e$ is always cut. Recall that the ultimate goal is to show some small, but strictly positive, probability of not cutting each edge.

Our modified algorithm resolves the first issue by utilizing truncated exponential distributions instead of geometric distributions[2]. Regarding the second issue, independence between the cutting probabilities we choose to analyze is critical for enabling us to multiply them together. To address this as well as the third issue, we propose a new dynamic-programming-style approach in Section 4. It completely eliminates the need to analyze conditional cutting probabilities as in [Bri+25], by a more careful analysis of how the value of $r$, relative to $r_0, r_1$, affects the cutting probabilities. Intuitively, the scenario above in the third issue should not hurt too much as it implies $r \in [r_1 - d_e, r_1)$, which only happens with very small probability because $r_1 - r_0 \gg d_e$ and tail probabilities of exponential distributions decay super fast.

As an added bonus, our approach also enjoys the benefit of easy generalization to the case of paths and subsets of edges, with little extra effort. This is not possible with [Bri+25]'s analysis. Actually, even one path may end in multiple segments due to cutting a single ball. It is unclear if there is an easy way to deal with the complicated conditional probabilities arising from this. In contrast, our proof in Section 4 directly works with the case of subsets of edges and the other two follow as direct corollaries. We are able to show that there are essentially only two situations to consider for each cutting step, regardless of the subset of edges. Specifically, we can assume without loss of generality that either all edges are included in the ball and go into recursion, or all edges are outside the ball. Any other partition of edges into the two parts will never lead to larger cutting probabilities.

We also remark that the independence stated in Theorem 2.2 holds so long as the decomposition is obtained by cutting balls with independently sampled diameters at most $\mathcal{O}(D)$. This is because for each ball, at least one of the two subsets of edges can never be cut, regardless of the radius, due to the distance assumption. The claimed independence then follows from the independence between sampled radii. So far, many known LDD results, including [BNW22; BCF23; Bri+25], follow this generic approach, while [Li25] is one notable exception.

[1] $r \geq \operatorname{dist}(x, u)$ because $u$ is included in $B$, and $r < r_1 \leq \operatorname{dist}(x, u) + d_e$ because no restarting is allowed anymore due to the first issue.

[2] It is the truncation that eliminates restarting. The choice of exponential distributions over geometric distributions is mainly for technical reasons as geometric distributions are not well defined for $p > 1$. Other than that, exponential distributions can be viewed as the analog of geometric distributions in the continuous setting, and they both have the memoryless property, which is the most important for the sake of analysis.

**No Directed Neighborhoods.** Regarding neighborhoods, it is not clear what is the proper notion of neighborhood for a vertex in the directed setting. One tempting option would be the union of its out-ball and its in-ball. Alternatively, we may consider the intersection of its out-ball and its in-ball. It turns out that neither of the two definitions makes much sense. Indeed, it is even impossible for every vertex to have a strictly positive probability of its out-ball (or in-ball) not being cut, because of the following promised counterexample, which shows the requirement of not cutting an out-ball may be in conflict with the requirement of having a small diameter.

**Lemma 2.3.** *There exists a strongly connected graph with diameter $n-1$ such that all vertices have distance at most 1 from some vertex.*

*Proof.* Consider a directed cycle $(v_1, v_2, \ldots, v_n)$ and add additional edges from $v_1$ to all other vertices. Clearly, $v_1$ reaches every vertex using at most one edge. The diameter of the constructed graph is $n-1$ because the only path from $v_2$ to $v_1$ goes around the entire cycle. □

Instead, suppose one only cares about the intersection of the $d$-radius in-ball and the $d$-radius out-ball. Note that the intersection is a strongly connected component with diameter at most $2d$. The desired property is already implied by Theorem 1.3 since we can consider the round-trip distance $\mathrm{dist}_G(u,v) + \mathrm{dist}_G(v,u)$, which is a symmetric metric and thus undirected LDDs are sufficient. This has for example been observed and used in [Hae+25].

From the round-trip distance, one can also get a partial clustering in which any two clusters are "separated" in the following weak sense: For any two clusters $C, C'$, and any two vertices $u \in C$ and $v \in C'$, either $\mathrm{dist}_G(u,v) > d$ or $\mathrm{dist}_G(v,u) > d$. However, such guarantee is not of much use because clusters as a whole are not meaningfully separated in either direction. Indeed, there can be vertices $u, u' \in C$ and $v, v' \in C'$ such that $\mathrm{dist}_G(u,v) \ll d$ but $\mathrm{dist}_G(v,u) > d$, while $\mathrm{dist}_G(u',v') > d$ and $\mathrm{dist}_G(v',u') \ll d$.

**Directed LDDs with Separation.** Roughly speaking, to achieve separation in Theorem 2.2, each time we execute a cut, any vertex within distance $d$ (in the direction of cutting) to the boundary is marked as unclustered. Crucially, vertex marking is needed on both sides of the boundary to ensure that any short path between unmarked vertices are entirely contained in induced subgraphs after cutting (Lemma 5.8). In Section 5, we adapt and simplify [Li25]'s algorithm to prove Theorem 2.2. The construction of [Li25], based on [CKR00], samples a single cutting radius, which is used for all randomly ordered balls in the same iteration, from a uniform distribution. Due to this nature of the cutting radius, it may be quite surprising that any exponential-type bound would even hold in the first place.

At a high level, the essence of [Li25]'s proof is showing that even though the number $k$ of balls to cut in each iteration can be huge, randomly ordering all balls ensures that the probability of not clustering[3] any vertex $v$ only grows linearly in $\log k$. Building upon this, we further observe that to achieve separation $d$, relevant balls must satisfy $\mathrm{dist}(x,v) \in (r_0 - d, r_1 + d]$ where $x$ is the center of the ball and $(r_0, r_1]$ is the range from which the cutting radius is sampled. With $k \gg (r_1 + d) - (r_0 - d)$ balls, each of which has $\mathrm{dist}(x,v)$ be an integer in the range $(r_0 - d, r_1 + d]$, we are actually able to prove a much tighter bound of the form $1 - \exp(-\mathcal{O}(\log k))$ using convexity arguments (Lemma 5.4).

[3] Technically, [Li25]'s proof is for edge cutting, but it can be adapted to clustering with separation using our modification to the algorithm.

**Remark on [Bri+25] vs. [Li25].** For the modified [Li25]'s algorithm, we will show that the same probability guarantee as in Theorem 2.1 also holds for edges, but it is not clear if this is the case for paths or subsets of edges. In fact, by using the same radius, sampled from a uniform distribution, for all balls in each iteration, the events of being cut may be correlated in complicated ways even for edges that are further apart than $\Omega(D)$, as opposed to the independence stated in Theorem 2.1, which is satisfied by the modified [Bri+25]'s algorithm.

We also remark that our modified version of [Bri+25]'s algorithm can be adapted, by the same vertex marking procedure, to provide the separation guarantee as in Theorem 2.2. Vertex clustering probabilities can be bounded by a similar argument to the one for cutting probabilities of subsets of edges (each cutting step has $2d$ "bad" radii instead of $d_\Gamma$). Regarding the running time, one has to be careful as vertices are marked on both sides of the boundary. In other words, not all marked vertices go into recursion. We have to bound the total contribution of remaining vertices to the running time. Fortunately, since centers of balls are sampled randomly, a similar argument to the one for the modified [Li25]'s algorithm can show the running time is still subsumed by that of Cohen's algorithm [Coh97] with high probability. We choose to present the analysis for the modified [Li25]'s algorithm, as the same type of analysis is also crucial to proving probability bounds there.

## 3 Preliminaries

We consider directed simple graphs $G = (V, E)$, where $E \subseteq V^2$, with $n = |V|$ vertices and $m = |E|$ edges. For each vertex $v \in V$, its *degree* $\deg(v)$ is the total number of edges with $v$ being one of the endpoints. For any $U \subseteq V$, its *volume* $\mathrm{vol}(U)$ is defined as $\sum_{v \in U} \deg(v)$. Note that $\mathrm{vol}(V) = 2m$. We write $\overline{U} = V \setminus U$. Let $G[U]$ be the subgraph induced by $U$. We denote with $\delta^+(U)$ the set of edges that have their starting point in $U$ and endpoint in $\overline{U}$. We define $\delta^-(U)$ symmetrically.

For each edge $e \in E$, we write $d_e$ for its length. The distance from vertex $u$ to $v$ is written $\mathrm{dist}_G(u, v)$. For any $U \subseteq V$, we say that $U$ has *diameter* $D$ if for all pairs of vertices $u, v \in U$, we have $\mathrm{dist}_G(u, v), \mathrm{dist}_G(v, u) \leq D$. A strongly connected component in a directed graph $G$ is a subgraph where for every pair of vertices $u, v$ in the component, there is a path from $u$ to $v$ and vice versa. For a radius $r \geq 0$, the *out-ball* of vertex $v$ is $B^+_G(v, r) = \{x \in V \mid \mathrm{dist}_G(v, x) \leq r\}$ and the *in-ball* is $B^-_G(v, r) = \{y \in V \mid \mathrm{dist}_G(y, v) \leq r\}$. We may omit the subscript if it is clear from context.

## 4 Strong Bounds for Modified [Bri+25]'s Algorithm

In this section, we adapt and strengthen [Bri+25] in two ways, proving Theorem 2.1. We improve the bound on the probability that an edge is cut from $\mathcal{O}(d_e/D \cdot \log n \log\log n)$ to $1 - \exp(-\mathcal{O}(d_e/D \cdot \log n \log\log n))$, which is stronger when $d_e$ is close to $D$. We also show that analogous bounds even hold for paths and subsets of edges in general. The following is a more formal version of Theorem 2.1.

**Theorem 4.1.** *Let $\epsilon > 0$ be a sufficiently small constant. There is a randomized algorithm that, given any directed graph $G = (V, E)$ with $n$ vertices and $m$ edges, and diameter parameter $D = \mathrm{poly}(n)$, returns an ordered (full) clustering $\sigma = (C_1, \ldots, C_k)$ of $V$, satisfying all of the following.*

- **Edge cutting probability:** *Each edge $e = (u, v)$ is* cut*, i.e., $u \in C_i$ and $v \in C_j$ for $i > j$, with probability at most $1 - \exp(-d_e/D \cdot \mathcal{O}(\log n \log\log n))$ if $d_e \leq \epsilon D / \log\log n$.*
- **Path cutting probability:** *Each path $P$ is* cut*, i.e., some of its edges are cut, with probability at most $1 - \exp(-d_P/D \cdot \mathcal{O}(\log n \log\log n))$ if $d_P \leq \epsilon D / \log\log n$.*

**Subset cutting probability:** *Each subset of edges* $\Gamma$ *is* cut*, i.e., some of its edges are cut, with probability at most* $1 - \exp(-d_\Gamma/D \cdot \mathcal{O}(\log n \log\log n))$ *if* $d_\Gamma \leq \epsilon D/\log\log n$*.*

**Independence property:** *For any subsets of edges* $\Gamma, X, Y$ *such that for every vertex* $v$*, either* $B^{\pm}_G(v, D) \cap \Gamma$ *or* $B^{\pm}_G(v, D) \cap (X \cup Y)$ *is empty, the above bound for* $\Pr(\Gamma$ *is cut*$)$ *also holds for* $\Pr(\Gamma$ *is cut* $\mid$ *all of* $X$ *is cut* $\wedge$ *none of* $Y$ *is cut*$)$*.*

**Diameter property:** *Each* $C_i$ *is strongly connected and has diameter at most* $D$*.*

**Running time:** *The algorithm runs in* $\mathcal{O}(m \log^5 n \log\log n)$ *time with high probability.*

### 4.1 Overview of [Bri+25]'s Algorithm and Our Modification

Before going into our modification, let us quickly summarize the algorithm of [Bri+25] and describe the slight modifications we need to make the strengthened analysis work. In the following, we ignore many subtle computational and other issues irrelevant to our analysis and focus more on intuition. A more detailed description can be found in Algorithms 1 and 2.

**Algorithm summary.** The algorithm of [Bri+25] repeatedly picks a vertex $v$, samples a radius $r$ and then carves out the ball $B = B^+(v, r)$ from the graph, by removing its vertices from the vertex set and cutting the out-edge set $\delta^+(B)$. Then it proceeds by carving more balls in the remaining graph, while also calling itself recursively on $B$.

Logistically, the algorithm is split into two parts: the *cutting lemma* (Lemma 6.2 of [Bri+25]) repeatedly carves out balls of a certain kind (specified by the caller) from $G$, until no such ball remains with high probability. It returns the set of cut edges and the union of all balls that have been carved. The main part of the algorithm repeatedly calls the cutting lemma on the remaining graph with different parameters. It also cleans up the returned graph and runs itself recursively on the returned balls.

The analysis of the edge cutting probability can be roughly split into two parts: first, one bounds the edge cutting probability in one invocation of the cutting lemma. Afterwards, one bounds the edge cutting probability over all the invocations of the cutting lemma that an edge participates in throughout the algorithm.

The most important part of the algorithm is how $v$ and $r$ are chosen in the cutting lemma. The following description is sufficient for our purposes. The cutting lemma receives two parameters, $r_0$ and $r_1$. Each ball center $v$ is sampled at random from the set of "viable" center vertices, then $r$ is sampled as $r \sim r_0 + \mathsf{Geom}(p)$, where $p$ is chosen such that $r_0 \leq r \leq r_1$ with a sufficiently high probability. Whenever $r > r_1$, this invocation of the cutting lemma fails, and the whole main algorithm restarts.

**Our change.** Our only change to the algorithm is as follows: we change $p \leftarrow 2p$ and sample $r \sim r_0 + \lfloor \mathsf{Exp}(p) \rfloor$ with rejection until $r < r_1$. In other words, we sample from a truncated exponential distribution. The cutting procedure never fails and the main algorithm never restarts.

The modified cutting procedure is given in Algorithm 1. For completeness, we also restate the main algorithm in Algorithm 2. For the rest of this section, we fix the following parameters, which are the same as in [Bri+25]:

- $L = \lceil \log\log m \rceil + 1$,
- $\delta = \frac{1}{\log^{10} m}$,

**Algorithm 1** Modified implementation of the cutting procedure of [Bri+25].

1. Run Cohen's algorithm (with parameter $\epsilon = \frac{1}{8}$) on $G$ to compute approximations $b_1(v)$ satisfying that $\frac{7}{8} \cdot |B^+(v, r_1)| \leq b_1(v) \leq \frac{9}{8} \cdot |B^+(v, r_1)|$. Mark all nodes $v$ satisfying that $b_1(v) \leq \frac{9}{8} \cdot \frac{m}{s_1}$ as *good* and all other nodes as *bad.*
2. Repeat the following steps $\log n + 1$ times:
   2.1. Repeat the following steps $s_0 \cdot 100 \log n$ times:
      2.1.1. Among the nodes that are marked good, pick a uniformly random node $v$.
      2.1.2. Test whether $|B^+(v, r_0)| \geq \frac{1}{2} \cdot \frac{m}{s_0}$, and otherwise continue with the next iteration of 2.1.
      2.1.3. Repeatedly sample $r \sim r_0 + \lfloor \mathsf{Exp}(p) \rfloor$ (where $p = \frac{2 \ln(2s_0/\delta)}{r_1 - r_0}$) until $r < r_1$.
      2.1.4. Compute $B^+(v, r)$. Update $S \leftarrow S \cup \delta^+(B^+(v, r))$, mark the nodes in $B^+(v, r)$ as bad, and remove these nodes from $G$.
   2.2. Run Cohen's algorithm (with parameter $\epsilon = \frac{1}{8}$) on $G$ to compute approximations $b_0(v)$ satisfying that $\frac{7}{8} \cdot |B^+(v, r_0)| \leq b_0(v) \leq \frac{9}{8} \cdot |B^+(v, r_0)|$. Mark all nodes $v$ satisfying that $b_0(v) < \frac{7}{8} \cdot \frac{m}{s_0}$ as bad.
3. Let $R$ denote the set of remaining nodes in $G$, and return $(S, R)$.

---

- $r_0 := 0$ and $r_\ell := r_{\ell-1} + \frac{D}{2^{L-\ell+3}} + \frac{D}{4L}$ (for $1 \leq \ell \leq L$),
- $s_\ell := \min(2^{2^{L-\ell}}, m + 1)$ (for $0 \leq \ell \leq L$).

We also analyze the algorithm via a different approach, albeit with a similar general idea. Instead of analyzing the cutting lemma and the main algorithm separately, we establish a single recursion that deals with all invocations of the cutting lemma. As a bonus, we manage to get rid of the need to analyze conditional probabilities.

We remark that for a slightly easier analysis, during the recursive calls of the algorithm where we have $m' < m$ edges, we keep using the same value of $\delta = (\log m)^{-10}$ instead of changing it to $\delta' = (\log m')^{-10}$. We set the parameters of our algorithm such that all failure events happen with $1/\operatorname{poly}(m)$ probability, instead of $1/\operatorname{poly}(m')$. This only changes some $\log m'$ factors to $\log m$ in the time complexity of the algorithm. Concretely, we need to explicitly use the initial $m$ and $n$ in two places: for Cohen's algorithm, and in step 2.1 of Algorithm 1, so that they all succeed "with high probability" $1 - 1/\operatorname{poly}(m)$.

Before focusing on the cutting probability, let us check that the rest of the original analysis still applies after our modification, including its correctness and its running time complexity. For this we need to look at both changes that we do:

- Changing the way $r$ is sampled. Clearly, the absence of restarts cannot increase the algorithm's time complexity. All the analysis except for the edge cutting probability does not rely on how $r$ is sampled, except that $r \in [r_{\ell-1}, r_\ell)$, which it still is after our change.
- Changing some factors from $\log n'$ and $\log m'$ (where $n'$ and $m'$ are the number of nodes and edges in the current recursive call) to $\log n$ and $\log m$ (where $n$ and $m$ are the number of nodes and edges in the original graph). This does not worsen the time complexity bound, as the original analysis anyway eventually has $\log m$ and $\log n$ factors.

**Algorithm 2** The directed LDD algorithm of [Bri+25].

1. Initially let $S \subseteq E$ be the set of edges of length at least $\frac{D}{4L}$, and remove these edges from $G$
2. For $\ell \leftarrow L, \ldots, 1$:
   2.1. Run Algorithm 1 on $G$ with parameters $\delta, r_{\ell-1}, r_\ell, s_{\ell-1}, s_\ell$, and let $S_\ell^+, R_\ell^+$ denote the resulting sets.
   2.2. Compute the strongly connected components in $(G \setminus S_\ell^+)[V \setminus R_\ell^+]$. Recur on each such component and add the recursively computed cut edges to $S$.
   2.3. Update $G \leftarrow G[R_\ell^+]$ and $S \leftarrow S \cup S_\ell^+$.
   2.4. Run Algorithm 1 on $\mathrm{rev}(G)$ with parameters $\delta, r_{\ell-1}, r_\ell, s_{\ell-1}, s_\ell$, and let $S_\ell^-, R_\ell^-$ denote the resulting sets.
   2.5. Compute the strongly connected components in $(G \setminus S_\ell^-)[V \setminus R_\ell^-]$. Recur on each such component and add the recursively computed cut edges to $S$.
   2.6. Update $G \leftarrow G[R_\ell^-]$ and $S \leftarrow S \cup S_\ell^-$.

More details can be found in [Bri+25].

### 4.2 Analysis of Modified [Bri+25]'s Algorithm

We start the analysis by investigating the truncated exponential distribution $r$ is sampled from:

**Definition 4.2.** *For $p > 0$, and integers $a < b$, the truncated exponential distribution $\mathsf{TrunExp}(p, a, b)$ is a distribution defined on $x \in \{a, a+1, \ldots, b-1\}$ as follows:*

$$\Pr_{\mathbf{x} \sim \mathsf{TrunExp}(p,a,b)}(\mathbf{x} = x) = \frac{\exp(-p(x-a)) - \exp(-p(x-a+1))}{1 - \exp(-p(b-a))}.$$

*For $x$ outside of this range, the probability is 0. We also define $\varepsilon = \exp(-p(b-a)/2)$. Then we can equivalently write:*

$$\Pr_{\mathbf{x} \sim \mathsf{TrunExp}(p,a,b)}(\mathbf{x} = x) = \frac{\exp(-p(x-a)) - \exp(-p(x-a+1))}{1 - \varepsilon^2}.$$

*Equivalently, $\mathsf{TrunExp}(p, a, b)$ is the distribution that one gets when repeatedly sampling $x \sim a + \lfloor \mathsf{Exp}(p) \rfloor$ and keeping the first outcome that satisfies $x \in [a, b)$.*

The following properties will be useful later in the analysis:

**Remark 1.** *Let $\mathbf{x} \sim \mathsf{TrunExp}(p, a, b)$. For $a \leq x < x + c \leq b$, we have*

$$\Pr(\mathbf{x} \in [x, x+c)) = \frac{e^{-p(x-a)} - e^{-p(x+c-a)}}{1 - \varepsilon^2} = e^{-p(x-a)} \cdot \frac{1 - e^{-pc}}{1 - \varepsilon^2}.$$

*Furthermore, if $c \geq (b-a)/2$, we have*

$$\frac{1 - e^{-pc}}{1 - \varepsilon^2} \geq 1 - e^{-pc} \geq 1 - e^{-(b-a)/2} = 1 - \varepsilon.$$

*and thus*

$$\Pr(\mathbf{x} \in [x, x+c)) \geq e^{-p(x-a)} \cdot (1-\varepsilon) \geq \exp(-p(x-a) - 2\varepsilon),$$

*for sufficiently small $\varepsilon$. Finally, for $a \leq x < b-1$, we have*

$$\Pr(\mathbf{x} = x+1) = \frac{e^{-p(x+1-a)} - e^{-p(x+2-a)}}{1-\varepsilon^2} = e^{-p} \cdot \frac{e^{-p(x-a)} - e^{-p(x+1-a)}}{1-\varepsilon^2} = e^{-p} \cdot \Pr(\mathbf{x} = x).$$

For iteration $\ell$ of Algorithm 2, in the invoked Algorithm 1, we sample $r \sim \mathsf{TrunExp}(p_\ell, r_{\ell-1}, r_\ell)$ for $p_\ell = \frac{2\ln(2s_{\ell-1}/\delta)}{r_\ell - r_{\ell-1}}$.

In [Bri+25], it is shown that for iteration $\ell$ of Algorithm 2, each invocation of Algorithm 1 cuts at most $2s_{\ell-1}$ balls. For the purposes of our analysis, assume that exactly $2s_{\ell-1}$ balls are cut, by padding with dummy balls of zero size.

The main idea of the analysis is simple: we have a lower bound on the probability that, assuming the algorithm has made some number of cuts without cutting our edge set, it will not cut the edge set later. We analyze this probability directly by looking at the current ball: with some probability the edge set survives the cut by this ball, a part of it ends up on the inside of the ball and it survives there, and the rest ends up on the outside of the ball and it survives there too. An important step in the analysis is to show that we do not need to analyze all the possible ways the edge set can be split, as the worst case happens either when all of the edge set ends up on the inside or on the outside of the cut, in a very specific way. After we show this, the rest of the proof boils down to showing a straightforward recurrence relation.

**Definition 4.3.** *For all $i \in \{0, \dots, 2L\}$, $0 \leq j \leq 2s_{\lceil L - i/2 \rceil - 1}$, $m \geq 1$, $d \geq 0$, we define the quantity $P_{i,j}(m,d)$ as follows:*

*Suppose Algorithm 2 is run on any graph $G$ with at most $m$ edges. Algorithm 1 has been invoked $i$ times[4], and in the $(i+1)$-th invocation, we have so far cut $j$ balls. Let $\Gamma$ be any set of edges with total length $\leq d$ that have not yet been cut or gone into recursion. Conditioned on this, let $p$ be the probability that we will not cut $\Gamma$ in the rest of the algorithm. Then $P_{i,j}(m,d)$ is defined as the minimum such $p$ over all $G$, $\Gamma$ and possible runs of the algorithm that have led to this state.*

*Furthermore, define*

$$Q_{i,j}(m,d) = \exp\left(-\frac{d}{D} \cdot 640L \log m - 3L \log_{4/3} m \cdot \delta - \left(2L - i - \frac{j}{2s_{\ell-1}}\right)\delta\right),$$

*where $\ell = \lceil L - i/2 \rceil$. As a special case, $Q_{i,j}(m,0) = 1$.*

**Lemma 4.4.** *For any $i, j, m$, and $d \leq \frac{D}{8L}$, we have $P_{i,j}(m,d) \geq Q_{i,j}(m,d)$.*

In the rest of this section, we prove Lemma 4.4, which will be used for the proof of Theorem 2.1, by induction with increasing $m, d$ and decreasing $i, j$. For any $i$, we have $\ell = \lceil L - i/2 \rceil$, which is the value of $\ell$ used by the algorithm for the $(i+1)$-th invocation of the cutting lemma.

Let us start with the base cases: the claim holds trivially for $m \leq 1$ or $d = 0$ or $(i,j) = (2L, 0)$, since then $P_{i,j}(m,d)$ is in fact 1, as the algorithm terminates without any cut. Moreover, for $j = 2s_{\ell-1}$, we have $P_{i,j}(m,d) = P_{i+1,0}(m,d)$ and $Q_{i,j}(m,d) = Q_{i+1,0}(m,d)$ and we can thus rely on the inductive hypothesis.

Now, let us have $m > 1$, $d > 0$, $i < 2L$, $j < 2s_{\ell-1}$. Without loss of generality, we only consider the case of cutting out-balls. Fix the edge set $\Gamma$ of total length $d$ that we want to preserve. Fix $v$ the center of the $(j+1)$-th ball. We sample the radius $r \in [r_{\ell-1}, r_\ell)$. The set $\Gamma$ determines which

[4] We count the invocations on $G$ and $\mathrm{rev}(G)$ in one iteration as two invocations.

values of $r$ are "good" or "bad", that is, if we sample this particular $r$ and make a cut, whether the set $\Gamma$ will survive or not. This can be formalized as follows: let $\mathbf{d}$ be a vector indexed by $r \in [r_{\ell-1}, r_\ell)$ such that $d_r$ counts the total number of edges of $\Gamma$ that would be cut by $B^+(v, r)$, that is, edges $e = (x, y)$ with $\mathrm{dist}_G(v, x) \le r < \mathrm{dist}_G(v, y)$. Note that a value of $r$ is good if and only if $d_r = 0$, i.e., $r \notin \mathrm{supp}(\mathbf{d})$. We have $\|\mathbf{d}\|_1 = d$ and $|\mathrm{supp}(\mathbf{d})| \le d$. The probability that $\Gamma$ survives is equal to

$$\sum_{r \notin \mathrm{supp}(\mathbf{d})} \Pr(\mathbf{r} = r \wedge \Gamma \text{ survives the rest of the algorithm}).$$

Let us express one term of this sum. Assuming the sampled radius is $r \notin \mathrm{supp}(\mathbf{d})$, let $B = B^+(v, r)$ be the ball being cut. The set $\Gamma$ gets split into two: $\Gamma \cap B$, of total length $d_{<r} = \sum_{r'<r} d_{r'}$, ends up inside $B$, while $\Gamma \setminus B$ of total length $d_{>r} = d - d_{<r}$ ends up on the outside. $\Gamma$ will survive the rest of the algorithm if and only if $\Gamma \cap B$ survives the recursion and $\Gamma \setminus B$ survives the rest of the algorithm. It is shown in [Bri+25] that the recursive call is on a graph with at most $\frac{3}{2} \cdot \frac{m}{s_\ell}$ edges. Thus, due to the independence of these two events, they occur with probability at least

$$P_{0,0}\left(\frac{3m}{2s_\ell}, d_{<r}\right) \cdot P_{i,j+1}(m, d_{>r})$$

Summing over $r$, we know that the probability of $\Gamma$ surviving the rest of the algorithm is at least

$$\sum_{r \notin \mathrm{supp}(\mathbf{d})} \Pr(r) \cdot P_{0,0}\left(\frac{3m}{2s_\ell}, d_{<r}\right) \cdot P_{i,j+1}(m, d_{>r})$$

and by the inductive hypothesis, this is at least

$$\sum_{r \notin \mathrm{supp}(\mathbf{d})} \Pr(r) \cdot Q_{0,0}\left(\frac{3m}{2s_\ell}, d_{<r}\right) \cdot Q_{i,j+1}(m, d_{>r}).$$

Fix $i, j, m, d$ and denote the above expression by $\mathcal{E}(\mathbf{d})$. Necessarily $P_{i,j}(m, d) \ge \mathcal{E}(\mathbf{d})$, since the former is the probability of $\Gamma$ surviving in the worst possible $G$ in the worst possible run of the algorithm so far, and the latter is a lower bound on the probability of $\Gamma$ surviving for a given $\mathbf{d}$. Thus, the proof will be completed if we prove that $\mathcal{E}(\mathbf{d}) \ge Q_{i,j}(m, d)$ no matter what $\mathbf{d}$ is.

The rest of the proof will be split in two parts: first we prove that we can without loss of generality assume that $\mathbf{d}$ is of one of two special vectors, and then we prove the inequality for both of them.

**Finding the minimizer of $\mathcal{E}(\mathbf{d})$.** Define $\mathbf{d}^{\leftarrow}$ as the 0-1 vector with $\mathrm{supp}(\mathbf{d}^{\leftarrow}) = \{r_{\ell-1}, \ldots, r_{\ell-1} + d - 1\}$. Similarly, define $\mathbf{d}^{\rightarrow}$ as the 0-1 vector with $\mathrm{supp}(\mathbf{d}^{\rightarrow}) = \{r_\ell - d, \ldots, r_\ell - 1\}$. We show that one of $\mathbf{d}^{\leftarrow}, \mathbf{d}^{\rightarrow}$ is always a minimizer of $\mathcal{E}(\mathbf{d})$. We show this in two steps: first we show that a non-0-1 vector is never a minimizer, then we show that any 0-1 vector can be gradually transformed into either $\mathbf{d}^{\leftarrow}$ or $\mathbf{d}^{\rightarrow}$ by a series of local changes that do not increase the value of $\mathcal{E}(\mathbf{d})$.

First assume $\mathbf{d}$ is not a 0-1 vector. Let $i$ be any index such that $d_i > 1$ and let $j$ be the closest index to $i$ such that $d_j = 0$ (so all coordinates between $d_i, d_j$ are nonzero). Construct a new $\mathbf{d}'$ such that $d'_i = d_i - 1$, $d'_j = 1$ and $d'_k = d_k$ for all other $k$. Note that $\mathrm{supp}(\mathbf{d}') = \mathrm{supp}(\mathbf{d}) \setminus \{j\}$ and that for all $r \in \mathrm{supp}(\mathbf{d}')$, we have $d_{<r} = d'_{<r}$ and $d_{>r} = d'_{>r}$. In other words, the terms present in both $\mathcal{E}(\mathbf{d})$ and $\mathcal{E}(\mathbf{d}')$ are the same, except that $\mathcal{E}(\mathbf{d}')$ lacks the term for $r = j$. Thus, $\mathcal{E}(\mathbf{d}) \ge \mathcal{E}(\mathbf{d}')$.

Now we know that the minimizing $\mathbf{d}$ is a 0-1 vector. Consider two 0-1 vectors $\mathbf{d}, \mathbf{d}'$ that are nearly identical, except that for some $r$, we have $d_r = d'_{r+1} = 0$ and $d_{r+1} = d'_r = 1$. We can write:

$$\begin{aligned}\mathcal{E}(\mathbf{d}) - \mathcal{E}(\mathbf{d}') = &\Pr(r) \cdot Q_{0,0}\left(\frac{3m}{2s_\ell}, d_{<r}\right) \cdot Q_{i,j+1}(m, d_{>r}) \\ &- \Pr(r+1) \cdot Q_{0,0}\left(\frac{3m}{2s_\ell}, d_{<r} + 1\right) \cdot Q_{i,j+1}(m, d_{>r} - 1),\end{aligned}$$

where we used $d'_{<r+1} = d_{<r} + 1$ and $d'_{>r+1} = d_{>r} - 1$, and the fact that all the other terms cancel out. Now let

$$\alpha = \frac{\Pr(r) \cdot Q_{0,0}\left(\frac{3m}{2s_\ell}, d_{<r}\right) \cdot Q_{i,j+1}(m, d_{>r})}{\Pr(r+1) \cdot Q_{0,0}\left(\frac{3m}{2s_\ell}, d_{<r} + 1\right) \cdot Q_{i,j+1}(m, d_{>r} - 1),}.$$

Clearly, $\mathcal{E}(\mathbf{d}) > \mathcal{E}(\mathbf{d}')$ if and only if $\alpha > 1$. After we expand all terms and cancel, we are left with

$$\alpha = \exp(p_\ell) \cdot \frac{\exp(-\frac{1}{D} \cdot 640L \log m)}{\exp\left(-\frac{1}{D} \cdot 640L \log\left(\frac{3m}{2s_\ell}\right)\right)}.$$

where we use Remark 1 to show that

$$\frac{\Pr(\mathbf{r} = r)}{\Pr(\mathbf{r} = r+1)} = \exp(p_\ell).$$

Note that the value of $\alpha$ does not depend on $r$, $\mathbf{d}$, or $\mathbf{d}'$. Suppose $\alpha \geq 1$. Then $\mathcal{E}(\mathbf{d}) \geq \mathcal{E}(\mathbf{d}')$. We can now repeat the same process by finding a subsequence $(0,1)$ in $\mathbf{d}'$ and changing it to $(1,0)$ to construct $\mathbf{d}''$. Since $\alpha$ does not depend on $r$, $\mathbf{d}'$ or $\mathbf{d}''$, the same analysis concludes that $\mathcal{E}(\mathbf{d}') \geq \mathcal{E}(\mathbf{d}'')$. We can continue this process until there is no $(0,1)$ subsequence in the vector, that is, until we end up with $\mathbf{d}^{\leftarrow}$. What we have shown is that assuming $\alpha \geq 1$, we have $\mathcal{E}(\mathbf{d}) \geq \mathcal{E}(\mathbf{d}^{\leftarrow})$, and this holds for all $\mathbf{d}$. Analogously, if instead $\alpha < 1$, we can do the same operation in reverse, changing $(1,0)$-subsequences to $(0,1)$ until there are no more $(1,0)$-subsequences in the vector. This way, we show that if $\alpha < 1$, then $\mathcal{E}(\mathbf{d}) > \mathcal{E}(\mathbf{d}^{\rightarrow})$ for all $\mathbf{d}$.

**Bounding $Q_{i,j}(m,d)$.** All that remains now is to show that $\mathcal{E}(\mathbf{d}^{\leftarrow}) \geq Q_{i,j}(m,d)$ and $\mathcal{E}(\mathbf{d}^{\rightarrow}) \geq Q_{i,j}(m,d)$. We start with the former. Recall that $\mathrm{supp}(\mathbf{d}^{\leftarrow}) = \{r_{\ell-1}, \ldots, r_{\ell-1} + d - 1\}$. For $r \notin \mathrm{supp}(\mathbf{d}^{\leftarrow})$, we have $d^{\leftarrow}_{<r} = d$, $d^{\leftarrow}_{>r} = 0$. The whole expression thus simplifies to:

$$\begin{aligned}\mathcal{E}(\mathbf{d}^{\leftarrow}) &= \sum_{r \geq r_{\ell-1}+d} \Pr(r) \cdot Q_{0,0}\left(\frac{3m}{2s_\ell}, d\right) \cdot \underbrace{Q_{i,j+1}(m, 0)}_{=1} \\ &= \left(\sum_{r \geq r_{\ell-1}+d} \Pr(r)\right) \cdot Q_{0,0}\left(\frac{3m}{2s_\ell}, d\right)\end{aligned}$$

Recall that $d \leq (r_\ell - r_{\ell-1})/2$ by our assumption and $\varepsilon = \delta/2s_{\ell-1} = o(1)$. Thus, we can use Remark 1 to bound:

$$\sum_{r \geq r_{\ell-1}+d} \Pr(r) = \Pr(\mathbf{r} \in [r_{\ell-1} + d, r_\ell)) \geq \exp(-p_\ell d - 2\varepsilon) \geq \exp(-p_\ell d - L\delta),$$

where in the last step, we used a very crude bound on $2\varepsilon = \delta/s_{\ell-1} \le L\delta$.

We also expand:

$$\begin{aligned} Q_{0,0}\left(\frac{3m}{2s_\ell}, d\right) &= \exp\left(-\frac{d}{D}\cdot 640L\log\frac{3m}{2s_\ell} - 3L\log_{4/3}\left(\frac{3m}{2s_\ell}\right)\cdot\delta - 2L\delta\right) \\ &\ge \exp\left(-\frac{d}{D}\cdot 640L\log\left(\frac{m}{2^{2^{L-\ell-1}}}\right) - 3L\log_{4/3}\left(\frac{3}{4}m\right)\cdot\delta - 2L\delta\right), \\ &= \exp\left(-\frac{d}{D}\cdot 640L\left(\log m - 2^{L-\ell-1}\right) - 3L(\log_{4/3} m - 1)\delta - 2L\delta\right), \end{aligned}$$

where we use $\frac{3m}{2s_\ell} \le \frac{3}{4}m$ and $\frac{3m}{2s_\ell} \le m/2^{2^{L-\ell-1}}$.

Before we put everything together, let us also bound $p_\ell d$ identically to [Bri+25]:

$$\begin{aligned} p_\ell d &= \frac{2d\ln(2s_{\ell-1}/\delta)}{r_\ell - r_{\ell-1}} \\ &\le \frac{2d\ln(2^{2^{L-\ell+1}}\cdot(\log m)^{10})}{\frac{D}{2^{L-\ell+3}} + \frac{D}{4L}} \\ &\le \frac{2d}{D}\cdot\frac{2^{L-\ell+1} + 10L}{\max\left\{\frac{1}{2^{L-\ell+3}}, \frac{1}{4L}\right\}} \\ &\le \frac{2d}{D}\cdot(2^{L-\ell+3} + 10L)\cdot\min\left\{2^{L-\ell+3}, 10L\right\} \\ &\le \frac{d}{D}\cdot 2^{L-\ell}\cdot 320L. \end{aligned}$$

In the last step we used that $(a+b)\min\{a,b\} \le 2\max\{a,b\}\min\{a,b\} = 2ab$.

Now we can write:

$$\begin{aligned} \log\mathcal{E}(\mathbf{d}^{\leftarrow}) &\ge -p_\ell d - L\delta + \log Q_{0,0}\left(\frac{3m}{2s_\ell}, d\right) \\ &\ge -\frac{d}{D}\cdot 2^{L-\ell}\cdot 320L - L\delta - \frac{d}{D}\cdot 640L\left(\log m - 2^{L-\ell-1}\right) - 3L(\log_{4/3} m - 1)\delta - 2L\delta \\ &= -\frac{d}{D}\cdot 640L\log m - 3L\log_{4/3} m\cdot\delta \\ &= \log Q_{2L,0}(m,d) \ge \log Q_{i,j}(m,d). \end{aligned}$$

Similarly, we can calculate $\mathcal{E}(\mathbf{d}^{\rightarrow})$. Recall that $\mathrm{supp}(\mathbf{d}^{\rightarrow}) = \{r_\ell - d, \dots, r_\ell - 1\}$.

$$\begin{aligned} \mathcal{E}(\mathbf{d}^{\rightarrow}) &= \sum_{r<r_\ell-d}\Pr(r)\cdot\underbrace{Q_{0,0}\left(\frac{3m}{2s_\ell}, 0\right)}_{=1}\cdot Q_{i,j+1}(m,d) \\ &= \left(\sum_{r<r_\ell-d}\Pr(r)\right)\cdot Q_{i,j+1}(m,d) \end{aligned}$$

Now we again start by bounding the first term using Remark 1:

$$\Pr(\mathbf{r}\in[r_{\ell-1}, r_\ell - d)) \ge \exp(-\varepsilon).$$

Recall that $\varepsilon = \delta/(2s_{\ell-1})$. We can directly write

$$\log\mathcal{E}(\mathbf{d}^{\rightarrow}) \ge -\varepsilon + \log Q_{i,j+1}(m,d)$$

$$
\begin{aligned}
&\geq -\frac{\delta}{2s_{\ell-1}} - \frac{d}{D} \cdot 640L \log m - 3L \log_{4/3} m \cdot \delta - \left(2L - i - \frac{j+1}{2s_{\ell-1}}\right)\delta \\
&= -\frac{d}{D} \cdot 640L \log m - 3L \log_{4/3} m \cdot \delta - \left(2L - i - \frac{j}{2s_{\ell-1}}\right)\delta \\
&= \log Q_{i,j}(m, d).
\end{aligned}
$$

Finally, we are ready to prove Theorem 4.1.

*Proof of Theorem 4.1.* As discussed previously, all invocations of Cohen's algorithm and step 2.1 of Algorithm 1 can be implemented to succeed with high probability $1 - 1/\operatorname{poly}(m)$. Together with Lemma 4.4 and by union bound, any subset $\Gamma$ of edges with total length $d_\Gamma$ is not cut with probability at least $Q_{0,0}(m, d_\Gamma) - 1/\operatorname{poly}(m)$. The extra $1/\operatorname{poly}\log(m)$ term in the exponent is subsumed if $d_\Gamma/D \cdot \log m \log\log m$ is at least some small constant. Otherwise, the desired bound already follows from [Bri+25]. The extra $1/\operatorname{poly}(m)$ loss can always be ignored as we only consider $d_\Gamma \leq \epsilon D/\log\log m$ for sufficiently small $\epsilon$. Note that guarantees for edges and paths are direct corollaries of guarantees for subsets of edges.

To see the independence property, we first note that distance between any two vertices in the remaining graph can only increase after each cutting step, and that all balls being cut have radius at most $D$. Then observe that in each cutting step, independent randomness is used and either $\Gamma$ or $X \cup Y$ can never be cut due to the assumption. Thus, the same probability bound for $\Gamma$ being cut also holds if it is conditioned on $X, Y$.

The theorem follows as our modification does not affect the diameter property and the running time proved in [Bri+25]. □

# 5 Strong Bounds for Modified [Li25]'s Algorithm

In this section, we modify [Li25]'s algorithm to prove the following result, which is a more formal version of Theorem 2.2, with probability guarantees for edges at the same time. The algorithm is presented in Section 5.1, and Section 5.2 constitutes the proof.

**Theorem 5.1.** *Let $\epsilon > 0$ be a sufficiently small constant. There is a randomized algorithm that, given any directed graph $G = (V, E)$ with $n$ vertices and $m$ edges, diameter parameter $D = \operatorname{poly}(n)$, and separation parameter $d$, returns an ordered (full) clustering $\sigma = (C_1, \ldots, C_k)$ of $V$ and boolean vertex marks $\operatorname{mk}(\cdot)$, satisfying all of the following.*

- **Clustering probability:** *Each vertex $v$ is* clustered*, i.e., $\operatorname{mk}(v) = 0$, with probability at least $\exp(-d/D \cdot \mathcal{O}(\log n \log\log n))$ if $d \leq \epsilon D/\log\log n$.*
- **Edge cutting probability:** *Each edge $e = (u, v)$ is* cut*, i.e., $u \in C_i$ and $v \in C_j$ for $i > j$, with probability at most $1 - \exp(-d_e/D \cdot \mathcal{O}(\log n \log\log n))$ if $d_e \leq \epsilon D/\log\log n$.*
- **Separation property:** *For any* clustered *vertices $u \in C_i$ and $v \in C_j$ with $i > j$, it holds that $\operatorname{dist}_G(u, v) > d$.*
- **Diameter property:** *Each $C_i$ is strongly connected and has diameter at most $D$.*
- **Running time:** *The algorithm runs in $\mathcal{O}((m + n \log\log n) \log^2 n)$ time with high probability on graphs with polynomially bounded, integral edge lengths.*

## 5.1 Overview of [Li25]'s Algorithm and Our Modification

For a union $B^+ = \bigcup_{i=1}^{k} B_G^+(t_i, r)$ of out-balls, define $\mathrm{dist}_{B^+}(v) = \min_{i\in[k]} \mathrm{dist}_G(t_i, v)$. Define $\mathrm{dist}_{B^-}(\cdot)$ for a union $B^-$ of in-balls similarly. Note that $B^\pm = \{v \mid \mathrm{dist}_{B^\pm}(v) \le r\}$.

The algorithm is shown in Algorithm 3.

**Remark 2.** *Algorithm 3 is essentially [Li25]'s algorithm with following modifications.*

1. *The separation parameter $d$ is used only for clustering, while having nothing to do with cutting. Intuitively, each time a cut is executed, vertices within distance $d$ to the boundary are marked and excluded from clustering.*
2. *Exactly one of iterations $i(\pm)$ is executed in an alternating way, while not affecting correctness. Consequently, heavy vertex elimination is not needed in cases not having both in-heavy and out-heavy vertices.*
3. *In case (b) of having both in-heavy and out-heavy vertices, following changes are made to simplify the algorithm as well as analysis.*
   (a) *Both $B_G^-$ and $B_G^+$ are computed, which are disjoint by the condition of case (b), so the recursive instance has at most $m/2$ edges.*
   (b) *Cutting $B^+$ eliminates in-heavy vertices, so step 6 can be executed. Similarly, cutting $B^-$ eliminates out-heavy vertices, so step 7 can be executed.*

## 5.2 Analysis of Modified [Li25]'s Algorithm

We show that Algorithm 3 satisfies all of the requirements of Theorem 5.1. In the rest of this section, assume $\epsilon > 0$ is a sufficiently small constant and $D = \mathrm{poly}(m)$.

**Clustering probability.** Assume $d \le \epsilon D/\log\log(mD)$. We use the following lemma of [Li25].

**Lemma 5.2** ([Li25]'s Lemma 5)**.** *For $i \in [L]$, with probability at least $1 - (mD)^{-3}$, all remaining vertices $v \in U$ satisfy $\mathrm{vol}_G(B_G^\pm(v, a_i) \cap U) \le 2m/2^{2^i}$ simultaneously after iteration $i(\mp)$.*

We also use the following technical lemma. Its proof is deferred to Appendix A.

**Lemma 5.3.** *For integers $r, d \ge 1$ such that $r \ge 2d$, and reals $k, k_1, \ldots, k_r \ge 0$ such that $k_1 \ge 1$ and $k_1 + \ldots + k_r = k$, it holds that*

$$\frac{1}{r} \cdot \sum_{i=1}^{r} \frac{\sum_{j=\max(i-d+1,1)}^{i} k_i}{\sum_{j=1}^{i} k_i} \le 1 - \left(1 - \frac{1}{\lfloor r/d \rfloor}\right) \cdot k^{-\frac{1}{\lfloor r/d \rfloor - 1}}.$$

At a high level, the proof of clustering probability follows a similar approach to [Li25] with two crucial differences.

- A much more delicate analysis is needed to derive an exponential bound on the probability of not marking a vertex in each iteration. The argument still relies on the random order of sampled vertices, while additionally exploiting the important observation that only $a_{i-1} - a_i$ possibilities of the distance may be relevant. Even with $k \gg a_{i-1} - a_i$ sampled vertices, we can obtain a much tighter bound than the one in [Li25] that depends linearly on $\ln k$. We elaborate on this in Lemma 5.4.

**Algorithm 3** Modified [Li25]'s Algorithm

**Input:** graph $G = (V, E)$ with $n$ vertices and $m$ edges, diameter $D$, separation $d$.

**Output:** ordered clustering $\sigma$, boolean marks $\mathrm{mk}(\cdot)$.

1. Let $L = \log\log m$, and $a_i = D/8 - \sum_{j=1}^{i} D/16 \cdot \max(1/L, 2^{-i})$ for $i \in [0, L]$.
2. Set $U \leftarrow V$, $\sigma^+, \sigma^- \leftarrow \perp$, and $\mathrm{mk}(v) \leftarrow 0$ for $v \in V$.
3. If $m \leq 1$, return the singleton clustering $\sigma$ induced by a topological order of $G$, and $\mathrm{mk}(\cdot)$.
4. Label each vertex as either in-heavy or in-light such that $|E[B_G^-(v, D/8]| \geq m/2$ for each in-heavy vertex $v$, and that $|E[B_G^-(v, D/8]| \leq 3m/4$ for each in-light vertex $v$.
5. Label each vertex as either out-heavy or out-light such that $|E[B_G^+(v, D/8]| \geq m/2$ for each out-heavy vertex $v$, and that $|E[B_G^+(v, D/8]| \leq 3m/4$ for each out-light vertex $v$.
6. If there is no in-heavy vertex,
   (a) For $i = 1, \ldots, L$, execute iteration $i(-)$ for odd $i$, and iteration $i(+)$ for even $i$.

   **Iteration $i(-)$:**
   i. Sample $r_i^- \in (a_i, a_{i-1}]$, and sample each vertex $v \in U$ with probability $\min(2\deg(v)/m \cdot 2^{2^i} \ln(mD), 1)$. Let $S_i^-$ be the set of sampled vertices.
   ii. For $v \in S_i^-$ in random order,
   (A) Set $\mathrm{mk}(v) \leftarrow 1$ for $v \in (B_G^-(v, r_i^- + d) \setminus B_G^-(v, r_i^- - d)) \cap U$.
   (B) Recurse on $G[B = B_G^-(v, r_i^-) \cap U]$. Let $\sigma', \mathrm{mk}'(\cdot)$ be returned by the recursion.
   (C) Set $U \leftarrow U \setminus B$, $\sigma^- \leftarrow \sigma^- \circ \sigma'$, and $\mathrm{mk}(v) \leftarrow \mathrm{mk}(v) \vee \mathrm{mk}'(v)$ for $v \in B$.

   **Iteration $i(+)$:**
   i. Sample $r_i^+ \in (a_i, a_{i-1}]$, and sample each vertex $v \in U$ with probability $\min(2\deg(v)/m \cdot 2^{2^i} \ln(mD), 1)$. Let $S_i^+$ be the set of sampled vertices.
   ii. For $v \in S_i^+$ in random order,
   (A) Set $\mathrm{mk}(v) \leftarrow 1$ for $v \in (B_G^+(v, r_i^+ + d) \setminus B_G^+(v, r_i^+ - d)) \cap U$.
   (B) Recurse on $G[B = B_G^+(v, r_i^+) \cap U]$. Let $\sigma', \mathrm{mk}'(\cdot)$ be returned by the recursion.
   (C) Set $U \leftarrow U \setminus B$, $\sigma^+ \leftarrow \sigma' \circ \sigma^+$, and $\mathrm{mk}(v) \leftarrow \mathrm{mk}(v) \vee \mathrm{mk}'(v)$ for $v \in B$.
7. Else if there is no out-heavy vertex,
   (a) For $i = 1, \ldots, L$, execute iteration $i(+)$ for odd $i$, and iteration $i(-)$ for even $i$.

**Algorithm 3** Modified [Li25]'s Algorithm (Continued)

8. Else if there are both in-heavy and out-heavy vertices,
   (a) If there is an in-heavy vertex $s$ and an out-heavy vertex $t$ such that $\text{dist}_G(s,t) \le D/4$,
      i. Sample $r \in (D/8, D/4]$.
      ii. Set $\text{mk}(v) \leftarrow 1$ for $v \in B_G^-(s, r+d) \setminus B_G^-(s, r-d)$.
      iii. Recurse on $G[B = V \setminus B_G^-(s,r)]$. Let $\sigma', \text{mk}'(\cdot)$ be returned by the recursion.
      iv. Set $\sigma^+ \leftarrow \sigma'$, and $\text{mk}(v) \leftarrow \text{mk}(v) \vee \text{mk}'(v)$ for $v \in B$.
      v. Set $\text{mk}(v) \leftarrow 1$ for $v \in (B_G^+(t, r+d) \setminus B_G^+(t, r-d)) \cap B_G^-(s,r)$.
      vi. Recurse on $G[B = B_G^-(s,r) \setminus B_G^+(t,r)]$. Let $\sigma', \text{mk}'(\cdot)$ be returned by the recursion.
      vii. Set $\sigma^- \leftarrow \sigma' \circ (B_G^-(s,r) \cap B_G^+(t,r))$, and $\text{mk}(v) \leftarrow \text{mk}(v) \vee \text{mk}'(v)$ for $v \in B$.
   (b) Else if $\text{dist}_G(s,t) > D/4$ for any in-heavy vertex $s$ and out-heavy vertex $t$,
      i. Sample $r \in (D/16, D/8]$
      ii. Let $B^-$ be the union of $B_G^-(t,r)$ over all out-heavy vertices $t$, and $B^+$ the union of $B_G^+(s,r)$ over all in-heavy vertices $s$.
      iii. If $|E[B^-]| \ge |E[B^+]|$,
         (A) Set $\text{mk}(v) \leftarrow 1$ for $v \in V$ if $\text{dist}_{B^+}(v) \in (r-d, r+d]$.
         (B) Recurse on $G[B^+]$. Let $\sigma', \text{mk}'(\cdot)$ be returned by the recursion.
         (C) Set $U \leftarrow V \setminus B^+$, $\sigma^+ \leftarrow \sigma'$, and $\text{mk}(v) \leftarrow \text{mk}(v) \vee \text{mk}'(v)$ for $v \in B^+$.
         (D) Execute step 6.
      iv. Else if $|E[B^+]| > |E[B^-]|$,
         (A) Set $\text{mk}(v) \leftarrow 1$ for $v \in V$ if $\text{dist}_{B^-}(v) \in (r-d, r+d]$ .
         (B) Recurse on $G[B^-]$. Let $\sigma', \text{mk}'(\cdot)$ be returned by the recursion.
         (C) Set $U \leftarrow V \setminus B^-$, $\sigma^- \leftarrow \sigma'$, and $\text{mk}(v) \leftarrow \text{mk}(v) \vee \text{mk}'(v)$ for $v \in B^-$.
         (D) Execute step 7.
9. Return $\sigma^- \circ \sigma^+$ and $\text{mk}(\cdot)$.

- To maintain such an exponential bound throughout the algorithm, independence between iterations and recursions must be carefully examined, as will be seen in Lemma 5.5.

**Lemma 5.4.** *For $i \in [L]$ and vertex $v \in U$ unmarked before iteration $i(\pm)$, $v$ remains unmarked after iteration $i(\pm)$ with probability at least* $\exp(-d/D \cdot \mathcal{O}(2^i \log\log(mD)))$.

*Proof.* We prove for iteration $i(+)$. Iteration $i(-)$ is symmetric. Let $k_j$ be the number of sampled vertices $t \in S_i^+$ such that $\mathrm{dist}_B(t, v) = j$, and $k = \sum k_j$. $v$ is marked by radius $r_i^+$ if and only if

- Radius $r_i^+$ is sampled with probability $1/(a_{i-1} - a_i)$.
- Some $t \in S_i^+$ with $\mathrm{dist}_G(t, v) \in (r_i^+ - d, r_i^+ + d]$ appears first among all $t' \in S_i^+$ with $\mathrm{dist}_G(t', v) \le r_i^+ + d$. This happens with probability[5]

$$\frac{\sum_{j=\max(r_i^+-d+1,0)}^{r_i^+ + d} k_j}{\sum_{j=0}^{r_i^+ + d} k_j}.$$

Altogether, $v$ is marked with probability

$$\frac{1}{a_{i-1} - a_i} \cdot \sum_{r_i^+ = a_i + 1}^{a_{i-1}} \frac{\sum_{j=\max(r_i^+-d+1,0)}^{r_i^+ + d} k_j}{\sum_{j=0}^{r_i^+ + d} k_j}.$$

Without loss of generality, we make the following assumptions without decreasing the probability.

**$\mathbf{k_0 = \cdots = k_{a_i+d} = 0}$:** Otherwise, we set $k_{a_i+d+1}$ to take the value of $\sum_{j=0}^{a_i+d+1} k_j$. Indeed, for each term of the above summation, its denominator is unchanged while its numerator may only increase. So the sum never decreases.

**$\mathbf{k_j = 0}$ for $\mathbf{j > a_{i-1} + d}$:** Otherwise, we set $k_{a_{i-1}+d}$ to take the value of $\sum_{j \ge a_{i-1}+d} k_j$. It may only increase the last term of the summation.

**$\mathbf{k_{a_i+d+1} \ge 1}$:** Otherwise, we set $k_{a_i+d+1} = 1$ and decrease $k_j$ by 1 for the smallest $j$ such that $k_j \ge 1$. This results in an increase from 0 to 1 for $\min(j - (a_i + d + 1), 2d)$ terms starting from $a_i + d + 1$ and a decrease by at most 1 for $\min(j - (a_i + d + 1), 2d)$ terms starting from $j$. The net effect is thus nonnegative.

With all these assumptions, the probability of $v$ being marked is at most

$$\frac{1}{a_{i-1} - a_i} \cdot \sum_{r_i^+ = a_i + 1}^{a_{i-1}} \frac{\sum_{j=\max(r_i^+-d+1,a_i+d+1)}^{r_i^+ + d} k_j}{\sum_{j=a_i+d+1}^{r_i^+ + d} k_j},$$

with $k = \sum_{j=a_i+d+1}^{a_{i-1}+d} k_j$. Applying Lemma 5.3 with parameters $a_{i-1} - a_i$ and $2d$, we get that the probability of $v$ not being marked is at least $\exp(-\mathcal{O}(d/(a_{i-1} - a_i) \cdot \ln k)$ because $1 - 1/z \ge \exp(-1/(z-1))$. Due to convexity, the same calculation as in [Li25]'s Lemma 6, taking expectation over $k$, concludes the proof. □

[5] $\frac{0}{0}$ is treated as 0.

**Lemma 5.5.** *Let $p(m)$ be such that for any vertex of any graph with at most $m$ edges, it is never marked during Algorithm 3 with probability at least $p(m)$. It holds that $p(m) \geq \exp(-d/D \cdot \mathcal{O}(\log m \log\log(mD)))$.*

*Proof.* We claim that $p(\cdot)$ satisfies the recursion

$$p(m) \geq \min_{\alpha \in [c_1, m]} \exp(-\frac{c_2 d}{D} \cdot \log \alpha \log\log(mD)) \cdot p\left(\frac{m}{\alpha}\right) - \frac{1}{(mD)^{c_3}},$$

for some constants $c_1, c_2 > 1$ and $c_3 \geq 2$, with the base case $p(1) = 1$. Assuming the claim for now, we can get

$$p(m) \geq \exp\left(-\frac{c_4 d}{D} \cdot \log m \log\log(mD)\right),$$

where $c_4 = c_2 + 1/\log c_1$, proving the lemma. Indeed, note that RHS evaluates to 1 for $m = 1$, which is satisfied by Algorithm 3 as no vertex is marked. For $m > 1$, observe that by induction,

$$\begin{aligned}
&\exp\left(-\frac{c_2 d}{D} \cdot \log \alpha \log\log(mD)\right) \cdot p\left(\frac{m}{\alpha}\right) - \frac{1}{(mD)^{c_3}} \\
&\quad \geq \exp\left(-\frac{c_2 d}{D} \cdot \log \alpha \log\log(mD)\right) \cdot \exp\left(-\frac{c_4 d}{D} \cdot \log \frac{m}{\alpha} \log\log \frac{mD}{\alpha}\right) - \frac{1}{(mD)^{c_3}} \\
&\quad \geq \exp\left(\frac{(c_4 - c_2)d}{D} \cdot \log \alpha \log\log(mD) - \frac{c_4 d}{D} \cdot \log \alpha \log\log(mD) - \frac{c_4 d}{D} \cdot \log \frac{m}{\alpha} \log\log(mD)\right) \\
&\qquad\quad - \exp(-c_3 \cdot \ln(mD)) \\
&\quad \geq \exp\left(\frac{(c_4 - c_2)d}{D} \cdot \log \alpha \log\log(mD) - \frac{c_4 d}{D} \cdot \log m \log\log(mD)\right) \cdot \left(1 - \exp\left(-\frac{c_3}{2} \cdot \ln(mD)\right)\right),
\end{aligned}$$

because the exponent of the first factor is at least

$$-c_4 \epsilon \log m \geq -c_3/2 \cdot \ln(mD),$$

for $\epsilon \leq c_3/(2c_4) \cdot \ln 2$. Also note that

$$\begin{aligned}
1 - \exp\left(-\frac{c_3}{2} \cdot \ln(mD)\right) &= 1 - \frac{1}{(mD)^{c_3/2}} \\
&\geq \exp\left(-\frac{1}{(mD)^{c_3/2} - 1}\right) \\
&\geq \exp\left(-\frac{1}{mD - 1}\right) \\
&\geq \exp\left(-\frac{1}{D}\right),
\end{aligned}$$

and that

$$(c_4 - c_2) \log \alpha \geq (c_4 - c_2) \log c_1 = 1.$$

Thus, all above together shows that the recursion is indeed satisfied.

It remains to prove the claim. By union bound, with probability at least $1 - 2L(mD)^{-3} \geq 1 - (mD)^{-2}$, all remaining vertices in $U$ satisfy the statement of Lemma 5.2 simultaneously in all $2L$ iterations. Also, as noted in [Li25], vertices can be efficiently labeled as heavy or light with high probability by [BCF23]. Its failure probability is subsumed by $(mD)^{-2}$ so long as $D = \text{poly}(m)$.

We assume there is no sampling or labeling failure in the rest of the proof. Let $c_0 > 0$ be such that the probability in Lemma 5.4 is at least $\exp(-c_0 d/D \cdot 2^i \log\log(mD))$.

In cases not having both in-heavy and out-heavy vertices, consider any unmarked vertex $v \in U$ at the beginning of iteration $i(\pm)$. By Lemma 5.4, it remains unmarked in the current iteration with probability at least $\exp(-c_0 d/D \cdot 2^i \log\log(mD))$. Conditioned on this event, it goes into recursion with some probability $q$, and continues to the next iteration with probability $1-q$.

Since we are lower bounding the probability, it is without loss of generality to assume either $q = 1$ or $q = 0$. If $q = 1$, the recursive instance has at most $1.5m/2^{2^{i-1}}$ edges. This is because for $i = 1$, $G[B_G^{\pm}(v, r_1^{\pm}) \cap U]$ has at most $|E[B_G^{\pm}(v, r_1^{\pm}) \cap U]| \le |E[B_G^{\pm}(v, D/8) \cap U]| \le 3m/4$ edges, due to $r_1^{\pm} \le a_0 = D/8$ and the assumption that there is no out-heavy or in-heavy vertex, respectively. Meanwhile, for $i > 1$, since Algorithm 3 executes iterations in an alternating way, $G[B_G^{\pm}(v, r_i^{\pm}) \cap U]$ has at most $\mathrm{vol}_G(B_G^{\pm}(v, r_i^{\pm}) \cap U)/2 \le \mathrm{vol}_G(B_G^{\pm}(v, a_{i-1}) \cap U)/2 \le m/2^{2^{i-1}}$ edges after iteration $(i-1)(\mp)$, due to $r_i^{\pm} \le a_{i-1}$ and Lemma 5.2. Since all computation in recursion uses independent randomness, the probability is lowered by a multiplicative factor of at least $p(1.5m/2^{2^{i-1}})$.

If $q = 0$, Algorithm 3 will eventually go into recursion at some later iteration because each vertex in $U$ is sampled with probability 1 in the last iteration. Furthermore, since all computation in different iterations also use independent randomness, it implies

$$\begin{aligned} p(m) &\ge \min_{i\in[L]} \left( \prod_{j=1}^{i} \exp\left( -\frac{c_0 d}{D} \cdot 2^j \log\log(mD) \right) \right) \cdot p\left( \frac{1.5m}{2^{2^{i-1}}} \right) - \frac{1}{(mD)^2} \\ &= \min_{i\in[L]} \exp\left( -\frac{c_0 d}{D} \cdot \sum_{j=1}^{i} 2^j \log\log(mD) \right) \cdot p\left( \frac{1.5m}{2^{2^{i-1}}} \right) - \frac{1}{(mD)^2} \\ &\ge \min_{i\in[L]} \exp\left( -\frac{c_0 d}{D} \cdot 2^{i+1} \log\log(mD) \right) \cdot p\left( \frac{1.5m}{2^{2^{i-1}}} \right) - \frac{1}{(mD)^2} \end{aligned}$$

By setting $c_1 = 4/3$, $c_2 = \max(10c_0, 64)$, $c_3 = 2$, and $\alpha = 2^{2^{i-1}}/1.5$, we get

$$p(m) \ge \min_{\alpha\in[c_1,m]} \exp\left( -\frac{c_2 d}{D} \cdot \log\alpha \log\log(mD) \right) \cdot p\left( \frac{m}{\alpha} \right) - \frac{1}{(mD)^{c_3}},$$

as claimed, because the minimization is relaxed to $\alpha \in [c_1, m]$.

If there are both in-heavy and out-heavy vertices, we verify that the inequality also holds. In case (a), any vertex is marked with probability at most $2 \cdot 2d/(D/4 - D/8) = 32d/D$ before possibly going into recursion. In other words, it is not marked with probability at least $1 - 32d/D \ge \exp(-64d/D)$ for sufficiently small $\epsilon$. The recursive instance has at most $m/2$ edges as a heavy ball is excluded. So we have

$$p(m) \ge \exp\left( -\frac{c_2 d}{D} \cdot \log\alpha \log\log(mD) \right) \cdot p\left( \frac{m}{\alpha} \right) - \frac{1}{(mD)^{c_3}},$$

for $\alpha = 2$, as desired.

In case (b), any vertex is marked with probability at most $2d/(D/8 - D/16) = 32d/D$. Equivalently, it is not marked with probability at least $1 - 32d/D \ge \exp(-64d/D)$ for sufficiently small $\epsilon$. As noted in Remark 2, it either goes into recursion with at most $m/2$ edges, or executes the same algorithm as in cases not having both in-heavy and out-heavy vertices, with a subset $U \subset V$ of remaining vertices initially. A similar argument shows that the same inequality is satisfied with an additional $\exp(-64d/D)$ factor. Thus, increasing $c_2$ by $64/\log c_1$ concludes the proof. □

**Edge cutting probability.** Fix an edge $e = (u, v)$ with $d_e \le \epsilon D / \log\log(mD)$. It turns out the edge cutting probability can be directly derived from the clustering probability.

**Lemma 5.6.** *For edge $e = (u, v) \in E$ with $d_e \le d$, $(u, v)$ is cut only if none of $u, v$ is clustered.*

*Proof.* We prove the lemma by induction. Observe that $(u, v)$ is cut only if $u, v$ go into the same recursion, in which $(u, v)$ is cut, or if Algorithm 3 cuts either a union $B$ of balls or its complement, with exactly one of $u, v$ in $B$. The former case is handled by induction.

For the latter case, first consider a single out-ball $B = B^+_G(t, r)$. By the construction of $\sigma$, to cut $(u, v)$, it must be that $u \in B$ and $v \notin B$, meaning that $\mathrm{dist}_G(t, u) \le r$ and $\mathrm{dist}_G(t, v) > r$. Combined with $d_e \le d$, it further implies that $\mathrm{dist}_G(t, u) \ge \mathrm{dist}_G(t, v) - d_e > r - d$ and $\mathrm{dist}_G(t, v) \le \mathrm{dist}_G(t, u) + d_e \le r + d$. In other words, $\mathrm{dist}_G(t, u), \mathrm{dist}_G(t, v) \in (r - d, r + d]$. Since Algorithm 3 marks all vertices in $B^+_G(t, r + d) \setminus B^+_G(t, r - d)$ whenever either $B$ or its complement is cut, $u, v$ must be marked at the same step, rendering both of them not clustered.

For a union of out-balls $B$, which only occurs in case (b) of having both in-heavy and out-heavy vertices, since Algorithm 3 marks all vertices $v$ with $\mathrm{dist}_B(v) \in (r - d, r + d]$ whenever $B$ is cut, the same argument as above applies with $\mathrm{dist}_B(\cdot)$.

This concludes the proof as in-balls are symmetric. □

**Lemma 5.7.** *For edge $e = (u, v) \in E$, $(u, v)$ is cut by Algorithm 3 with probability at most $1 - \exp(-\mathcal{O}(d_e/D \cdot \log m \log\log(mD)))$.*

*Proof.* As noted in Remark 2, $d$ is used only for clustering, while having nothing to do with cutting. In other words, the event of $(u, v)$ being cut is independent of the value of $d$. Suppose Algorithm 3 is executed with $d = d_e$ instead. The above argument shows the probability of $(u, v)$ being cut is unchanged. By Lemma 5.6, the probability is upper bounded by the probability of $u$ not being clustered, which is at most $1 - p(m)$ by Lemma 5.5. □

**Separation property.** As noted in Remark 2, the intuition for the separation property comes from that vertices within distance $d$ to the cutting boundary are marked and excluded from clustering. Furthermore, we emphasize that it is necessary to mark vertices on both sides of the boundary. To see this, consider two vertices on the same side of the cut with distance at most $d$ between them. It is possible that the distance increases to larger than $d$ on the induced subgraph in the recursion, which happens if some vertex on the shortest path is on the other side of the cut. From the view of the recursion, these two vertices are already separated, while actually they are not in the original graph. This would fail the attempt to prove the separation property by induction. Nevertheless, by marking vertices on both sides of the cutting boundary, at least one of the two vertices must be marked in such case. This is formalized in the following lemma.

**Lemma 5.8.** *For integers $r, d, k \ge 1$ such that $r \ge d$, and vertices $t_1, \ldots, t_k \in V$, let $B = \bigcup_{i=1}^k B^+_G(t_i, r)$, $G' = G[B]$, and $G'' = G[V \setminus B]$. For $u, v \in V$, it holds that:*

1. *If $\mathrm{dist}_B(u) \le r - d$ and $\mathrm{dist}_B(v) > r + d$, then $\mathrm{dist}_G(u, v) \ge 2d$.*
2. *If $\mathrm{dist}_B(u), \mathrm{dist}_B(v) \le r - d$ and $\mathrm{dist}_G(u, v) \le d$, then $\mathrm{dist}_{G'}(u, v) = \mathrm{dist}_G(u, v)$.*
3. *If $\mathrm{dist}_B(u), \mathrm{dist}_B(v) > r + d$ and $\mathrm{dist}_G(u, v) \le d$, then $\mathrm{dist}_{G''}(u, v) = \mathrm{dist}_G(u, v)$.*

*Proof.* For the first item, note that $\mathrm{dist}_G(t_i, v) \le \mathrm{dist}_G(t_i, u) + \mathrm{dist}_G(u, v)$ for any $i \in [k]$ by triangle inequality. Minimizing over $i \in [k]$ on both sides, we get $\mathrm{dist}_B(v) \le \mathrm{dist}_B(u) + \mathrm{dist}_G(u, v)$. Thus, $\mathrm{dist}_G(u, v) \ge \mathrm{dist}_B(v) - \mathrm{dist}_B(u) \ge (r + d) - (r - d) = 2d$.

For the second item, consider the shortest path $P$ from $u$ to $v$ in $G$. If $\mathrm{dist}_G(u,v) \le d$, by a similar argument, any vertex $x$ on $P$ satisfies $\mathrm{dist}_B(x) \le \mathrm{dist}_B(u) + \mathrm{dist}_G(u,x) \le (r-d) + \mathrm{dist}_G(u,v) \le (r-d) + d = r$. In other words, $P$ is entirely contained in $G'$ and thereby $\mathrm{dist}_{G'}(u,v) = \mathrm{dist}_G(u,v)$.

For the third item, also consider the shortest path $P$ from $u$ to $v$ in $G$. If $\mathrm{dist}_G(u,v) \le d$, any vertex $x$ on $P$ satisfies $\mathrm{dist}_B(x) \ge \mathrm{dist}_B(v) - \mathrm{dist}_G(x,v) > (r+d) - \mathrm{dist}_G(u,v) \ge (r+d) - d = r$. So $P$ is entirely contained in $G''$ and thereby $\mathrm{dist}_{G''}(u,v) = \mathrm{dist}_G(u,v)$. □

**Lemma 5.9.** *Let $\sigma = (C_1, \ldots, C_k)$ and $\mathrm{mk}(\cdot)$ be returned by Algorithm 3. For $i, j \in [k]$ such that $i > j$, and vertices $u \in C_i$ and $v \in C_j$ such that $\mathrm{mk}(i) = \mathrm{mk}(j) = 0$, it holds that $\mathrm{dist}_G(u,v) > d$.*

*Proof.* We prove the lemma by induction. In the base case of $m \le 1$, $\sigma$ is induced by a topological order, implying that there is no path from $u$ to $v$. Otherwise, Algorithm 3 constructs $\sigma$ step by step while cutting either unions of balls or their complements. At each step, we only consider the case of out-balls. The case of in-balls is symmetric.

First suppose both of $u, v$ are in a union of out-balls $B$. In order for $u, v$ not to be marked, we must have $\mathrm{dist}_B(u), \mathrm{dist}_B(v) \le r - d$. If $\mathrm{dist}_G(u,v) \le d$, we get $\mathrm{dist}_{G[B]}(u,v) = \mathrm{dist}_G(u,v) \le d$ by Lemma 5.8. In other words, the shortest path from $u$ to $v$ is entirely contained in $G[B]$. Regardless of Algorithm 3 recursing on either $G[B]$ or $G[U \setminus B]$, by induction, at least one of $u, v$ will eventually be marked. The same argument applies to the case of $u, v$ both in $U \setminus B$.

If exactly one of $u, v$ is in $B$, observe that Algorithm 3 constructs $\sigma$ such that each time recursing on an "out-component" (i.e., the subgraph induced by a union of out-balls or complement of a union of in-balls), it concatenates the ordered clustering $\sigma'$ of the out-component to the beginning $\sigma^+$. Meanwhile, each time recursing on an "in-component" (i.e., the subgraph induced by a union of in-balls or complement of a union of out-balls), it concatenates the ordered clustering $\sigma'$ of the in-component to the end $\sigma^-$. Algorithm 3 returns $\sigma^- \circ \sigma^+$ after all recursions.

Altogether, it ensures that any vertex of an out-component occurs after vertices of all in-components and remaining vertices of $U$ when recursing on the component. Also, any vertex of an in-component occurs before vertices of all out-components and remaining vertices of $U$ when recursing on the component. As a result, it must be that $u \in B$ and $v \notin B$ as $B$ is a union of out-balls. In order for $u, v$ not to be marked, we must have $\mathrm{dist}_B(u) \le r - d$ and $\mathrm{dist}_B(v) > r + d$. By Lemma 5.8, we get $\mathrm{dist}_G(u,v) \ge 2d > d$ as claimed. □

**Diameter property.**

**Lemma 5.10.** *Let $\sigma = (C_1, \ldots, C_k)$ be returned by Algorithm 3. For $i \in [k]$, $C_i$ has diameter at most $D$.*

*Proof.* We prove by induction with two base cases. One is when $m \le 1$, implying that each $G[C_i]$ is a single vertex. The other is case (a) of having both in-heavy and out-heavy vertices. The same argument as in [Li25]'s Lemma 10 shows that $B_G^-(s,r) \cap B_G^+(t,r)$ has diameter at most $D$. □

We remark that $C_i$ may not be strongly connected as required by Theorem 5.1. Nevertheless, the diameter property can be satisfied by clustering $C_i$ into strongly connected components and ordering them topologically. No other property is affected.

**Running time.** To bound the running time, we have to implement vertex marking as efficiently as ball cutting. This can be achieved by adapting Dijkstra's algorithm as will be seen in Lemma 5.11. We emphasize that [Li25]'s analysis for the straightforward implementation of the algorithm bounds the running time in expectation because

- The number $k$ of sampled vertices in each iteration is bounded in expectation.
- For fixed $k$, the number of times each vertex is enqueued in Dijkstra's algorithm is bounded in expectation as well.

[Li25] makes the running time bound also hold with high probability by restarting. However, the same cannot be trivially applied to Algorithm 3. Indeed, the event of restarting is correlated with $k$ and the random order, while the restarting probability can be as large as a constant. So it may overwhelm the exponentially small probability guaranteed by Lemma 5.4. Nevertheless, we show that both of the above bounds actually happen with high probability.

**Lemma 5.11.** *Algorithm 3 can be implemented in $\mathcal{O}((m + n \log\log n) \log^2 n)$ time with high probability on graphs with polynomially bounded, integral edge lengths.*

*Proof.* We first bound the running time in expectation. Compared to [Li25]'s algorithm, the only additional running time of Algorithm 3 comes from marking vertices. We focus on cases not having both in-heavy and out-heavy vertices, which are the most time consuming. It turns out that it can be implemented in the same way as [Li25]'s approach to computing balls, which is based on Dijkstra's algorithm.

Concretely, we execute Dijkstra's algorithm for each vertex $v \in S^{\pm}$ in the sampled random order, to compute $B_G^{\pm}(v, r^{\pm}) \cap U$ while marking vertices in $(B_G^{\pm}(v, r^{\pm} + d) \setminus B_G^{\pm}(v, r^{\pm} - d)) \cap U$. Throughout, each vertex $u$ maintains the following additional information.

- The shortest distance at $u$ for any $v' \in S^{\pm}$ before $v$ in the random order.
- An indicator of whether $u \in B_G^{\pm}(v', r^{\pm})$ for some $v' \in S^{\pm}$ before $v$ in the random order.
- An indicator of whether $u$ has been marked.

All above information can be updated in constant time. Note that $u$ is marked by $v$ only if $u \notin B_G^{\pm}(v', r^{\pm})$ for all $v' \in S^{\pm}$ before $v$ in the random order. Meanwhile, $u$ is enqueued in Dijkstra's algorithm only if the shortest distance at $u$ is shortened by $v$. This is because otherwise, for any vertex $u' \in B_G^{\pm}(v, r^{\pm} + d)$ reached via $u$, either $u \in B_G^{\pm}(v', r^{\pm})$ for some $v' \in S^{\pm}$, or $u$ has been marked. Thus, the same analysis as in [Li25]'s Lemmas 8 and 13 bounds the running time in expectation.

We now show that the running time actually happens with high probability. In the following, $m$ always denotes the number of edges in the original graph (as opposed to the induced subgraph for the current recursive instance), and high probability means $1 - 1/\operatorname{poly}(mD)$. For each recursive instance and each iteration $i(\pm)$, by Chernoff bound, the number $k$ of sampled vertices is at most $\mathcal{O}(2^{2^i} \log(mD))$ with high probability. By union bound, with high probability, it holds for all recursive instances and all iterations.

Furthermore, consider fixed vertex $u$ and the random order for sampled vertices. With probability $1/2$, the first vertex in the random order is among the $k/2$ closest to $u$, so at least $k/2$ sampled vertices will not cause $u$ to be enqueued in Dijkstra's algorithm. Similarly, for any sampled vertex that has not been eliminated by previous sampled vertices, it eliminates at least half of remaining sampled vertices with probability $1/2$. All sampled vertices are eliminated once $\log k = \mathcal{O}(2^i + \log\log(mD))$ such vertices are encountered. Summing over all iterations, a total of $\mathcal{O}(\log(mD))$ such vertices are encountered. Since this property holds with probability $1/2$ each time $u$ is enqueued, again by Chernoff bound, $u$ is enqueued a total of $\mathcal{O}(\log(mD))$ times with high probability. Another union bound over all recursive instances and all vertices concludes the proof as $D = \operatorname{poly}(m)$. □

## References


[Abb+22] Amir Abboud, Karl Bringmann, Seri Khoury, and Or Zamir. "Hardness of approximation in p via short cycle removal: cycle detection, distance oracles, and beyond". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2022, pp. 1487–1500.

[ABN08] Ittai Abraham, Yair Bartal, and Ofer Neiman. "Nearly Tight Low Stretch Spanning Trees". In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2008, pp. 781–790. DOI: 10.1109/FOCS.2008.62.

[AN19] Ittai Abraham and Ofer Neiman. "Using Petal-Decompositions to Build a Low Stretch Spanning Tree". In: *SIAM J. Comput.* 48.2 (2019), pp. 227–248. DOI: 10.1137/17M1115575.

[AGM12] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. "Graph sketches: sparsification, spanners, and subgraphs". In: *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. 2012, pp. 5–14. DOI: 10.1145/2213556.2213560.

[Alo+95] Noga Alon, Richard M. Karp, David Peleg, and Douglas B. West. "A Graph-Theoretic Game and Its Application to the k-Server Problem". In: *SIAM J. Comput.* 24.1 (1995), pp. 78–100. DOI: 10.1137/S0097539792224474.

[Awe85] Baruch Awerbuch. "Complexity of Network Synchronization". In: *J. ACM* 32.4 (1985), pp. 804–823. DOI: 10.1145/4221.4227.

[Awe+92a] Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. "Fast Network Decomposition (Extended Abstract)". In: *Proceedings of the International Symposium on Principles of Distributed Computing (PODC)*. 1992, pp. 169–177. DOI: 10.1145/135419.135456.

[Awe+98] Baruch Awerbuch, Bonnie Berger, Lenore Cowen, and David Peleg. "Near-Linear Time Construction of Sparse Neighborhood Covers". In: *SIAM J. Comput.* 28.1 (1998), pp. 263–277. DOI: 10.1137/S0097539794271898.

[Awe+89] Baruch Awerbuch, Andrew V. Goldberg, Michael Luby, and Serge A. Plotkin. "Network Decomposition and Locality in Distributed Computation". In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 1989, pp. 364–369. DOI: 10.1109/SFCS.1989.63504.

[Awe+92b] Baruch Awerbuch, Boaz Patt-Shamir, David Peleg, and Michael E. Saks. "Adapting to Asynchronous Dynamic Networks (Extended Abstract)". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 1992, pp. 557–570. DOI: 10.1145/129712.129767.

[AP90] Baruch Awerbuch and David Peleg. "Sparse Partitions (Extended Abstract)". In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 1990, pp. 503–513. DOI: 10.1109/FSCS.1990.89571.

[Bar98] Yair Bartal. "On Approximating Arbitrary Metrices by Tree Metrics". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 1998, pp. 161–168. DOI: 10.1145/276698.276725.

[Bar96] Yair Bartal. "Probabilistic Approximations of Metric Spaces and Its Algorithmic Applications". In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 1996, pp. 184–193. DOI: 10.1109/SFCS.1996.548477.

[BGS21] 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)*. 2021, pp. 1000–1008. DOI: 10.1109/FOCS52979.2021.00100.

[BNW22] Aaron Bernstein, Danupon Nanongkai, and Christian Wulff-Nilsen. "Negative-Weight Single-Source Shortest Paths in Near-linear Time". In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2022, pp. 600–611. DOI: 10.1109/FOCS54457.2022.00063.

[Ble+14] Guy E. Blelloch, Anupam Gupta, Ioannis Koutis, Gary L. Miller, Richard Peng, and Kanat Tangwongsan. "Nearly-Linear Work Parallel SDD Solvers, Low-Diameter Decomposition, and Low-Stretch Subgraphs". In: *Theory Comput. Syst.* 55.3 (2014), pp. 521–554. DOI: 10.1007/S00224-013-9444-5.

[BCF23] Karl Bringmann, Alejandro Cassis, and Nick Fischer. "Negative-Weight Single-Source Shortest Paths in Near-Linear Time: Now Faster!" In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2023, pp. 515–538. DOI: 10.1109/FOCS57990.2023.00038. URL: https://doi.org/10.1109/FOCS57990.2023.00038.

[Bri+25] Karl Bringmann, Nick Fischer, Bernhard Haeupler, and Rustam Latypov. "Near-Optimal Directed Low-Diameter Decompositions". In: *CoRR* abs/2502.05687 (2025). DOI: 10.48550/ARXIV.2502.05687. arXiv: 2502.05687.

[BLT14] Costas Busch, Ryan LaFortune, and Srikanta Tirthapura. "Sparse Covers for Planar Graphs and Graphs that Exclude a Fixed Minor". In: *Algorithmica* 69.3 (2014), pp. 658–684. DOI: 10.1007/S00453-013-9757-4.

[Che+20] Shiri Chechik, Yang P. Liu, Omer Rotem, and Aaron Sidford. "Constant girth approximation for directed graphs in subquadratic time". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2020, pp. 1010–1023. DOI: 10.1145/3357713.3384330.

[Coh98] Edith Cohen. "Fast Algorithms for Constructing t-Spanners and Paths with Stretch t". In: *SIAM J. Comput.* 28.1 (1998), pp. 210–236. DOI: 10.1137/S0097539794261295.

[Coh00] Edith Cohen. "Polylog-time and near-linear work approximation scheme for undirected shortest paths". In: *J. ACM* 47.1 (2000), pp. 132–166. DOI: 10.1145/331605.331610.

[Coh97] Edith Cohen. "Size-Estimation Framework with Applications to Transitive Closure and Reachability". In: *J. Comput. Syst. Sci.* 55.3 (1997), pp. 441–453. DOI: 10.1006/JCSS.1997.1534.

[CKR00] Gruia Cualinescu, Howard J. Karloff, and Yuval Rabani. "An Improved Approximation Algorithm for MULTIWAY CUT". In: *J. Comput. Syst. Sci.* 60.3 (2000), pp. 564–574.

[CD21] Artur Czumaj and Peter Davies. "Exploiting spontaneous transmissions for broadcasting and leader election in radio networks". In: *Journal of the ACM (JACM)* 68.2 (2021), pp. 1–22.

[Elk+08] Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. "Lower-Stretch Spanning Trees". In: *SIAM J. Comput.* 38.2 (2008), pp. 608–628. DOI: 10.1137/050641661.

[EN19] Michael Elkin and Ofer Neiman. "Efficient Algorithms for Constructing Very Sparse Spanners and Emulators". In: *ACM Trans. Algorithms* 15.1 (2019), 4:1–4:29. DOI: 10.1145/3274651.

[Fil21] Arnold Filtser. "Hop-Constrained Metric Embeddings and their Applications". In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2021, pp. 492–503. DOI: 10.1109/FOCS52979.2021.00056.

[FGN24] Arnold Filtser, Yuval Gitlitz, and Ofer Neiman. "Light, Reliable Spanners". In: *Proceedings of the International Symposium on Computational Geometry (SoCG)*. Vol. 293. LIPIcs. 2024, 56:1–56:15. DOI: 10.4230/LIPICS.SOCG.2024.56.

[FKM23] Arnold Filtser, Michael Kapralov, and Mikhail Makarov. "Expander Decomposition in Dynamic Streams". In: *Proceedings of the Conference on Innovations in Theoretical Computer Science (ITCS)*. Vol. 251. LIPIcs. 2023, 50:1–50:13. DOI: 10.4230/LIPICS.ITCS.2023.50.

[Fis+25] Nick Fischer, Bernhard Haeupler, Rustam Latypov, Antti Roeyskoe, and Aurelio L Sulser. "A Simple Parallel Algorithm with Near-Linear Work for Negative-Weight Single-Source Shortest Path". In: *Proceedings of the SIAM Symposium on Simplicity in Algorithms (SOSA)*. 2025, pp. 216–225.

[Hae+24] Bernhard Haeupler, D. Ellis Hershkowitz, Jason Li, Antti Roeyskoe, and Thatchaphol Saranurak. "Low-Step Multi-commodity Flow Emulators". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. Ed. by Bojan Mohar, Igor Shinkar, and Ryan O'Donnell. 2024, pp. 71–82. DOI: 10.1145/3618260.3649689.

[HHG25] Bernhard Haeupler, Jonas Huebotter, and Mohsen Ghaffari. "A cut-matching game for constant-hop expanders". In: *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2025, pp. 1651–1678.

[HLS24] Bernhard Haeupler, Yaowei Long, and Thatchaphol Saranurak. "Dynamic Deterministic Constant-Approximate Distance Oracles with $n^{\epsilon}$ Worst-Case Update Time". In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2024, pp. 2033–2044. DOI: 10.1109/FOCS61266.2024.00121.

[Hae+25] Bernhard Haeupler, Yaowei Long, Thatchaphol Saranurak, and Shengzhe Wang. "Length-Constrained Directed Expander Decomposition and Length-Constrained Vertex-Capacitated Flow Shortcuts". In: *CoRR* abs/2503.23217 (2025). DOI: 10.48550/ARXIV.2503.23217. arXiv: 2503.23217.

[HW16] Bernhard Haeupler and David Wajc. "A faster distributed radio broadcast primitive". In: *Proceedings of the International Symposium on Principles of Distributed Computing (PODC)*. 2016, pp. 361–370.

[HMO23] Sariel Har-Peled, Manor Mendel, and Dániel Oláh. "Reliable Spanners for Metric Spaces". In: *ACM Trans. Algorithms* 19.1 (2023), 7:1–7:27. DOI: 10.1145/3563356.

[KS97] Philip N. Klein and Sairam Subramanian. "A Randomized Parallel Algorithm for Single-Source Shortest Paths". In: *J. Algorithms* 25.2 (1997), pp. 205–220. DOI: 10.1006/JAGM.1997.0888.

[KMP11] Ioannis Koutis, Gary L. Miller, and Richard Peng. "A Nearly-$m$ log $n$ Time Solver for SDD Linear Systems". In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2011, pp. 590–598. DOI: 10.1109/FOCS.2011.85.

[Kra+04] Robert Krauthgamer, James R. Lee, Manor Mendel, and Assaf Naor. "Measured Descent: A New Embedding Method for Finite Metrics". In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2004, pp. 434–443. DOI: 10.1109/FOCS.2004.41.

[Li25] Jason Li. "Simpler and Faster Directed Low-Diameter Decompositions". In: *CoRR* abs/2505.10244 (2025). DOI: 10.48550/ARXIV.2505.10244. arXiv: 2505.10244.

[LS93] Nathan Linial and Michael E. Saks. "Low diameter graph decompositions". In: *Comb.* 13.4 (1993), pp. 441–454. DOI: 10.1007/BF01303516. URL: https://doi.org/10.1007/BF01303516.

[Mil+15] Gary L Miller, Richard Peng, Adrian Vladu, and Shen Chen Xu. "Improved parallel algorithms for spanners and hopsets". In: *Proceedings of the Symposium on Parallel Algorithms and Architectures (SPAA)*. 2015, pp. 192–201.

[MPX13] Gary L. Miller, Richard Peng, and Shen Chen Xu. "Parallel graph decompositions using random shifts". In: *Proceedings of the Symposium on Parallel Algorithms and Architectures (SPAA)*. 2013, pp. 196–203. DOI: 10.1145/2486159.2486180.

[Pel00] David Peleg. "Proximity-preserving labeling schemes". In: *J. Graph Theory* 33.3 (2000), pp. 167–176. DOI: 10.1002/(SICI)1097-0118(200003)33:3<167::AID-JGT7>3.0.CO;2-5.

[PU89] David Peleg and Eli Upfal. "A trade-off between space and efficiency for routing tables". In: *J. ACM* 36.3 (1989), pp. 510–530. DOI: 10.1145/65950.65953.

[Roz+22] Václav Rozhon, Michael Elkin, Christoph Grunau, and Bernhard Haeupler. "Deterministic Low-Diameter Decompositions for Weighted Graphs and Distributed and Parallel Applications". In: *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*. 2022, pp. 1114–1121. DOI: 10.1109/FOCS54457.2022.00107.

[RG20] Václav Rozhon and Mohsen Ghaffari. "Polylogarithmic-time deterministic network decomposition and distributed derandomization". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2020, pp. 350–363. DOI: 10.1145/3357713.3384298.

[Sey95] Paul D. Seymour. "Packing Directed Circuits Fractionally". In: *Comb.* 15.2 (1995), pp. 281–288. DOI: 10.1007/BF01200760.

[TZ01] Mikkel Thorup and Uri Zwick. "Approximate distance oracles". In: *Proceedings of the Symposium on Theory of Computing (STOC)*. 2001, pp. 183–192. DOI: 10.1145/380752.380798.

[Zuz+22] Goran Zuzic, Gramoz Goranci, Mingquan Ye, Bernhard Haeupler, and Xiaorui Sun. "Universally-Optimal Distributed Shortest Paths and Transshipment via Graph-Based $\ell_1$-Oblivious Routing". In: *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2022, pp. 2549–2579. DOI: 10.1137/1.9781611977073.100.

# A Missing Proofs from Section 5

**Lemma A.1.** *For reals $k_1, k_2, k_3 > 0$, it holds that*

$$\frac{k_2}{k_1+k_2} + \frac{k_3}{k_1+k_2+k_3} \leq 2 - 2\sqrt{\frac{k_1}{k_1+k_2+k_3}}.$$

*Equality is achieved if and only if $k_1 + k_2 = \sqrt{k_1(k_1+k_2+k_3)}$.*

*Proof.* By AM-GM inequality, we have

$$\begin{aligned}\frac{k_2}{k_1+k_2}+\frac{k_3}{k_1+k_2+k_3} &= 1-\frac{k_1}{k_1+k_2}+1-\frac{k_1+k_2}{k_1+k_2+k_3}\\ &\le 2-2\sqrt{\frac{k_1}{k_1+k_2}\cdot\frac{k_1+k_2}{k_1+k_2+k_3}}\\ &= 2-2\sqrt{\frac{k_1}{k_1+k_2+k_3}}.\end{aligned}$$

□

**Lemma A.2.** *For integer $r>1$, and reals $k,k_1,\ldots,k_r\ge 0$ such that $k_1\ge 1$ and $k_1+\ldots+k_r=k$, it holds that*

$$\frac{1}{r}\cdot\sum_{i=1}^{r}\frac{k_i}{\sum_{j=1}^{i}k_i}\le 1-\left(1-\frac{1}{r}\right)\cdot k^{-\frac{1}{r-1}}.$$

*Proof.* We claim that for fixed $r$ and $k$, the maximum possible value of LHS is achieved when $k_{\le i}=n^{(i-1)/(r-1)}$. To see this, consider any possible $k_1,\ldots,k_r$. Suppose there exists $i\in(1,r)$ such that $k_{\le i}\ne\sqrt{k_{\le(i-1)}k_{\le(i+1)}}$. By Lemma A.1, setting $k_{\le i}=\sqrt{k_{\le(i-1)}k_{\le(i+1)}}$, which is same as appropriately setting $k_i$ conditioned on fixed $k_i+k_{i+1}$, strictly increases the value of LHS. Consequently, the value of LHS is maximized only if $k_{\le i}=\sqrt{k_{\le(i-1)}k_{\le(i+1)}}$ for all $i\in(1,r)$. In other words, $k_{\le i}=k_1\cdot(k/k_1)^{\frac{i-1}{r-1}}$ for $i\in[r]$, implying that

$$\text{LHS}=\frac{1}{r}\cdot\left(1+\sum_{i=2}^{r}\frac{k_i}{k_{\le i}}\right)=1-\frac{1}{r}\cdot\sum_{i=2}^{r}\frac{k_{\le(i-1)}}{k_{\le i}}=1-\left(1-\frac{1}{r}\right)\cdot\left(\frac{k}{k_1}\right)^{-\frac{1}{r-1}}.$$

This concludes the proof by setting $k_1=1$. □

*Proof of Lemma 5.3.* Observe that

$$\text{LHS}=\frac{1}{r}\cdot\sum_{\hat{i}=1}^{d}\sum_{i=0}^{\lfloor(r-\hat{i})/d\rfloor}\frac{\sum_{j=\max(\hat{i}+(i-1)\cdot d+1,1)}^{\hat{i}+i\cdot d}k_j}{\sum_{j=1}^{\hat{i}+i\cdot d}k_j}.$$

For fixed $\hat{i}\in[d]$, each term of the inner summation corresponds to a unique interval in $[1,\hat{i}],[\hat{i}+1,\hat{i}+d],\ldots,[\hat{i}+(\hat{r}-1)\cdot d,\hat{i}+\hat{r}\cdot d]$, where $\hat{r}=\lfloor(r-\hat{i})/d\rfloor$. Let $\hat{k}=k_{\le(\hat{i}+\hat{r}\cdot d)}$. Applying Lemma A.2, we get that the inner summand for $\hat{i}$ is at most

$$(\hat{r}+1)\cdot\left(1-\left(1-\frac{1}{\hat{r}+1}\right)\cdot\hat{k}^{-\frac{1}{\hat{r}}}\right)\le(\hat{r}+1)\cdot\left(1-\left(1-\frac{1}{\lfloor r/d\rfloor}\right)\cdot k^{-\frac{1}{\lfloor r/d\rfloor-1}}\right),$$

because $\hat{r}+1\ge\lfloor r/d\rfloor$ and $\hat{k}\le k$. Combining all above concludes the proof as the summation of $\hat{r}+1$ over $i\in[d]$ is just $r$. □