Improved Bounds with a Simple Algorithm for Edge Estimation for Graphs of Unknown Size

Debarshi Chanda
Indian Statistical Institute
Kolkata, India

Abstract

We propose a randomized algorithm with query access that given a graph G with arboricity α, and average degree d, makes $\widetilde{O}\left(\alpha/\varepsilon^2 d\right)$[1] `Degree` and $\widetilde{O}\left(1/\varepsilon^2\right)$ `Random Edge` queries to obtain an estimate $\widehat{d}$ satisfying $\widehat{d} \in (1 \pm \varepsilon)d$. This improves the $\widetilde{O}_{\varepsilon,\log n}\left(\sqrt{n/d}\right)$[2] query algorithm of [Beretta et al., SODA 2026] that has access to `Degree`, `Neighbour`, and `Random Edge` queries. Our algorithm does not require any graph parameter as input, not even the size of the vertex set, and attains both simplicity and practicality through a new estimation technique. We complement our upper bounds with a lower bound that shows for all valid n, d, and α, any algorithm that has access to `Degree`, `Neighbour`, and `Random Edge` queries, must make at least $\Omega\left(\min\left(d, \alpha/d\right)\right)$ queries to obtain a $(1\pm\varepsilon)$-multiplicative estimate of d, even with the knowledge of n and α. We also show that even with `Pair` and `FullNbr` queries, an algorithm must make $\Omega\left(\min\left(d, \alpha/d\right)\right)$ queries to obtain a $(1 \pm \varepsilon)$-multiplicative estimate of d. Our work addresses both the questions raised by the work of [Beretta et al., SODA 2026].

[1] By $\widetilde{O}(\cdot)$, we hide only $\mathrm{poly}\left(\log \alpha\right)$ terms. In particular, we do not hide $\mathrm{poly}\left(\log n\right)$ factors.
[2] The $\widetilde{O}_{\varepsilon,\log n}(\cdot)$ notation hides $\mathrm{poly}\left(\varepsilon^{-1}\log n\right)$ terms.

1 Introduction

The problem of estimating the average degree of a graph is a fundamental problem in sublinear-time graph algorithms [20, 24, 22]. Consider $G = (V, E)$ to be a simple, unweighted, undirected graph with n vertices, and m edges. Given a vertex $v \in V$, we write its set of neighbours as $\mathsf{N}(v)$, and its degree as $d_v = |\mathsf{N}(v)|$. The average degree of the graph is $d := {}^{1}/_{n} \sum_{v \in V} d_v = {}^{2m}/_{n}$. We also denote α to be the arboricity (Definition 6) of the graph. The goal is to design a randomized algorithm with only query access to G to estimate $\widehat{d}$ such that $\widehat{d} \in (1 \pm \varepsilon)\, d$, for some $\varepsilon \in (0, 1)$. We say $\widehat{d}$ is a $(1 \pm \varepsilon)$-multiplicative approximation of d if it satisfies this condition.

For sublinear-time graph algorithms, the graph can be accessed by various queries to the graph. We begin by describing the queries that will be relevant to our work:

- $\texttt{Degree}(v)$: Given a vertex v, this query returns d_v. We also assume this query can be made on a $v \in V$ uniformly at random. In that case, the query is called with no argument, i.e. as just `Degree`.
- $\texttt{Neighbour}(v, i)$: Given a vertex $v \in V$ and $i \in [n]$, this query returns the i-th vertex $u \in \mathsf{N}(v)$ if it exists; otherwise, we get $\perp$.
- $\texttt{Pair}(u, v)$: Given two vertices $u, v \in V$, this query returns 1 if $(u, v) \in E$, and 0, otherwise.
- `RandEdge`: This query returns an edge $e \in E$ uniformly at random.
- $\texttt{FullNbr}(v)$: Given a vertex $v \in V$, this query returns the entire $\mathsf{N}(v)$

The first three queries, `Degree`, `Neighbour`, and `Pair`, have been extensively used in the sublinear-time graph algorithms [15, 22, 3, 18]. The `RandEdge` query has recently found a lot of interest [2, 3, 6, 35, 8, 7, 19]. The `FullNbr` query has also been studied in the context of average degree estimation [35, 7].

The first work of estimating the average degree in sublinear-time model is due to Feige [20], who used $\widetilde{O}_{\varepsilon,\log n}\left(\sqrt{n/d}\right)$ `Degree` queries to obtain a $(2 \pm \varepsilon)$-multiplicative approximation. This work also showed that any algorithm that uses only `Degree` queries are required to make $\Omega(n)$ queries to obtain a $(1 \pm \varepsilon)$-multiplicative approximation to the average degree. Goldreich and Ron [24] later showed that additional access to `Neighbour` queries gives a $\widetilde{O}_{\varepsilon,\log n}\left(\sqrt{n/d}\right)$-query algorithm to obtain a $(1 \pm \varepsilon)$-multiplicative estimate of the average degree. The bound was also shown to be almost (up to $\mathrm{poly}\left(\varepsilon^{-1} \log n\right)$ factors) optimal. On the other hand, motivated by the weighted sampling framework, Motwani et al. [30], Beretta and Tětek [6] established algorithm that uses $\widetilde{O}_{\varepsilon,\log n}\left(n^{1/3}\right)$ `Degree` and `RandEdge` queries.

All the algorithms discussed above assumes the knowledge of the number of vertices n. Designing a size-oblivious algorithm, i.e. an algorithm that does not require to know the size of the input vertex set, n [23]. In this setting, Beretta and Tětek [6] first established an algorithm that estimates the average degree using $O\left({}^{\sqrt{n}}/_{\varepsilon} + {}^{\log n}/_{\varepsilon^2}\right)$ `Degree` and `RandEdge` queries. They also established an exactly matching lower bound for the general sum-estimation problem for the unknown n case. Recently, Beretta et al. [7] established the first graph-specific algorithm to estimate average degree when n is unknown. They showed that even without any knowledge of n, when

RandEdge is allowed in addition to the Degree, and Neighbour queries, average degree can be estimated using $\widetilde{O}_{\varepsilon,\log n}\left(\sqrt{n/d}\right)$ queries. They also showed that when n is known, allowing Pair queries reduces the number of queries to $\widetilde{O}_{\varepsilon,\log n}\left(\sqrt[3]{n/d}\right)$, and allowing FullNbr further reduces it to $\widetilde{O}_{\varepsilon,\log n}\left(\sqrt[4]{n/d}\right)$.

On the other hand, the graph parameter α has been used to parametrize the complexity of various problems in the sublinear-time settings [16, 18, 8, 23]. These studies are motivated by the fact that many real world graphs have low arboricity [21, 12, 18]. In the context of estimating average degree, Eden et al. [17] obtained an algorithm that uses $\widetilde{O}_{\varepsilon,\log n}\left(\alpha/d\right)$ Degree and Neighbour queries, and established an almost matching lower bound. This parametrization on α is interesting as $\alpha/d \leq \sqrt{n/d}$, and can be much smaller for low-arboricity graphs. However, this algorithm needed both the knowledge of n, and an upper bound that is at most a constant factor of α. They also established that any algorithm that makes only Degree and Neighbour queries must make $\Omega\left(\sqrt{n}\right)$ queries without an advice on the arboricity α. Recently, Eden et al. [19] showed if RandEdge queries are allowed in addition, the same upper bound can be attained without the advice on α. Their algorithm, however, still requires the knowledge of n. This gives rise to the following two natural questions:

Does there exist an algorithm that, without any prior knowledge of the graph's parameters, can estimate the average degree of a graph with query complexity depending only on its arboricity α and average degree d?

and

What additional power do structural queries (e.g. FullNbr) provide in this setting?

We answer the first question in the affirmative. In fact, perhaps surprisingly, we show that an algorithm that makes only Degree and RandEdge queries achieves this. For the second part, our lower bounds show that even with all the queries discussed above, no better bounds can be achieved for all ranges of α.

The remainder of the paper is organized as follows. Section 2 presents our main results and discusses their implications in the context of existing literature. Section 3 provides an overview of the key ideas and techniques underlying our work. Section 4 introduces the necessary preliminaries. Section 5 describes our main algorithmic contribution. Finally, Section 6 establishes the corresponding lower bounds.

2 Overview of Our Results

In this section, we give a broad overview of our results. Section 2.1 states the main results of the paper. Section 2.2 discusses some interesting aspects of our results with respect to the literature.

2.1 The Bounds

In this section, we state and place our main results in context. Throughout this paper, we assume that $m = \Omega\left(n\right)$. Such assumptions are standard in the literature [24, 17, 6, 7], and made to facilitate a simpler exposition. When this assumption is violated, given access to RandEdge queries, Ron and Tsur [33] provides a collision-based estimator that uses $o\left(\sqrt{n}\right)$ queries. All our upper bound

results are for the size-oblivious case. First, we state the main algorithmic contribution of our work.

Theorem 1 (Upper Bound - General Graphs)**.** *There exists a randomized algorithm that, given access to* `Degree` *and* `RandEdge` *queries to a graph G with arboricity α, and average degree $d = \Omega(1)$, outputs an estimate $\widehat{d}$ such that $\widehat{d} \in (1 \pm \varepsilon)d$ with probability at least $\frac{9}{10}$, and makes $\widetilde{O}\left(\frac{\alpha}{\varepsilon^2 d}\right)$* `Degree`*, and $\widetilde{O}\left(\frac{1}{\varepsilon^2}\right)$* `RandEdge` *queries in expectation.*

Note that unlike many other algorithms for edge estimation [24, 17, 7], we do not need to assume any lower bound on the degree of individual vertices for this result to hold. However, given a guarantee that a graph does not have any isolated vertices, we can improve this bound using a subroutine of the algorithms presented in earlier works [6, 7].

Theorem 2 (Upper Bound - No Isolated Vertices)**.** *There exists a randomized algorithm that, given access to* `Degree` *and* `RandEdge` *queries to a graph G with arboricity α, average degree d, and no isolated vertices outputs an estimate $\widehat{d}$ such that $\widehat{d} \in (1 \pm \varepsilon)d$ with probability at least $\frac{9}{10}$, and makes $\widetilde{O}\left(\min\left(\frac{d}{\varepsilon^2}, \frac{\alpha}{\varepsilon^2 d}\right)\right)$ queries in expectation.*

Next, we state the lower bound results of this work. It considers a hierarchy of query accesses, starting with access to `Degree`, `Neighbour`, and `RandEdge`, and subsequently adding `Pair`, and `FullNbr` queries, and establishes ranges of arboricity α where our algorithm is optimal. Note that this result does not preclude the algorithm from knowing any graph parameters.

Theorem 3 (Lower Bound)**.** *Consider n, α, and d such that $\alpha/4 \geq d \geq 4$, and there exists a graph with n vertices, average degree d, and arboricity α. Then, there exists a graph with n vertices, average degree d, arboricity α, and no isolated vertices such that any algorithm that can obtain a $(1 \pm \varepsilon)$-multiplicative approximation of the average degree d of the graph for any $\varepsilon < \frac{1}{3}$:*

1. *Using* `Degree`*,* `Neighbour`*, and* `RandEdge` *queries, must make at least $\Omega\left(\min\left(d, \frac{\alpha}{d}\right)\right)$ queries.*
2. *Using* `Degree`*,* `Neighbour`*,* `Pair` *and* `RandEdge` *queries, must make at least $\Omega\left(\min\left(d, \frac{\alpha}{d}\right)\right)$ queries, given $\alpha \leq \sqrt{n}$.*
3. *Using* `Degree`*,* `Neighbour`*,* `Pair`*,* `FullNbr` *and* `RandEdge` *queries, must make at least $\Omega\left(\min\left(d, \frac{\alpha}{d}\right)\right)$ queries, given $\alpha \leq n^{2/5}$.*

2.2 Discussion on the Results

In this section, we discuss the implications of our results, beginning with an overview of the main consequences of our bounds.

Regarding Upper Bounds: Our algorithm, despite having no prior knowledge of n, consistently matches or outperforms the best-known algorithms across a range of query accesses. In particular, it performs at least as well as algorithms that has access to `Degree`, `Neighbour`, and `Pair` queries [20, 24, 14], those employing `Degree` and `RandEdge` queries [6, 30], and those relying on Independent Set queries [10]. Moreover, as summarized in Table 1, our algorithm remains competitive with methods combining `Degree`, `RandEdge`, `Neighbour` and `Pair` queries when

$\alpha \leq \sqrt{n}$, and with approaches additionally leveraging structural queries such as `FullNbr` when $\alpha \leq n^{2/5}$.

Regarding Lower Bounds: Our lower bounds characterize the complexity of edge estimation across different query models in terms of the arboricity α. In particular, we provide tight bounds for algorithms using `Degree`, `Neighbour`, and `RandEdge` queries for all values of α; for those additionally using `Pair` queries when $\alpha \leq \sqrt{n}$; and for those further using `FullNbr` queries when $\alpha \leq n^{2/5}$.

Queries Allowed	$\alpha \leq n^{2/5}$	$n^{2/5} \leq \alpha \leq \sqrt{n}$	$\sqrt{n} \leq \alpha$
`Degree` + `RandEdge`	$\widetilde{O}\left(\min\left(\frac{d}{\varepsilon^2}, \frac{\alpha}{\varepsilon^2 d}\right)\right)$ [Theorem 2]		
+`Neighbour`	$\Omega\left(\min\left(d, \frac{\alpha}{d}\right)\right)$ [Theorem 3]		
+`Pair`	$\Omega\left(\min\left(d, \frac{\alpha}{d}\right)\right)$ [Theorem 3]		$\widetilde{O}_{\varepsilon,\log n}\left(\min\left(d, \sqrt[3]{\frac{n}{d}}\right)\right)$ [7] $\Omega\left(\min\left(d, \sqrt[3]{\frac{n}{d}}\right)\right)$ [7]
+`FullNbr`	$\Omega\left(\min\left(d, \frac{\alpha}{d}\right)\right)$ [Theorem 3]	$\widetilde{O}_{\varepsilon,\log n}\left(\min\left(d, \sqrt[4]{\frac{n}{d}}\right)\right)$ [7] $\Omega\left(\min\left(d, \sqrt[4]{\frac{n}{d}}\right)\right)$ [7]	

Table 1: Summary of the complexity of estimating the average degree across various ranges of arboricity for the queries considered in this work. The cells highlighted in Blue contain results presented in this paper. The other bounds are due to Beretta et al. [7]. Note that both their improved upper bounds for `Pair` and `FullNbr` query accesses presented here requires the knowledge of n, while our algorithm does not.

Now we state some interesting aspects of our result.

- **Comparison with Weighted Sampling Query Access:** Our results show that the improvement previously achieved using additional `Neighbour` queries can, in fact, be obtained solely by leveraging the underlying graph structure—without requiring any extra query access. This advances the line of work on weighted sampling access models [30, 6], where Beretta and Tětek [6] established a $\Omega(\sqrt{n})$ lower bound for the more general problem of sum estimation, and Beretta et al. [7] later overcame this bound by augmenting the model with `Neighbour` queries. In contrast, our bounds demonstrate that comparable improvements arise naturally from exploiting structural properties of the graph itself.

- **Parameter-oblivious algorithms:** Obtaining size-oblivious query based algorithms is an interesting problem across query models [23]. Our algorithm is an effort in that direction as it operates without prior knowledge of the number of vertices n or the arboricity α. Unlike previous algorithms designed for unknown n [30, 6, 7], it does not rely on searching for an appropriate guess, thereby avoiding additional $\operatorname{poly}(\log n)$ factors in its query complexity. Moreover, although α offers a valid upper bound during the search phase, our algorithm does not explicitly depend on it and can terminate well before reaching the value of α.

- **Practical relevance:** `RandEdge` queries have been widely studied in the data mining community in the context of edge sampling [32, 1, 26]. These queries are easy to implement in

practice since many real-world graphs are stored as edge lists, allowing random access to edges directly [27, 34]. Since the work of Aliakbarpour et al. [2], `RandEdge` queries have also been studied extensively for sublinear-time graph algorithms [3, 14, 35, 8, 9, 7, 19], often yielding simple and efficient procedures. In this context, our algorithm stands out for its simplicity and ease of implementation.

- **Regarding the questions of Beretta et al. [7]:** Beretta et al. [7] raised two explicit questions for future work. (Question 1.7) asks whether the $\text{poly}\left(\varepsilon^{-1}\log n\right)$ terms in their bound can be improved, and in particular avoid the binning technique involved in their work. Our algorithm does not employ any binning technique, and does not have any $\text{poly}\left(\log n\right)$ factors in the query complexity. This, partially resolves the first question of Beretta et al. [7]. Secondly, (Question 1.8) asked whether their bounds can be parametrized by α. Our upper bound provides an algorithm that uses only `Degree` and `RandEdge` to achieve this, avoiding the additional access to `Neighbour` in their algorithm. On the other hand, our lower bounds establish that any algorithm that works for all ranges of α, even with access to all the queries they considered, must make $\Omega\left(\alpha/d\right)$ queries. This resolves the second question of Beretta et al. [7].

3 Overview of Techniques and Key Ideas

In this section, we give a brief overview of the technical challenges and ideas of this work. We set up notations in Section 3.1, describe the ideas behind the upper bounds in section 3.2, describe the ideas behind the lower bounds in Section 3.3, and provide some additional observations in Section 3.4.

3.1 Setup and Notation

We begin by setting up some notations that will be used throughout the rest of the paper. We denote d_e to be the degree of the edge $e = (u, v)$ defined as $d_{(u,v)} = \min\left(d_u, d_v\right)$. Throughout the paper, we denote c to be a large enough universal constant. We define a threshold τ and define vertices to be *heavy* or *light* depending on whether their degrees are higher or lower than the given threshold τ.

Definition 4 (Heavy and Light Vertices - (τ))**.** Given a threshold τ, we say a vertex v is heavy if $d_v > \tau$, and light if $d_v \leq \tau$.

In this work, the threshold τ is clear from the context. We denote the set of heavy (resp. light) vertices as H (resp. L). Formally, we have:

$$H = \{v \in V | d_v > \tau\} \qquad \text{and} \qquad L = \{v \in V | d_v \leq \tau\}$$

Note that the sets H, and L is a partition of the set of vertices V. We denote m_H, and m_L to be the sums of degrees of the heavy and light vertices, respectively. Formally, we have:

$$m_H = \sum_{h \in H} d_h \qquad \text{and} \qquad m_L = \sum_{l \in L} d_l$$

We also denote $E(H), E(L)$, and $E(H, L)$ to be the set of edges between $h_i, h_j \in H$, $l_i, l_j \in L$, and $h_i \in H, l_j \in L$, respectively. Based on these notations, we have:

$$m_H = 2\,|E(H)| + |E(H, L)| \qquad \text{and} \qquad m_L = 2\,|E(L)| + |E(H, L)|$$

We denote ρ_L to be the fraction of edges incident to light vertices, i.e., $\rho_L = {}^{m_L}/_{2m}$. Thus, one can think of ρ_L as the density of the light edges. We now define a *good* threshold property, that we will leverage in our algorithm.

Definition 5 (Good Threshold)**.** We define a threshold τ to be good if $m_L \geq \frac{m}{2}$, and consequently $\rho_L \geq \frac{1}{4}$.

3.2 Upper Bound

The problem of estimating the average degree is essentially a problem of mean estimation from samples. The natural strategy to establish an approximation guarantee on a mean estimation problem is to bound the variance of the estimator. This task of bounding the variance can be achieved by setting a maximum value that any such sample can take. In this case, that amounts to setting a threshold τ, and only considering those vertices v with degree values d_v less than or equal to τ. Given such a fixed upper bound, the variance to expectation ratio can be bounded as τ for a sample. Then, a Chebyshev's inequality can be used to bound the deviation of the estimator given sufficient number of samples.

With this idea in mind, our first approach is to partition the set of vertices V into heavy (H) and light (L) vertices. Instead of estimating the average degree exactly, let us first try to estimate it with a slight modification. For any vertex $h \in H$, we set its degree to be 0, and the degree of the vertices in L remains the same. This modification can be implemented with the `Degree` query with a simple post-processing. The sample mean in this case is an unbiased estimator of the quantity ${}^{m_L}/_{n}$, let us denote this quantity to be w_L. Let us denote this estimate to be $\widehat{w}_L$. Due to the upper bound of τ on the samples considered, we can obtain a $(1 \pm \varepsilon)$-estimate using $O\left({}^{\tau}/_{(\varepsilon^2\,\mathbb{E}[\widehat{w}_L])}\right)$ queries by Chebyshev's inequality.

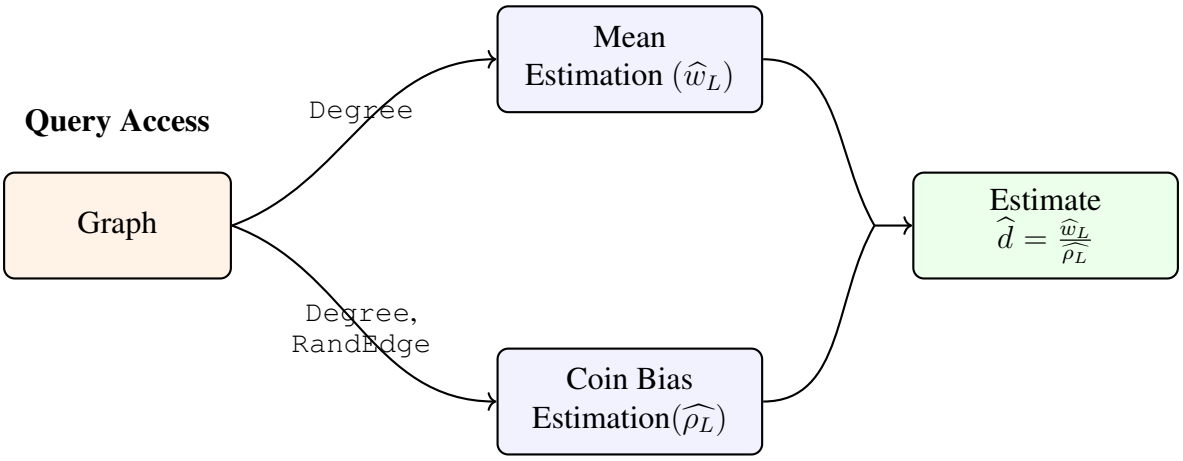

Figure 1: An overall picture of the main ideas for the algorithm.

However, in this work we want to obtain a $(1 \pm \varepsilon)$-multiplicative approximation of the average degree. Our first observation is that the average degree can be expressed as $d = {}^{w_L}/_{\rho_L}$. Hence, if we could approximate ρ_L up to $(1 \pm \varepsilon)$-multiplicative error, we can combine the two estimates $\widehat{w}_L$ and $\widehat{\rho_L}$ to obtain a good estimation of the average degree. Observe that, the definition of the good threshold ensures that $\rho_L = {}^{m_L}/_{m} \geq {}^{1}/_{4}$. Hence, if we can simulate a coin toss with bias ρ_L, we can obtain a $(1 \pm \varepsilon)$-multiplicative approximation using $\widetilde{O}\left({}^{1}/_{\varepsilon^2}\right)$ tosses. We simulate this coin toss using `RandEdge` queries in conjunction with `Degree` queries. Consider taking a random edge, and chose one of its end points uniformly at random. The probability that this end point is light is exactly ρ_L. Hence, return 1 if the end point is light, and 0, otherwise.

The next problem is to find a good threshold that satisfies this criteria. Observe that for the threshold to be good, we need the number of edges in the induced subgraph of H to be small. By a corollary of the Nash-Williams Theorem (Corollary 8), we know that any threshold $\tau \geq c\alpha$ is a good threshold. But this bound only enables us to have an algorithm that takes as input a value of τ that satisfies this condition. However, note that for our purposes, any threshold τ that satisfies the good threshold property is sufficient. We again leverage the coin-toss framework to search for a good threshold. We estimate ρ_L up to an additive ${}^{1}/_{16}$-approximation using $\widetilde{O}\left({}^{1}/_{\varepsilon^2}\right)$ queries, and reject if $\widehat{\rho_L} \leq {}^{5}/_{16}$. This gives us a good threshold.

Now we need to combine these ideas for our algorithm. The idea is to search for a good threshold. If we obtain a good threshold, we can call our algorithm with an advice on the threshold to estimate the average degree. Our checks for a good threshold ensure that any threshold τ much higher than the arboricity α of the graph occurs with decreasing probability. Taking an expectation over the possible values yields our α dependent bound.

3.3 Lower Bound

Our lower bound uses an indistinguishability argument on two families of graphs. The two cases we consider consists of either k or $2k$ cliques, and the rest of the vertices form a matching. The construction is similar to that considered by Beretta et al. [7]. We divide the proof into two parts, the low degree case (when $d \leq \sqrt{\alpha}$), and the high degree case (when $d \geq \sqrt{\alpha}$). Fixing the k appropriately with respect to n, d, and α establishes our bounds.

3.4 Further Observations

One of the important aspects of our bound is that it does not have any $\mathrm{poly}\left(\log n\right)$ term. Since the work of Goldreich and Ron [24], a standard strategy to bound the variance of the estimator has been to use a binning technique to divide the vertex set V of the graph into $\mathrm{poly}\left(\log n\right)$ parts [4, 7, 10] and combine the estimators. However, this technique requires $\mathrm{poly}\left(\log n\right)$ factors to handle the required union bound over all the parts. This issue was raised by Beretta et al. [7] as this did not seem inherent to the problem. Our technique of directly controlling the variance through controlling the maximum degree resolves this issue. In particular, the only $\mathrm{poly}\left(\log \alpha\right)$ factors incurred in our algorithm is due to the search procedures.

One of the starting point of arboricity dependent graph algorithms is the work of Chiba and Nishizeki [11]. One of the key results of their work is that the sum of the degrees of the edges can be upper bounded as $\sum_{e \in E} d_e \leq m\alpha$. Many of the works in property testing [17, 18, 8, 19], as well as in other models [5, 28] leverage this property. In the context of edge estimation, the variance

of the estimator of Eden et al. [17] is bounded in terms of α using this result. Our result works by characterizing the necessary conditions for our estimator to work, and enforcing that condition in terms of the Nash-Williams Theorem [31].

4 Preliminaries

In this section, we set up the preliminaries that will be relevant to our work. We start by giving a formal definition of the arboricity of a graph.

Definition 6 (Arboricity(α))**.** The arboricity of a graph $G = (V, E)$, denoted by α, is the minimum number of spanning forests that the set of edges E can be partitioned into.

We next state the Nash-Williams Theorem, and two simple corollaries that will be relevant to our work.

Lemma 7 (Nash-Williams Theorem [31])**.** *Given a graph $G = (V, E)$, and for a subgraph H of G, denote n_H, and m_H to be its number of vertices and edges, respectively. Then, the arboricity α of the graph G satisfies:*

$$\alpha = \max_{H \subseteq G} \left\lceil \frac{m_H}{n_H - 1} \right\rceil$$

The next corollary provides an upper bound on the number of edges in a graph on n vertices, and arboricity α. The proof follows simply as G is a valid subgraph of itself.

Corollary 8 (Arboricity based bound on m)**.** *Given a graph on n vertices, and arboricity α, the number of edges, m of the graph is bounded as $m \leq n\alpha$.*

The next corollary provides upper bounds on the subgraphs, and directly follows from the theorem.

Corollary 9. *Given a graph G with arboricity α_G, the arboricity α_H of any subgraph $H \subseteq G$ of G satisfies $\alpha_H \leq \alpha_G$.*

The next result establishes the existence of a good threshold depending only on the arboricity α of the graph.

Lemma 10. *Given a graph with arboricity α, a threshold $\tau \geq p\alpha$ ensures $m_L \geq m\left(1 - \frac{2}{p}\right)$. In particular, any threshold $\tau \geq 8\alpha$ satisfies $\rho_L \geq \frac{3}{8}$.*

Proof. Given there are m edges, the number of heavy vertices with respect to the threshold $\tau \geq p\alpha$ is at most $\frac{2m}{p\alpha}$. Notice that $|E_H|$ is the size of the induced subgraph of the vertices H. Let α_H denote the arboricity of this induced subgraph.

By Corollary 9, we have $\alpha_H \leq \alpha$. Then, by Corollary 8, we have $|E(H)| \leq \frac{2m\alpha_H}{p\alpha} \leq \frac{2m}{p}$. Then, we have:

$$m_L = 2\,|E(L)| + |E(H, L)| \geq |E(L)| + |E(H, L)| = m - |E(H)| \geq m\left(1 - \frac{2}{p}\right)$$

□

Some standard concentration inequalities are mentioned in Section A of the Appendix.

5 The Algorithm

In this section, we present the algorithmic contribution of our work. We start by setting up the building blocks in Section 5.1. Next, we describe the algorithms that work with advice in Section 5.2, the first one with advices on both a good threshold, and the average degree, and the second one with advice only on a good threshold. Finally, in Section 5.3, we establish an algorithm that works without any advice.

5.1 The Building Blocks

In this section, we set up the building blocks of our algorithm: `CoinToss` that estimates the density of light edges ρ_L, and `MeanEst` that estimates the quantity m_L/n.

5.1.1 `CoinToss`

We start with the algorithm `CoinToss` that takes as input a possible threshold τ, and the number of queries r it makes, and access to `Degree` and `RandEdge` queries, and outputs $\widehat{\rho_L}$ as an estimate of the quantity ρ_L.

Algorithm 1 `CoinToss`(τ, r)

Require: Access to `Degree` and `RandEdge` queries, a threshold τ, and number of queries r

1: **for** $i \in [r]$ **do**
2: $(u, v) \leftarrow$ `RandEdge` ▷ Makes r `RandEdge` queries
3: $w \leftarrow \text{Unif}\{u, v\}$
4: $X_i \leftarrow 1$ if $w \in L$ (i.e. `Degree`$(w) \leq \tau$), and 0 otherwise(i.e. `Degree`$(w) > \tau$) ▷ Makes r `Degree` queries
5: **return** $\widehat{\rho_L} \leftarrow \frac{1}{r}\sum_{i\in[r]} X_i$

The next lemma provides the performance guarantees of the Algorithm 1. The performance of the algorithm depends on the number of queries made and the quantity ρ_L.

Lemma 11. *For the algorithm `CoinToss`, the following holds:*

1. $\mathbb{E}\left[\widehat{\rho_L}\right] = \rho_L$
2. *If* $r = \Omega\left(\log\left(1/\delta\right)\right)$ *and* $\rho_L < \frac{1}{4}$*, i.e., the threshold* τ *is not good,* $\widehat{\rho_L} \leq \frac{5}{16}$ *with probability at least* $1 - \delta$*.*
3. *If* $r = \Omega\left(\log\left(1/\delta\right)\right)$ *and* $\rho_L \geq \frac{3}{8}$*, the estimate satisfies* $\widehat{\rho_L} \geq \frac{5}{16}$ *with probability at least* $1 - \delta$*.*
4. *If the threshold* τ *is good, and* $r = \Omega\left(\frac{\log(1/\delta)}{\varepsilon^2}\right)$*,* $\widehat{\rho_L} \in (1 \pm \varepsilon)\rho_L$ *with probability at least* $1 - \delta$*.*

Proof. First, we note that the X_i-s are defined in `CoinToss` are i.i.d. Bernoulli random variables with $p = \rho_L$.

$$\Pr[X_i = 1] = \sum_{l \in L} \sum_{i \in [d_l]} \frac{1}{2m} = \sum_{l \in L} \frac{d_l}{2m} = \frac{m_L}{2m} = \rho_L$$

The result of part (1) follows directly by taking expectation. We obtain the parts (2) and (3) of the claim through an additive Chernoff bound with additive error $1/16$. For the part (4), as the threshold is good, we have $\rho_L \geq 1/4$. Hence, by multiplicative Chernoff bound, the claim holds. □

5.1.2 `MeanEst`:

Next, we describe the algorithm `MeanEst` that is given a threshold τ, the number of queries q it makes, and access to `Degree` query, and outputs $\widehat{w}_L$ as an estimate of the quantity m_L/n.

Algorithm 2 `MeanEst`(τ, q)

Require: Access to `Degree` query, a good threshold τ, and number of queries q

1: **for** $i \in [q]$ **do**
2: $\quad d_i \leftarrow$ `Degree` ▷ Makes q `Degree` queries
3: $\quad w_i \leftarrow d_i$ if $d_i \leq \tau$, and 0 otherwise
4: **return** $\widehat{w}_L \leftarrow \frac{1}{q} \sum_{i \in q} w_i$

Now, we want to show that $\widehat{w}_L$ gives a good approximation to m_L/n if the number of `Degree` queries q is sufficiently high. First, we state the expectation of $\widehat{w}_L$.

Lemma 12. *In the algorithm* `MeanEst`*, we have* $\mathbb{E}[\widehat{w}_L] = \frac{m_L}{n}$.

Proof. Note that only vertices in L contribute to the estimate. Hence,

$$\mathbb{E}[\widehat{w}_L] = \mathbb{E}\left[\frac{1}{q} \sum_{i \in [q]} w_i\right] = \frac{1}{q} \sum_{i \in [q]} \mathbb{E}[w_i] = \mathbb{E}[w_i] = \sum_{l \in L} \frac{1}{n} d_l = \frac{m_L}{n}$$

□

Next, we bound the variance of the estimate $\widehat{w}_L$.

Lemma 13. *In the algorithm* `MeanEst`*, we have* $\mathrm{Var}[\widehat{w}_L] \leq \frac{\tau}{q} \mathbb{E}[\widehat{w}_L]$.

Proof. First observe that w_i and w_j are independent $\forall i, j \in [q], i \neq j$. Hence,

$$\mathrm{Var}[\widehat{w}_L] = \mathrm{Var}\left[\frac{1}{q} \sum_{i \in [q]} w_i\right] = \frac{1}{q^2} \mathrm{Var}\left[\sum_{i \in [q]} w_i\right] = \frac{1}{q^2} \sum_{i \in [q]} \mathrm{Var}[w_i]$$

Here, the last equality is due to the fact that $\mathrm{Cov}(w_i, w_j) = 0$ for all $i, j \in [q]$. Now, we focus on the individual w_is,

$$\mathrm{Var}[w_i] \leq \mathbb{E}[w_i^2] \leq \tau \, \mathbb{E}[w_i] = \tau \, \mathbb{E}[\widehat{w}_L]$$

Here, the second inequality is due to the fact that $\tau \geq w_i$ for all $i \in [q]$. Combining the results above, we have:

$$\mathrm{Var}\left[\frac{1}{q}\sum_{i\in[q]} w_i\right] \leq \frac{\tau}{q}\,\mathbb{E}[\widehat{w}_L]$$

□

The next lemma shows that if the number of queries q made by `MeanEst` is sufficiently high, then $\widehat{w}_L$ is a $(1 \pm \varepsilon)$-multiplicative estimate to ${}^{m_L}/_{n}$.

Lemma 14. *If* $q = \Omega\left(\frac{\tau}{\delta\varepsilon^2 d}\right)$, *the algorithm* `MeanEst` *satisfies* $\Pr\left[\widehat{w}_L \notin (1\pm\varepsilon)\,\mathbb{E}[\widehat{w}_L]\right] \leq \delta$.

Proof. By Chebyshev's inequality, we have:

$$\Pr\left[\widehat{w}_L \notin (1\pm\varepsilon)\,\mathbb{E}[\widehat{w}_L]\right] \leq \frac{\mathrm{Var}[\widehat{w}_L]}{\varepsilon^2\,\mathbb{E}^2[\widehat{w}_L]} \leq \frac{\tau}{\varepsilon^2 q\,\mathbb{E}[\widehat{w}_L]} \leq \frac{2\tau}{\varepsilon^2 q d} \leq \delta.$$

Here, the third inequality follows from the fact that the given threshold τ is good. □

5.2 Estimating with Advice

In this section, we present two algorithms, `AllAdvice` and `ThresholdAdvice`. `AllAdvice` is given as advice a good threshold τ, a bound on the average degree $\widetilde{d}$, and ε, δ to obtain an estimate $\widehat{d}$ that is a $(1 \pm \varepsilon)$-multiplicative approximation to the quantity d. `ThresholdAdvice` then leverages `AllAdvice` to obtain an algorithm that does not require the advice $\widetilde{d}$.

5.2.1 `AllAdvice`

We begin with the algorithm `AllAdvice` that combines the algorithms `CoinToss` and `MeanEst` to get an algorithm to obtain an $(1 \pm \varepsilon)$-multiplicative estimate of the average degree with both advices τ, and $\widetilde{d}$.

Algorithm 3 `AllAdvice`$\left(\tau, \widetilde{d}, \varepsilon, \delta\right)$

Require: A good threshold τ, $\widetilde{d}$, $\varepsilon \in (0,1)$, and $\delta \in \left(0, \frac{1}{2}\right)$

1: $\widehat{w}_L \leftarrow$ `MeanEst`$\left(\tau, q = \frac{\tau}{\delta\varepsilon^2\widetilde{d}}\right)$ ▷ Makes $\frac{\tau}{\delta\varepsilon^2\widetilde{d}}$ `Degree` queries.

2: $\widehat{\rho_L} \leftarrow$ `CoinToss`$\left(\tau, r = \frac{c\log(1/\delta)}{\varepsilon^2}\right)$ ▷ Makes $\frac{c\log(1/\delta)}{\varepsilon^2}$ `Degree` and `RandEdge` queries.

3: **return** $\widehat{d} \leftarrow \frac{\widehat{w}_L}{\widehat{\rho_L}}$

The following lemma provides the performance guarantee of the algorithm. The result combines the performance guarantees of `MeanEst` (Lemma 11) and `CoinToss` (Lemma 14).

Lemma 15. `AllAdvice` *makes* $q = O\left(\max\left(\frac{\log(1/\delta)}{\varepsilon^2}, \frac{\tau}{\delta\varepsilon^2\widetilde{d}}\right)\right)$ `Degree` *queries, and* $r = O\left(\frac{\log(1/\delta)}{\varepsilon^2}\right)$ `RandEdge` *queries, and if* $\widetilde{d} \leq 16d$, *returns* $\widehat{d} \in (1\pm\varepsilon)d$ *with probability at least* $1-\delta$. *Furthermore, for all values of* $\widetilde{d}$, *the estimator satisfies* $\mathbb{E}[\widehat{d}] \leq 8d$ *with probability at least* $1-\delta$.

Proof. The query bounds follow directly from the algorithm, as highlighted in the comments of the pseudocode. Then, by Lemma 11, we have $\widehat{\rho_L} \in (1 \pm \varepsilon/2)\,\rho_L$, and by Lemma 14, we have $\widehat{w}_L \in (1 \pm \varepsilon/2)\,m_L/n$ with probability at least $1 - \delta/2$. Combining through union bound, we have $\widehat{d} = \widehat{w}_L/\widehat{\rho_L} \in (1 \pm \varepsilon)\,m_L/n\rho_L = (1 \pm \varepsilon)d$ with probability at least $1 - \delta$.

For the expectation, observe that $\widehat{d} = \widehat{w}_L/\widehat{\rho_L}$. By Corollary 8 and Lemma 12, for any good threshold, we have $\mathbb{E}[\widehat{w}_L] = m_L/n \leq d$. By Lemma 11, for any good threshold, we have $\widehat{\rho_L} \geq \rho_L/2 \geq 1/8$ with probability at least $1 - \delta$. Combining these two bounds completes the proof. □

5.2.2 `ThresholdAdvice`

Now, we present `ThresholdAdvice`, which does not require the advice $\widetilde{d}$ on the average degree of the graph. The algorithm leverages the `AllAdvice` algorithm to search for an appropriate bound $\widetilde{d}$ on the average degree. This form of parameter search is widely used in the property testing literature, see for example Eden et al. [15, 18].

Algorithm 4 `ThresholdAdvice`$(\tau, \varepsilon, \delta)$

Require: A good threshold τ, $\varepsilon \in (0,1)$, and $\delta \in \left(0, \frac{1}{2}\right)$

1: **for** $i \in [\log \tau]$ **do** ▷ Outer Loop
2: $\quad\widetilde{d}_i \leftarrow \frac{\tau}{2^i}$
3: $\quad$**for** $j \in \left[\left\lceil c \log(\frac{\log \tau}{\delta})\right\rceil\right]$ **do** ▷ Inner Loop
4: $\quad\quad\widehat{d}_j \leftarrow$ `AllAdvice`$\left(\tau, \widetilde{d}_i, \varepsilon, \frac{\delta}{c \log^2 \tau}\right)$ ▷ Makes $\frac{c\tau \log^2 \tau}{\delta\varepsilon^2 \widetilde{d}_i}$ `Degree` and $\frac{c \log(\log \tau/\delta)}{\varepsilon^2}$ `RandEdge` queries.
5: $\quad d_{min} \leftarrow \min_j \widehat{d}_j$
6: $\quad$**if** $d_{min} \geq \widetilde{d}_i$ **then** ▷ Valid $\widetilde{d}$
7: $\quad\quad$**return** $\widehat{d} \leftarrow$ `AllAdvice`$\left(\tau, \widetilde{d}_i, \varepsilon, \frac{\delta}{2}\right)$ ▷ Makes $\frac{c\tau}{\delta\varepsilon^2 \widetilde{d}_i}$ `Degree` and $\frac{c \log(1/\delta)}{\varepsilon^2}$ `RandEdge` queries.

Now, we establish the performance guarantees of the `ThresholdAdvice` algorithm.

Lemma 16. *`ThresholdAdvice` returns $\widehat{d} \in (1 \pm \varepsilon)d$ with probability at least $1 - \delta$, and makes $O\left(\frac{\tau \log^4 \tau}{\delta\varepsilon^2 d}\right)$ `Degree`, and $O\left(\log^2 \tau \frac{\log(\log \tau/\delta)}{\varepsilon^2}\right)$ `RandEdge` queries in expectation.*

Proof. We start with the guarantees of the estimate $\widehat{d}$ produced by the algorithm. First, we consider any value of $\widetilde{d}$ such that $\widetilde{d} \geq 16d$. The bound on $\mathbb{E}[\widehat{d}]$ of Lemma 15 holds throughout all the runs with probability at least $1 - \delta/3$ by union bound. Conditioning on this event, a Markov's inequality argument based on Lemma 15 shows that for any j, we have $\Pr\left[\widehat{d}_j \geq \widetilde{d}\right] \leq 1/2$, for any $\widetilde{d} \geq 8d$. The probability that d_{min}, the minimum over $c \log(\log \tau/\delta)$ such estimates, is smaller than $\widetilde{d}$ is at most $\delta^c/\log^c \tau$. There are at most $\log \tau$ such values that appear in this algorithm due to the outer loop (Line 1). Hence, the probability that the algorithm returns $\widehat{d}$ by satisfying the valid $\widetilde{d}$ criteria at Line 6 for some $\widetilde{d} \geq 16d$ is at most $\delta/3$.

Alternatively, if a valid $\widetilde{d}$, i.e. $\widetilde{d} \leq 16d$ satisfies the condition, then by Lemma 15, it returns $\widehat{d} \in (1 \pm \varepsilon)\,d$ with probability at least $1 - \delta/3$. Combining, we obtain that the algorithm returns $\widehat{d} \in (1 \pm \varepsilon)\,d$ with probability at least $1 - \delta$.

Now, we bound the number of queries made by the algorithm. Note that for all values of $\widetilde{d}$ such that $\widetilde{d} > d/8$, the query complexity for each round is $O\left(\frac{\tau \log^2 \tau}{\delta \varepsilon^2 d}\right)$ `Degree`, and $O\left(\frac{\log(\log \tau/\delta)}{\varepsilon^2}\right)$ `RandEdge` queries. There are at most $\log^2 \tau$ such rounds, hence overall query bound for these values remain $O\left(\frac{\tau \log^4 \tau}{\delta \varepsilon^2 d}\right)$ `Degree`, and $O\left(\log^2 \tau \frac{\log(\log \tau/\delta)}{\varepsilon^2}\right)$ `RandEdge` queries. The bound on `RandEdge` queries are independent of $\widetilde{d}$, and hence remains $O\left(\log^2 \tau \frac{\log(\log \tau/\delta)}{\varepsilon^2}\right)$ if both the loops are executed entirely.

Hence, we only consider the `Degree` queries for the lower values of $\widetilde{d}$ such that $\widetilde{d} \leq d/8$. We argue that the probability of reaching any such value goes down as the value becomes lower. First observe that for any $\widetilde{d} \leq d/8$, the probability that it satisfies the valid $\widetilde{d}$ condition (Line 6) is at least $1 - \delta$ by union bound. Then, the probability to hit $\widetilde{d}_i < d/2^j$ is at most δ^{j-2}, as all the previous valid $\widetilde{d}$ must have failed. Hence, we can bound the expected query complexity over the entire run of the algorithm as:

$$\delta^{j-2} \cdot \frac{c\tau \log^2 \tau}{2^j \delta \varepsilon^2 d} = \frac{c\delta^{j-3} \tau \log^2 \tau}{2^j \varepsilon^2 d} = O\left(\frac{\tau \log^2 \tau}{\varepsilon^2 d}\right)$$

Here, the last equality follows from the fact that $\delta < 1/2$. Thus, the algorithm makes $O\left(\frac{\tau \log^4 \tau}{\delta \varepsilon^2 d}\right)$ `Degree`, and $O\left(\log^2 \tau \frac{\log(\log \tau/\delta)}{\varepsilon^2}\right)$ `RandEdge` queries in expectation. □

5.3 Estimating without Advices - `NoAdvice`

Now we present the algorithm `NoAdvice` that can estimate the average degree without any advice. The idea here is that we do not need to actually know α for our algorithm to succeed. Any τ that satisfies the good threshold property will be sufficient, and we can verify the goodness of a threshold using the `CoinToss` algorithm. On the other hand, Lemmas 10 and 11 show that any value larger than 8α will ensure $\widehat{\rho_L} \geq 5/8$. We combine these two observations to search for a good threshold and bound the number of queries made by the algorithm.

Algorithm 5 `NoAdvice`(ε, δ)

Require: $\varepsilon \in (0, 1)$, and $\delta \in \left(0, \frac{1}{2}\right)$

1: $i \leftarrow 0$
2: **while** `True` **do**
3: $\quad \tau_i \leftarrow 2^i$
4: $\quad \widehat{\rho_{Li}} \leftarrow$ `CoinToss`$\left(\tau_i, c \log \frac{2\tau_i}{\delta}\right)$ ▷ Makes $c \log \frac{2\tau_i}{\delta}$ `RandEdge` queries
5: $\quad$ **if** $\widehat{\rho_{Li}} \geq \frac{5}{16}$ **then** ▷ Getting good threshold
6: $\quad\quad$ **return** $\widehat{d} \leftarrow$ `ThresholdAdvice`$\left(\tau_i, \varepsilon, \frac{\delta}{2}\right)$ ▷ Makes $\frac{c\tau \log^3 \tau}{\delta \varepsilon^2 d}$ `Degree` and $\log^2 \tau \frac{c \log(\log \tau/\delta)}{\varepsilon^2}$ `RandEdge` queries
7: $\quad i \leftarrow i + 1$

The next result provides the performance guarantees of `NoAdvice`. Here, we combine the performance guarantees of `CoinToss` (Lemma 11), and `ThresholdAdvice` (Lemma 16) along with the role of arboricity as a good threshold (Lemma 10) to obtain our result.

Lemma 17 (Estimating Without Advice). *`NoAdvice` makes $\widetilde{O}\left(\frac{\alpha}{\delta\varepsilon^2 d}\right)$ `Degree` queries, and $\widetilde{O}\left(\frac{\log(1/\delta)}{\varepsilon^2}\right)$ `RandEdge` queries, and returns $\widehat{d} \in (1 \pm \varepsilon)d$ with probability at least $1 - \delta$.*

Proof. By Lemma 11, the probability that at the i-th iteration, the check for good threshold (Line 5) succeeds for a threshold that is not good is at most $\delta/2^{i+1}$. Hence, by union bound, with probability at least $1-\delta/2$, `ThresholdAdvice` is called with a good threshold. Consequently, by Lemma 16, we have $\widehat{d} \in (1 \pm \varepsilon)\, d$ with probability at least $1 - \delta$.

Now, we focus on the number of queries made by the algorithm. First, note that the algorithm makes $\widetilde{O}\left(\log(2^{i+1}/\delta)\right)$ `Degree` and `RandEdge` queries for each call of the `CoinToss`. The queries made up to the i-th iteration such that $\tau_i \leq 8\alpha$ can be bounded as $\widetilde{O}\left(\log(1/\delta)\right)$. For the thresholds such that $\tau > 8\alpha$, observe that by Corollary 8 and Lemma 11, the probability that the algorithm reaches $\tau_i > 2^{j+3}\alpha$ is at most $1/2^j$, as each of the earlier good thresholds with $\rho_L \geq \frac{3}{8}$ must have been missed for this to happen, and $\delta \leq 1/2$. Then, we can bound the queries as $\widetilde{O}\left(\log(1/\delta)\right) \sum_j 2^{-j} j = \widetilde{O}\left(\log(1/\delta)\right)$.

The number of queries made by the call to `ThresholdAdvice` depends on the value of i for which the check for good threshold (Line 5) succeeds. If it is called for any threshold $\tau \leq 8\alpha$, it makes at most $\widetilde{O}\left(\frac{\alpha}{\delta\varepsilon^2 d}\right)$ `Degree` queries.

Now we consider the higher values, i.e., $\tau > 8\alpha$. Again, we argue that the probability that the call is made with a high threshold decreases as the value increases. Recall that by Lemma 10, and 11, any threshold $\tau \geq 8\alpha$ passes the check for good threshold (Line 5) with probability at least $1 - \delta$. Correspondingly, the probability that the threshold at which the check at Line 5 succeeds is larger than $2^{j+4}\alpha$ is at most $1/2^j$ as the probability that each of the earlier checks failed is at most $\delta < 1/2$.

Note that the only term in the query complexity of `ThresholdAdvice` is $\tau \log^3 \tau$. We can bound its expectation as $\widetilde{O}\left(\alpha\right)$, the proof is deferred to Section B of the Appendix. Similarly, we can also bound the number of `RandEdge` queries made for the `ThresholdAdvice` call as $\widetilde{O}\left(\frac{\log(1/\delta)}{\varepsilon^2}\right)$. Hence, the expected query complexity for the `ThresholdAdvice` call is $\widetilde{O}\left(\frac{\alpha}{\delta\varepsilon^2 d}\right)$ `Degree` queries, and $\widetilde{O}\left(\frac{\log(1/\delta)}{\varepsilon^2}\right)$ `RandEdge` queries, by Lemma 16. □

5.4 Final Bounds

In this section, we state the main upper bounds of our work. First, we establish an upper bound for the case of general graphs by fixing $\delta = 2/3$.

Theorem 1 (Upper Bound - General Graphs). *There exists a randomized algorithm that, given access to `Degree` and `RandEdge` queries to a graph G with arboricity α, and average degree $d = \Omega\left(1\right)$, outputs an estimate $\widehat{d}$ such that $\widehat{d} \in (1 \pm \varepsilon)d$ with probability at least $\frac{9}{10}$, and makes $\widetilde{O}\left(\frac{\alpha}{\varepsilon^2 d}\right)$ `Degree`, and $\widetilde{O}\left(\frac{1}{\varepsilon^2}\right)$ `RandEdge` queries in expectation.*

Next, we state a result due to Beretta and Tětek [6] that can estimate average degree when at most a constant fraction of vertices are isolated. For exposition purpose, we state the result under a slightly more stricter assumption that no vertex is isolated.

Lemma 18 (Edge Estimation [6, 7]). *There exists an algorithm that, given access to `Degree` and `RandEdge` queries to a graph G with no isolated vertices, and average degree d, and an advice*

$\bar{d} \geq d$ *outputs an estimate* $\widehat{d}$ *such that* $\widehat{d} \in (1 \pm \varepsilon)d$ *of the average degree* d *with probability at least* $\frac{9}{10}$ *if the advice is correct. If* $\bar{d} \leq \frac{d}{2^i}$*, the algorithm rejects the advice with probability at least* $1 - \frac{1}{2^{i-1}}$*. The algorithm makes* $\widetilde{O}\left(\frac{\bar{d}}{\varepsilon^2}\right)$ *queries in expectation.*

We can combine this result with our `NoAdvice` algorithm. We use the algorithm of [6, 7] with $\bar{d} = \sqrt{\tau}$ at each guess of the threshold, and output the average degree if the algorithm returns an estimate $\widehat{d}$, or move to the next threshold if the advice is rejected. The other modification concerns the call to the `ThresholdAdvice` algorithm once a good threshold τ^* has been detected. In this case, we check only up to $\widetilde{d} \geq \sqrt{\tau^*}$, and otherwise invoke the algorithm of [6, 7] with $\bar{d} = \sqrt{\tau^*}$. These modifications yield the following result, established through an argument similar to that of Lemma 17.

Theorem 2 (Upper Bound - No Isolated Vertices)**.** *There exists a randomized algorithm that, given access to* `Degree` *and* `RandEdge` *queries to a graph* G *with arboricity* α*, average degree* d*, and no isolated vertices outputs an estimate* $\widehat{d}$ *such that* $\widehat{d} \in (1 \pm \varepsilon)d$ *with probability at least* $\frac{9}{10}$*, and makes* $\widetilde{O}\left(\min\left(\frac{d}{\varepsilon^2}, \frac{\alpha}{\varepsilon^2 d}\right)\right)$ *queries in expectation.*

6 Lower Bounds

In this section, we establish the lower bound of our work. The proofs are divided in two parts. In Theorem 19, we show that $\Omega\left(\frac{\alpha}{d}\right)$ queries are necessary for the high degree case, i.e. when $d \geq \sqrt{\alpha}$. Then, in Theorem 20, we show that $\Omega(d)$ queries are necessary for the low degree case, i.e. when $d \leq \sqrt{\alpha}$. We start with the higher degree case Theorem 19.

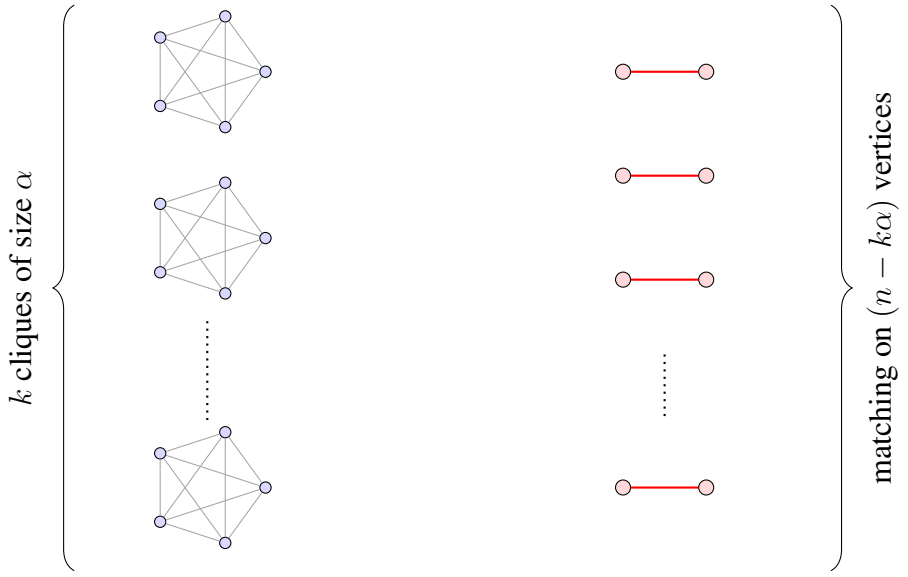

Figure 2: The construction for the Lower Bounds (Theorem 19 and 20)

Theorem 19 (Lower Bound - High Degree Case)**.** *Consider* n*,* α*, and* d *such that* $\alpha/4 \geq d \geq \sqrt{\alpha}$ *and* $d \geq 4$*, and there exists a graph with* n *vertices, average degree* d*, and arboricity* α*. Then,*

there exists a graph with n vertices, average degree d, arboricity α, and no isolated vertices such that any algorithm that can obtain a $(1 \pm \varepsilon)$-multiplicative approximation of the average degree d of the graph for any $\varepsilon < \frac{1}{3}$:

1. *Using* `Degree`*,* `Neighbour`*, and* `RandEdge` *queries must make at least $\Omega\left(\frac{\alpha}{d}\right)$ queries.*
2. *Using* `Degree`*,* `Neighbour`*,* `Pair` *and* `RandEdge` *queries must make at least $\Omega\left(\frac{\alpha}{d}\right)$ queries, given $\alpha \leq \sqrt{n}$.*
3. *Using* `Degree`*,* `Neighbour`*,* `Pair`*,* `FullNbr` *and* `RandEdge` *queries must make at least $\Omega\left(\frac{\alpha}{d}\right)$ queries, given $\alpha \leq n^{2/5}$.*

Proof. We first describe the graph that we use to construct the lower bound. The graphs consist of $k \geq 1$ cliques of size α, and the remaining vertices form a matching. We denote the set of vertices forming the clique (resp. matching) to be V_c (resp. V_m). Similarly, we denote the set of edges forming the clique (resp. matching) to be E_c (resp. E_m). Note that by our construction, we have $|V_c| = k\alpha$, $|V_m| = n - k\alpha$, $|E_c| = \Theta(k\alpha^2)$, and $|E_m| = \Theta(n - k\alpha)$. We denote the set of vertices obtained by making `Degree` queries as Q_d, and the endpoints of the edges obtained by making `RandEdge` queries as Q_e.

For the lower bound instances, we consider two cases, one where $k = \frac{nd}{\alpha^2}$, and the other where $k = \frac{2nd}{\alpha^2}$. Note that this must be larger than 1 by Corollary 8. Hence, $k \geq 1$ is satisfied. Furthermore, note that $k \leq \frac{n}{8}$ as we have $d \leq \frac{\alpha}{4}$, and $\alpha \geq d \geq 4$. Note that any algorithm that obtains a $(1 \pm \varepsilon)$-multiplicative approximation for an $\varepsilon \leq \frac{1}{3}$ to the average degree d should be able to distinguish between these two cases.

Now, we want to show that across these two cases, the samples received are indistinguishable if the algorithm makes $q = o\left(\frac{\alpha}{d}\right)$ queries. We first show that in this case, $Q_d \subseteq V_m$, $Q_e \subseteq V_c$, and all these samples are distinct with probability $1 - o(1)$. Next, we show that all the samples obtained are distinct, i.e. there are no collisions. Hence, the two cases remain indistinguishable.

First, we show that $Q_d \subseteq V_m$ with probability $1 - o(1)$. The probability that a `Degree` query hits a vertex in V_c is $\frac{|V_c|}{n} = \frac{k\alpha}{n} \leq \frac{2d}{\alpha}$, as $k \leq \frac{2nd}{\alpha^2}$. Hence, the probability that any of the q `Degree` queries hits a vertex in V_c is $\frac{2qd}{\alpha} = o(1)$. Furthermore, all the samples in Q_d are distinct with probability $1 - o(1)$ as $\sqrt{V_m} = \sqrt{n - \frac{2nd}{\alpha}} \geq \sqrt{\frac{n}{2}} \geq \frac{\alpha}{2\sqrt{d}} \geq \frac{\alpha}{d}$. Here, the first inequality follows from the fact that $d \leq \frac{\alpha}{4}$, the second inequality follows from the fact that $\alpha \leq \sqrt{nd}$, and the last inequality follows from the fact that $d \geq 4$.

Next, we show that $Q_e \subseteq V_c$ with probability $1 - o(1)$. The probability that a `RandEdge` query hits an edge in E_m is $\frac{|E_m|}{nd} \leq \frac{n - nd/\alpha}{nd} \leq \frac{1}{d}$. Hence, the probability that any of the q `RandEdge` queries hits an edge in E_m is at most $\frac{q}{d} = o(1)$, as $d \geq \sqrt{\alpha}$. Furthermore, all the endpoints in Q_e are distinct with probability $1 - o(1)$ as $\sqrt{|V_c|} = \sqrt{k\alpha} = \sqrt{\frac{nd}{\alpha}} = \frac{\sqrt{nd}}{\sqrt{\alpha}} \geq \sqrt{\alpha} \geq \frac{\alpha}{d}$. Here, the second last inequality follows from Corollary 8, and the last inequality follows from the fact that $d \geq \sqrt{\alpha}$,.

Now that we have shown that the output of the `Degree` and `RandEdge` queries are indistinguishable between the two cases considered, we now prove the three parts of the result by showing that the `Neighbour`, `Pair`, and `FullNbr` queries indistinguishable within the given ranges of α.

Proof of part (1): We start with the `Neighbour` queries. First we consider the `Neighbour` queries on the vertices in V_m. We assume that the algorithm does not make redundant `Neighbour` queries, i.e. does not make `Neighbour` queries on the same vertex twice. In that case, as $\sqrt{|V_m|} \geq \frac{\alpha}{d}$, there is no collision in q queries with probability $1 - o(1)$.

Next, we consider the `Neighbour` queries on the vertices in V_c. Again, we assume that the algorithm does not make redundant `Neighbour` queries, i.e. does not make `Neighbour` queries on the same vertex twice. In that case, even if there is only a single clique, it has α vertices. Note that as $d \geq \sqrt{\alpha}$, , we have $\sqrt{\alpha} \geq \frac{\alpha}{d}$. Hence, there will not be a collision in q queries with probability $1 - o(1)$. Therefore, the algorithm can not distinguish the two cases using $o\left(\frac{\alpha}{d}\right)$ `Degree`, `Neighbour`, and `RandEdge` queries. This completes the proof of part (1) of the claim.
Proof of part (2): Next, we consider the `Pair` queries. We show that if the algorithm makes q `Pair` queries, then all such queries return 0 with probability $1 - o(1)$ for both instances. The algorithm can not gain any information using `Pair` on vertices in V_m beyond that obtained through `Neighbour` queries. Hence, we consider the algorithm does not make these redundant queries, and the `Pair` queries are made only on vertices in V_c. Let us consider the queries to be $e_1, e_2, ..., e_q$ where each $e_i = (u_i, v_i)$ for some $u_i, v_i \in V_c$. The probability that any such query returns 1 is at most $\frac{1}{k}$. Hence, in q queries, the expected number of queries to return 1 is $\frac{2q}{k} \leq o\left(\frac{\alpha^3}{nd^2}\right) \leq o\left(\frac{\alpha^2}{n}\right) = o(1)$. Here, the last inequality follows from the fact that $d \geq \sqrt{\alpha}$, and the last follows from the fact that $\alpha = o(\sqrt{n})$. Then, by Markov's Inequality, the no queries returns 1 with probability $1 - o(1)$. This completes the proof for part (2).
Proof of part (3): Next, we consider the `FullNbr` queries. Again for any vertex in V_m, `FullNbr` is equivalent to `Neighbour`, and we consider the algorithm does not make such redundant queries. For vertices in V_c, each `FullNbr` query reveal all the id-s in the clique containing the vertex. There are $k = \frac{nd}{\alpha^2}$ cliques, and we have $\sqrt{\frac{nd}{\alpha^2}} \geq \sqrt{\alpha^{1/2}d} \geq \sqrt{\alpha^{1/2}d\frac{\alpha^{3/2}}{d^3}} = \frac{\alpha}{d}$. Here, the first inequality follows from the fact that $n \geq \alpha^{5/2}$, and the second inequality follows from the fact that $d \geq \sqrt{\alpha}$. Hence, with probability $1 - o(1)$, the $q = o\left(\frac{\alpha}{d}\right)$ `FullNbr` queries does not hit any clique twice. Hence, the all the cliques obtained by `FullNbr` queries are distinct, and the two cases remain indistinguishable. This completes the proof for part (3). □

Next, we establish our lower bound for the low degree case.

Theorem 20 (Lower Bound - Low Degree Case)**.** *Consider n, α, and d such that $d \leq \sqrt{\alpha} \leq n^{1/3}$ and $d \geq 4$, and there exists a graph with n vertices, average degree d, and arboricity α. Then, there exists a graph with n vertices, average degree d, arboricity α, and no isolated vertices such that any algorithm that can obtain a $(1 \pm \varepsilon)$-multiplicative approximation of the average degree d of the graph for any $\varepsilon < \frac{1}{3}$:*

1. *Using `Degree`, `Neighbour`, and `RandEdge` queries must make at least $\Omega(d)$ queries.*
2. *Using `Degree`, `Neighbour`, `Pair` and `RandEdge` queries must make at least $\Omega(d)$ queries, given $\alpha \leq \sqrt{n}$.*
3. *Using `Degree`, `Neighbour`, `Pair`, `FullNbr` and `RandEdge` queries must make at least $\Omega(d)$ queries, given $\alpha \leq n^{2/5}$.*

Proof. The proof uses the same construction as Theorem 19, we consider the graphs to contain k cliques, and the remaining vertices form a matching. The two instances considered has k set to $\frac{nd}{\alpha^2}$ and $\frac{2nd}{\alpha^2}$. This value is still valid as $1 \leq k \leq \frac{n}{8}$ in this parameter regime as well. In this case, for contradiction, we assume that there exists an algorithm that makes $q = o(d)$ queries. As before, we show that any such algorithm can not distinguish between the two cases.

First, we consider the case of `Degree` queries. The probability that any of them hit a vertex in V_c is at most $\frac{2d}{\alpha}$. The probability that any of the q `Degree` queries hit V_c is at most $\frac{2qd}{\alpha}$, which is $o(1)$, as $d \leq \sqrt{\alpha}$. As before, note that $\sqrt{|V_m|} \geq \frac{\alpha}{d}$. This, given we consider $d \leq \sqrt{\alpha}$, ensures that all the samples are distinct with probability $1 - o(1)$.

Next, we consider `RandEdge` queries, the probability that any such query hits an edge in E_m is, as before, at most $\frac{1}{d}$. Hence, the probability that any of the q `RandEdge` queries hits an edge in E_m is $1 - o(1)$. We now consider whether all the endpoints of Q_e are distinct. Here, we have $\sqrt{|V_c|} = \sqrt{\frac{nd}{\alpha}} \geq \sqrt{\frac{\alpha^{3/2}d}{\alpha}} = \sqrt{\alpha d} \geq d$. Here, the first inequality follows from the fact that $\alpha \leq n^{2/3}$, and the second inequality follows from the fact that $d \leq \sqrt{\alpha}$. Now we focus on the other queries.

Proof of part (1): Now, we consider the `Neighbour` queries. As before, we concern ourselves only with the `Neighbour` queries on the vertices in V_c. Even if we have only a single clique, as $\sqrt{\alpha} \geq d$, there is no collision with probability $1 - o(1)$. Thus, with probability $1 - o(1)$, any algorithm that makes $q = o(d)$ queries can not distinguish between the two cases.

Proof of part (2): Next, we consider the `Pair` queries. Again, the `Pair` queries to vertices in V_m does not yield any additional information compared to `Neighbour` queries. The probability that any `Pair` queries on the vertex pairs of V_c returns 1 is at most $\frac{q}{k} = o\left(\frac{\alpha^2}{n}\right) = o(1)$, where the last equality follows from the fact that $\alpha \leq \sqrt{n}$.

Proof of part (2): We now consider the `FullNbr` queries. As before, we only need to consider the queries made on the vertices in V_c. To ensure that there is no collision with probability $1 - o(1)$, we must have $\sqrt{k} \geq d$. This holds true as $\sqrt{\frac{nd}{\alpha^2}} \geq \sqrt{\alpha^{1/2}d} \geq d$. Here, the first inequality is due to the fact that $\alpha \leq n^{2/5}$, and the second inequality is due to the fact that $d \leq \sqrt{\alpha}$. □

Combining Theorem 19 and 20, we obtain the main lower bound result of our work. The key observation here is the fact that $d \geq \alpha^2/n$ (Corollary 8) ensures that for $\alpha \geq n^{2/3}$, we always have $d \geq \alpha/d$.

Theorem 3 (Lower Bound)**.** *Consider n, α, and d such that $\alpha/4 \geq d \geq 4$, and there exists a graph with n vertices, average degree d, and arboricity α. Then, there exists a graph with n vertices, average degree d, arboricity α, and no isolated vertices such that any algorithm that can obtain a $(1 \pm \varepsilon)$-multiplicative approximation of the average degree d of the graph for any $\varepsilon < \frac{1}{3}$:*

1. *Using `Degree`, `Neighbour`, and `RandEdge` queries, must make at least $\Omega\left(\min\left(d, \frac{\alpha}{d}\right)\right)$ queries.*
2. *Using `Degree`, `Neighbour`, `Pair` and `RandEdge` queries, must make at least $\Omega\left(\min\left(d, \frac{\alpha}{d}\right)\right)$ queries, given $\alpha \leq \sqrt{n}$.*
3. *Using `Degree`, `Neighbour`, `Pair`, `FullNbr` and `RandEdge` queries, must make at least $\Omega\left(\min\left(d, \frac{\alpha}{d}\right)\right)$ queries, given $\alpha \leq n^{2/5}$.*

References

[1] N. K. Ahmed, J. Neville, and R. Kompella. Network sampling: From static to streaming graphs. *ACM Trans. Knowl. Discov. Data*, 8(2), June 2013. ISSN 1556-4681. doi: 10.1145/2601438. URL https://doi.org/10.1145/2601438.

[2] M. Aliakbarpour, A. S. Biswas, T. Gouleakis, J. Peebles, R. Rubinfeld, and A. Yodpinyanee. Sublinear-time algorithms for counting star subgraphs via edge sampling. *Algorithmica*, 80 (2):668–697, Feb. 2018. ISSN 0178-4617. doi: 10.1007/s00453-017-0287-3. URL https://doi.org/10.1007/s00453-017-0287-3.

[3] S. Assadi, M. Kapralov, and S. Khanna. A Simple Sublinear-Time Algorithm for Counting Arbitrary Subgraphs via Edge Sampling. In A. Blum, editor, *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, volume 124 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:20, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-095-8. doi: 10.4230/LIPIcs.ITCS.2019.6. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITCS.2019.6.

[4] P. Beame, S. Har-Peled, S. N. Ramamoorthy, C. Rashtchian, and M. Sinha. Edge estimation with independent set oracles. *ACM Trans. Algorithms*, 16(4), Sept. 2020. ISSN 1549-6325. doi: 10.1145/3404867. URL https://doi.org/10.1145/3404867.

[5] S. K. Bera and C. Seshadhri. How the degeneracy helps for triangle counting in graph streams. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS'20, page 457–467, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371087. doi: 10.1145/3375395.3387665. URL https://doi.org/10.1145/3375395.3387665.

[6] L. Beretta and J. Tětek. Better sum estimation via weighted sampling. *ACM Trans. Algorithms*, 20(3), June 2024. ISSN 1549-6325. doi: 10.1145/3650030. URL https://doi.org/10.1145/3650030.

[7] L. Beretta, D. Chakrabarty, and C. Seshadhri. Faster estimation of the average degree of a graph using random edges and structural queries. In *Proceedings of the Thirty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '26, USA, 2026. Society for Industrial and Applied Mathematics. URL https://arxiv.org/abs/2507.06925.

[8] A. Bishnu, D. Chanda, and G. Mishra. Arboricity and random edge queries matter for triangle counting using sublinear queries, 2025. URL https://arxiv.org/abs/2502.15379.

[9] A. Bishnu, D. Chanda, and G. Mishra. Towards tight bounds for estimating degree distribution in streaming and query models, 2025. URL https://arxiv.org/abs/2507.21784.

[10] X. Chen, A. Levi, and E. Waingarten. Nearly optimal edge estimation with independent set queries. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*,

pages 2916–2935, 2020. doi: 10.1137/1.9781611975994.177. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611975994.177.

[11] N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. Comput.*, 14(1):210–223, 1985. doi: 10.1137/0214017. URL https://doi.org/10.1137/0214017.

[12] M. Danisch, O. Balalau, and M. Sozio. Listing k-cliques in sparse real-world graphs*. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 589–598, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. ISBN 9781450356398. doi: 10.1145/3178876.3186125. URL https://doi.org/10.1145/3178876.3186125.

[13] D. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, USA, 1st edition, 2009. ISBN 0521884276.

[14] T. Eden and W. Rosenbaum. Lower Bounds for Approximating Graph Parameters via Communication Complexity. In E. Blais, K. Jansen, J. D. P. Rolim, and D. Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*, volume 116 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 11:1–11:18, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-085-9. doi: 10.4230/LIPIcs.APPROX-RANDOM.2018.11. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.APPROX-RANDOM.2018.11.

[15] T. Eden, A. Levi, D. Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. *SIAM Journal on Computing*, 46(5):1603–1646, 2017. doi: 10.1137/15M1054389. URL https://doi.org/10.1137/15M1054389.

[16] T. Eden, D. Ron, and W. Rosenbaum. The Arboricity Captures the Complexity of Sampling Edges. In C. Baier, I. Chatzigiannakis, P. Flocchini, and S. Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 52:1–52:14, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-109-2. doi: 10.4230/LIPIcs.ICALP.2019.52. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ICALP.2019.52.

[17] T. Eden, D. Ron, and C. Seshadhri. Sublinear time estimation of degree distribution moments: The arboricity connection. *SIAM Journal on Discrete Mathematics*, 33(4):2267–2285, 2019. doi: 10.1137/17M1159014. URL https://doi.org/10.1137/17M1159014.

[18] T. Eden, D. Ron, and C. Seshadhri. Faster sublinear approximation of the number of k-cliques in low-arboricity graphs. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '20, page 1467–1478, USA, 2020. Society for Industrial and Applied Mathematics.

[19] T. Eden, R. Rubinfeld, and A. Vasilyan. Testable algorithms for approximately counting edges and triangles in sublinear time and space, 2025. URL https://arxiv.org/abs/2509.20351.

[20] U. Feige. On sums of independent random variables with unbounded variance and estimating the average degree in a graph. *SIAM Journal on Computing*, 35(4):964–984, 2006. doi: 10.1137/S0097539704447304. URL https://doi.org/10.1137/S0097539704447304.

[21] G. Goel and J. Gustedt. Bounded arboricity to determine the local structure of sparse graphs. In *Proceedings of the 32nd International Conference on Graph-Theoretic Concepts in Computer Science*, WG'06, page 159–167, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3540483810. doi: 10.1007/11917496_15. URL https://doi.org/10.1007/11917496_15.

[22] O. Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

[23] O. Goldreich. Open problems in property testing of graphs, 2021.

[24] O. Goldreich and D. Ron. Approximating average parameters of graphs. *Random Structures & Algorithms*, 32(4):473–493, 2008.

[25] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley, Reading, MA, 2 edition, 1994.

[26] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 631–636, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933395. doi: 10.1145/1150402.1150479. URL https://doi.org/10.1145/1150402.1150479.

[27] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[28] Q. C. Liu and C. Seshadhri. Brief announcement: Improved massively parallel triangle counting in o(1) rounds. In *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing*, PODC '24, page 519–522, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706684. doi: 10.1145/3662158.3662819. URL https://doi.org/10.1145/3662158.3662819.

[29] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, Cambridge, 2005.

[30] R. Motwani, R. Panigrahy, and Y. Xu. Estimating Sum by Weighted Sampling. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming, ICALP*, pages 53–64, 2007.

[31] C. S. A. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *Journal of the London Mathematical Society*, s1-36(1):445–450, 1961. doi: https://doi.org/10.1112/jlms/s1-36.1.445. URL https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/jlms/s1-36.1.445.

[32] B. Ribeiro and D. Towsley. On the estimation accuracy of degree distributions from graph sampling. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 5240–5247, Maui, Hawai, 2012. IEEE. doi: 10.1109/CDC.2012.6425857.

[33] D. Ron and G. Tsur. The power of an example: Hidden set size approximation using group queries and conditional sampling. *ACM Trans. Comput. Theory*, 8(4), June 2016. ISSN 1942-3454. doi: 10.1145/2930657. URL https://doi.org/10.1145/2930657.

[34] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015. URL http://networkrepository.com.

[35] J. Tětek and M. Thorup. Edge sampling and graph parameter estimation via vertex neighborhood accesses. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, page 1116–1129, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392648. doi: 10.1145/3519935.3520059. URL https://doi.org/10.1145/3519935.3520059.

Appendix

A Concentration Inequalities

Here, we state some well-known concentration inequalities that is relevant to our work. These bounds are widely known, see e.g. Mitzenmacher and Upfal [29], Dubhashi and Panconesi [13] for more details.

Lemma 21 (Markov's Inequality)**.** *For a non-negative random variable X, we have:*

$$\Pr\left[X \geq t\right] \leq \frac{\mathbb{E}[X]}{t}$$

Lemma 22 (Chebyshev's Inequality)**.** *For a random variable X, we have:*

$$\Pr\left[|X - \mathbb{E}[X]| \geq \varepsilon\right] \leq \frac{\mathrm{Var}[X]}{t^2}$$

Lemma 23 (Multiplicative Chernoff Bound)**.** *Given i.i.d. random variables $X_1, X_2, ..., X_t$ where $\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = (1-p)$, define $X = \sum_{i\in[t]} X_i$. Then, we have:*

$$\Pr[|X - \mathbb{E}[X]| \geq \varepsilon\,\mathbb{E}\left[X\right]] \leq 2\exp\left(-\frac{\varepsilon^2\,\mathbb{E}\left[X\right]}{2}\right) \qquad 0 \leq \varepsilon < 1$$

Lemma 24 (Additive Chernoff Bound)**.** *Given i.i.d. random variables $X_1, X_2, ..., X_t$ where $\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = (1-p)$, define $X = \sum_{i\in[t]} X_i$. Then, we have:*

$$\Pr[|X - \mathbb{E}[X]| \geq \varepsilon t] \leq 2\exp\left(-\frac{\varepsilon^2 t}{2}\right) \qquad 0 \leq \varepsilon$$

B Deferred Proof for `NoAdvice`

Lemma 25 (Expected Stopping-Cost bound of `NoAdvice`)**.** *Suppose the stopping threshold τ satisfies $\Pr[\tau = 2^{j+4}\alpha] \leq 2^{-j}$ for all integers $j \geq 0$. Then we have $\mathbb{E}[\tau \log^4 \tau] = O(\alpha \log^4 \alpha)$. In particular, $\mathbb{E}[\tau \log^4 \tau] = \tilde{O}(\alpha)$.*

Proof. First we use the upper bound on the probability to obtain:

$$\mathbb{E}[\tau \log^4 \tau] = \sum_{j\geq 0} \Pr[\tau = 2^{j+4}\alpha] \cdot 2^{j+4}\alpha(\log(2^{j+4}\alpha))^4 \leq c\alpha \sum_{j\geq 0} 2^{-j}\,(\log\alpha + j)^4.$$

We analyse the sum in two parts, for $j \leq \lceil \log_2 \alpha \rceil$ and $j > \lceil \log_2 \alpha \rceil$. We denote $L = \lceil \log_2 \alpha \rceil$. For the lower range, i.e. $0 \leq j \leq L$, we have $\log\alpha + j = O(\log\alpha)$, hence

$$\sum_{j=0}^{L} 2^{-j}(\log\alpha + j)^4 = O(\log^4\alpha) \sum_{j=0}^{L} 2^{-j} = O(\log^4\alpha).$$

For the higher range, i.e. $j > L$, we have $(\log \alpha + j)^4 = O(j^4)$. In this case, we have:

$$\sum_{j=L+1}^{\infty} 2^{-j}(\log \alpha + j)^4 = O\left(\sum_{j=0}^{\infty} 2^{-j} j^4\right) = O(1),$$

where the final sum is a finite constant [25]. Combining the two parts gives

$$\sum_{j \geq 0} 2^{-j}(\log \alpha + j)^4 = O(\log^4 \alpha).$$

Therefore we have $\mathbb{E}[\tau \log^4 \tau] \leq \alpha \cdot O(\log^4 \alpha) = O(\alpha \log^4 \alpha)$, as claimed. □