Almost Optimal Bounds for Sublinear-Time Sampling of k-Cliques in Bounded Arboricity Graphs

Talya Eden
Boston University, MA, USA
MIT, Cambridge, MA, USA

Dana Ron
Tel Aviv University, Israel

Will Rosenbaum
Amherst College, MA, USA

Abstract

Counting and sampling small subgraphs are fundamental algorithmic tasks. Motivated by the need to handle massive datasets efficiently, recent theoretical work has examined the problems in the sublinear time regime. In this work, we consider the problem of sampling a k-clique in a graph from an almost uniform distribution. Specifically the algorithm should output each k-clique with probability $(1 \pm \epsilon)/n_k$, where n_k denotes the number of k-cliques in the graph and ϵ is a given approximation parameter. To this end, the algorithm may perform degree, neighbor, and pair queries. We focus on the class of graphs with arboricity at most α, and prove that the query complexity of the problem is

$$\Theta^*\left(\min\left\{n\alpha, \max\left\{\left(\frac{(n\alpha)^{k/2}}{n_k}\right)^{\frac{1}{k-1}}, \frac{n\alpha^{k-1}}{n_k}\right\}\right\}\right),$$

where n is the number of vertices in the graph, and $\Theta^*(\cdot)$ suppresses dependencies on $(\log n/\epsilon)^{O(k)}$.

Our upper bound is based on defining a special auxiliary graph H_k, such that sampling edges almost uniformly in H_k translates to sampling k-cliques almost uniformly in the original graph G. We then build on a known edge-sampling algorithm (Eden, Ron and Rosenbaum, ICALP19) to sample edges in H_k. The challenge is simulating queries to H_k while being given query access only to G. Our lower bound follows from a construction of a family of graphs with arboricity α such that in each graph there are n_k k-cliques, where one of these cliques is "hidden" and hence hard to sample.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases sublinear time algorithms, graph algorithms, cliques, arboricity, uniform sampling

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.56

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: https://arxiv.org/abs/2012.04090 [24]

Funding *Talya Eden*: This research was partially supported by Ben Gurion Postdoctoral Scholarship.
Dana Ron: This research was partially supported by the Israel Science Foundation (grant No. 1041/18).

1 Introduction

We consider the problem of sampling k-cliques in sublinear-time. Sampling subgraphs is a fundamental computational task in randomized algorithms, statistics, data science, and many other disciplines. Sampling k-cliques, and triangles in particular, has numerous applications across various fields, see, e.g. [38, 7, 19, 47, 16] and references therein. The best exact combinatorial algorithm for this task is an $O((n/\log n)^k)$-time algorithm by Vassilevska [46], and Chen et al. [14] proved that, under the exponential time hypothesis, there is no $n^{o(k)}$-time algorithm.

Motivated by the need to handle massive datasets efficiently, we consider algorithms that are given query access to the graph, in the form of degree, neighbor and pair queries.[1] We refer to this model as the *general query model*. Our goal is to design an algorithm that samples k-cliques while performing as few queries as possible.

Fichtenberger, Gao, and Peng [32] recently studied the problem of sampling *arbitrary* subgraphs. They assumed access to the above queries, as well as access to uniform edge samples. Thus, they considered a strictly stronger model. Specifically, for sampling k-cliques uniformly at random, their algorithm has expected complexity[2] $\tilde{O}(m^{k/2}/n_k)$, where m and n_k denote the number of edges and k-cliques in the graph, respectively. Their result is known to be essentially tight, due to a lower bound by Eden and Rosenbaum [28].

However, the lower bound of [28] only holds when considering the worst-case over all possible inputs. In this work we ask whether the lower bound can be circumvented when considering graphs with bounded arboricity. The arboricity of a graph G, denoted $\alpha(G)$, is the minimal number of forests required to cover its edge set. Up to a factor of 2, it is equivalent to the average degree of the densest subgraph in G. Hence, arboricity is a natural and useful measure of density "everywhere". Graphs with bounded arboricity constitute an important and rich family of graphs, including planar graphs, minor-closed graphs, graphs with bounded treewidth, and preferential attachment graphs. On the applied side, in most real-world graphs the arboricity is at most an order of magnitude larger than the average degree, while the maximum degree could be up to three or four orders of magnitude larger [34, 30, 43]. Many applied algorithms exploit the property of bounded arboricity in order to design faster algorithms for clique and dense subgraph counting and listing [30, 33, 41, 37, 17, 9]. Furthermore, in a recent work, Eden, Mossel and Ron presented an algorithm for approximating the arboricity in sublinear time [21], whose output can be used as input to our algorithm (as an upper bound on the arboricity of the graph).

We seek algorithms that, given a parameter α and query access to a graph whose arboricity is upper bounded by α, "beat" the aforementioned lower bound when α is sufficiently bounded away from $\sqrt{m}$ (recall that the arboricity of a graph is always at most $\sqrt{m}$).

Such an algorithm was recently designed for the special case of sampling edges (2-cliques) almost uniformly. Here and elsewhere, when we say "almost uniformly" we mean in the strong sense of *pointwise-close* to uniform. Namely, where *each* element is returned with almost equal probability. We further discuss the benefits of this notion as compared to the strictly weaker notion of proximity with respect to the total variation distance in Section 1.1.2.

---

[1] Degree queries return the degree, $d(v)$, of a given vertex v; neighbor queries return the i^{th} neighbor of v for any given vertex v and $1 \leq i \leq d(v)$; and pair queries return whether there is an edge between a given pair of vertices.

[2] Throughout the introduction, when we discuss the "complexity" of previous results, we mean the running time of the algorithm. The query complexity is always bounded by the minimum between the running time and $n + m \leq n\alpha$.

Specifically, it is shown in [23] that the complexity of the almost-uniform edge sampling problem is $\Theta^*(n\alpha/m)$.[3] Comparing this to the $\Theta^*(n/\sqrt{m})$ complexity of the problem in general graphs [29, 45], exhibits an improvement by (roughly) a factor of $\sqrt{m}/\alpha$. In particular, for graphs with constant arboricity, this implies an exponential improvement, from $O^*(\sqrt{n})$ to $O^*(1)$. Similar improvements were obtained for the related question of approximately counting the number of k-cliques in the graph, where Eden, Ron and Seshadhri [26] obtained significant improvements for the class of bounded arboricity graphs, compared to the (essentially optimal) result for the general case [27].

In this work we show that, indeed, the complexity of the task of sampling k-cliques for *any* constant $k \geq 3$, is significantly better for the class of graphs of bounded arboricity, as compared to general graphs.

▶ **Theorem 1.1.** *Let $\varepsilon \in (0,1)$ be a constant. There exists an almost uniform sampling algorithm for k-cliques in graphs with arboricity at most α, that returns each k-clique in the graph with probability $\frac{(1\pm\varepsilon)}{n_k}$. Given a constant factor estimate of n_k,[4] the query complexity of the algorithm is*

$$O^*\left(\min\left\{n\alpha, \max\left\{\left(\frac{(n\alpha)^{k/2}}{n_k}\right)^{\frac{1}{k-1}}, \frac{n\alpha^{k-1}}{n_k}\right\}\right\}\right).$$

The running time is the same as the second term of the minimum.

While our upper bound on the complexity might seem unnatural at first glance, we also prove an almost-matching lower bound, thus resolving the complexity of the problem up to $(\log n/\varepsilon)^{O(k)}$ factors.

▶ **Theorem 1.2.** *Let $\mathcal{A}$ be an algorithm that given query access to a graph with arboricity at most α, returns each k-clique with probability $\Theta\left(\frac{1}{n_k}\right)$. Then the query complexity of $\mathcal{A}$ is*

$$\Omega^*\left(\min\left\{n\alpha, \max\left\{\left(\frac{(n\alpha)^{k/2}}{n_k}\right)^{\frac{1}{k-1}}, \frac{n\alpha^{k-1}}{n_k}\right\}\right\}\right).$$

We note that we chose not to parameterize our bounds in terms of m, but rather, only in terms of n, α, k and n_k. Hence, both the upper bound and the lower bound are stated for worst case m, which is $n\alpha$. We note that it is possible to obtain a finer expression that does depend on m for both bounds. However, for the sake of exposition, we chose not to include an additional parameter.

1.1 Discussion of the results

1.1.1 Comparison to previous results

For simplicity, assume that ε and k are constants, and ignore lower order terms. To compare the complexity of our algorithm to the upper bound of [32], consider, for example, the family of graphs G with n vertices, $m = n^{3/2}$ edges and arboricity $\alpha = n^{1/2}$, and assume that

[3] We use O^*, Ω^* and Θ^* to suppress a dependence on functions $g(\log n, k, 1/\varepsilon)$, which are at most $(\log n/\varepsilon)^{O(k)}$, where ε is the given approximation parameter.

[4] If the algorithm is not provided with an estimate of n_k, then an estimate of n_k can be obtained by applying the algorithm of [26] whose expected query complexity is dominated by the runtime of Theorem 1.1.

$k = 3$ and $n_3 = n^2$. Then the complexity of the problem in the general case grows like[5] $\tilde{\Theta}(m^{3/2}/n_3) = \tilde{\Theta}(n^{1/4})$, while we get $\Theta^*(n^{1/8})$. That is, we obtain a quadratic improvement, despite the fact that we work in a strictly weaker query model.

If we also allow access to uniform edge queries, then our algorithm can be adapted to run in time $O^*\left(\max\left\{\left(\frac{m^{k/2}}{n_k}\right)^{\frac{1}{k-1}}, \frac{m\alpha^{k-2}}{n_k}\right\}\right)$. To compare this to the $O(m^{k/2}/n_k)$ upper bound of [32] (for k-cliques), observe that if the first term in our bound is the dominant one, then we get the bound of [32] taken to the power of $1/(k-1)$. If the second term is the dominant one, then we improve on the bound of [32] by a factor of $(\sqrt{m}/\alpha)^{k-2}$ (recall that for every graph, $\alpha \leq \sqrt{m}$).

1.1.2 The importance of point-wise uniform sampling

In our results we measure "almost uniformity" with respect to pointwise distance between distributions. This notion of approximately uniform is a very strong one, as it requires every element to be returned with almost equal probability. In contrast, one could also consider the strictly weaker requirement that the distribution is close to uniform with respect to total variation distance (TVD). Here, it might be the case that the distributions assigns zero probability to an ε-fraction of the domain elements.

Sampling almost uniformly with respect to TVD may be sufficient in some contexts. However, there are scenarios in which the stronger notion of pointwise almost uniform sampling is crucial. Consider a domain in which each element has some significance score attributed to it, and assume that a small fraction of the domain elements have non-zero score and the others have score zero. If we have access to a distribution whose TVD distance to uniform is larger than the fraction of elements with positive score, then it is useless if we want to get any information regarding the (non-zero) significance scores of the elements. This is in contrast to having access to a distribution that is point-wise close to uniform, even for constant point-wise distance, where each element is returned with probability $\Theta(1/N)$, where N is the size of the domain. In the latter case, the probability of hitting non-zero score elements is slightly reduced as compared to uniform sampling, but does not fall to zero as in the former case.

For a more concrete example, consider protein networks, where cliques correspond to folding sites [18, 13, 1]. One can use point-wise sampling in order to get access to the folding sites of the protein at question, and then continue to further study their surrounding neighborhood. In TVD sampling however, it might be the case that exactly the folding sites of interest are the ones "missed" by the TVD sampler. Furthermore, as biological networks tend to huge and sparse [3, 39], allowing for improved results in bounded arboricity graphs is of major interest.

1.1.3 Approximate counting vs. point-wise sampling

We first observe that the complexities of approximately counting edges and point-wise almost uniformly sampling edges (in both bounded arboricity and general graphs) are of the same order. In contrast, for $k \geq 3$, the two complexities might differ significantly in bounded arboricity graphs. For example, consider the case of triangles ($k = 3$), $\alpha = O(1)$, and

[5] We note that the $\Omega(m^{k/2}/n_k)$ lower bound of [28] also holds for the task of sampling k-cliques almost uniformly.

$n_3 = \Theta(n)$. The complexity of sampling triangles almost uniformly is $\Theta^*(n^{1/4})$, while the complexity of approximately counting problem, is $O^*(1)$. This implies an exponential gap between the complexities of counting and sampling for certain ranges of parameters.[6] [7]

1.2 The high level ideas behind the clique-sampling algorithm

We start by briefly describing the ideas behind the algorithm of [23] for sampling edges almost uniformly, which we employ both as a subroutine and as a starting point of our algorithm for sampling k-cliques. We then turn to describe our algorithm. Throughout, we assume that an upper bound α on the arboricity of the input graph is known.

1.2.1 The edge sampling algorithm

Let L_0 be the set of all vertices in the graph with degree at most (roughly) α (so that almost all the vertices in the graph belong to L_0). The edge sampling algorithm samples a vertex v_0 uniformly at random, and if v_0 is in L_0, it performs a short random walk $v_0, v_1, \ldots, v_j$ of length j, for an index j chosen uniformly in $[\log n]$. If at any point the walk returns to L_0, then the algorithm aborts, and otherwise, it returns the last edge traversed.

The analysis of the algorithm relies on a layered decomposition of the graph vertices. The vertices in L_0 comprise the first layer. Subsequent layers are defined inductively: a vertex v is in L_j if (1) it is not in any of the layers L_i for $i < j$, and (2) most of its neighbors are in layers $L_0, L_2, \ldots, L_{j-1}$. While the algorithm is completely oblivious to the levels of the encountered vertices v_i for $i > 0$, using the aforementioned layering, it can be shown that each edge is sampled with almost equal probability $\approx \frac{1}{n\alpha}$.

1.2.2 The auxiliary graph H_k and the clique-sampling algorithm

In order to sample k-cliques in G, we first define an auxiliary graph H_k, whose edges correspond to k-cliques of G. Specifically, for each $(k-1)$-clique Q in G, there is a node v_Q in H_k, and for each k-clique C in G, there is a single edge in H_k between a pair of nodes $v_Q, v_{Q'}$ corresponding to two of its $(k-1)$-cliques, Q and Q'. Specifically, the first two $(k-1)$-cliques according to an ordering on all $(k-1)$-cliques, which will be defined later on. We say that C is *assigned* to Q and Q'. When $k = 2$, this assignment is uniquely determined (since every 2-clique (edge) contains exactly two 1-cliques (vertices)), and we have $H_2 = G$. For larger values of k, the assignment rule is such that Q and Q' both contain the vertex in C that has minimum degree in G. In general, since there is a one-to-one correspondence between the edges in H_k and the k-cliques of G, sampling an almost uniform edge in H_k is equivalent to sampling an almost uniform k-clique in G. An important observation is that if G has arboricity at most α, then so does H_k.

Given the aforementioned relation between k-cliques in G and edges in H_k, the basic underlying idea of our algorithm is emulating the edge sampling algorithm of [23] on the graph H_k, *while only having query access to the graph* G. Indeed this approach is natural (having defined H_k). However, emulating the edge sampling algorithm by performing random walks on H_k requires us to overcome several challenges:

[6] We note that the separation between approximately counting triangles and sampling triangles almost uniformly was already mentioned in [23] in passing as a preliminary result. However, [23] did not include any proof details nor a full characterization of the complexity of the sampling problem (for any $k \geq 3$).

[7] We believe that the algorithm of [26] can be adapted to sample a k-clique almost uniformly with respect to TVD with essentially the same complexity. However, this is not immediate.

1. We do not have query access to uniformly random nodes of H_k;
2. Determining whether a node in H_k is in layer L_0 cannot be performed by a single degree query (as was the case in [23]);
3. In order to sample a random neighbor of a node v_Q in H_k, we must sample a k-clique in G that is assigned to Q. (In [23] this could be implemented by a single neighbor query.) The emulation is "noisy" in the sense that it obtains only approximate answers to queries on H_k. In particular, it only estimates the degrees of nodes in H_k and selects nodes according to a distribution that is close to uniform. This is in contrast to the [23] algorithm, which gets precise answers to its queries. Hence, we must prove that the new algorithm still returns an edge (in H_k) that is close to uniform.
4. Emulating each query on H_k is implemented by performing multiple queries on G. Hence, one of the main challenges of this work is in bounding the query complexity of the clique-sampling algorithm.

We now outline how we address these challenges.

Addressing the challenges

Challenge 1: Emulating uniform node queries. The algorithm of [23] starts by sampling vertices uniformly at random in G. As stated previously, we do not have direct access to uniform node samples in H_k. Instead, in order to sample nodes in H_k, we recursively invoke our algorithm for sampling $(k-1)$-cliques in G almost uniformly. This results in a distribution that is only close to uniform, but we prove that this is sufficient for our needs.

Challenge 2: Determining whether a node belongs to $\mathbf{L_0(H_k)}$. Recall that in the edge sampling algorithm of [23], L_0 is the set of all vertices with degree roughly α. Therefore, in that algorithm, checking if a vertex belongs to L_0 requires a single degree query. In our setting, the degree of a node V_Q in H_k is equivalent to the number of k-cliques that are assigned to Q in G. Hence,

given a sampled node v_Q in H_k, we implement a procedure to check whether $v_Q \in L_0 = L_0(H_k)$, by trying to approximate the number of k-cliques that are assigned to Q in G. To do so efficiently, we replace the threshold α used to define L_0 in [23], by a value $\tau \geq \alpha$, where we will explain how τ is chosen later in the presentation.

Challenge 3: Emulating a random neighbor query. We next explain how we emulate a random neighbor query for a node v_Q in H_k (so as to emulate a random walk on H_k). Let $\mathcal{A}(Q)$ denote the set of k-cliques assigned to Q. By the definition of H_k, sampling an edge incident to v_Q translates to sampling a k-clique C in $\mathcal{A}(Q)$. Let u be the minimum degree vertex in Q, and define $d(Q) = d(u)$, where $d(u)$ is u's degree (in G). As explained above, for $k > 2$, the assignment rule is such that if a k-clique C is assigned to Q, then u is also the minimum degree vertex in C. Hence, in order to select a random neighbor of v_Q in H_k, we need only consider k-cliques C obtained from Q by adding a vertex with degree at least $d(u) = d(Q)$ (that neighbors all vertices in Q). By dealing separately with the case that $d(Q) \leq \sqrt{n\alpha}$ and the case that $d(Q) > \sqrt{n\alpha}$, we can design a procedure that for every $(k-1)$-clique Q samples each k-clique in $\mathcal{A}(Q)$ with probability (roughly) $\frac{1}{\min\{d(Q),\sqrt{n\alpha}\}}$ (and may fail to output any k-clique).

Given the above, to emulate a random neighbor query from a node v_Q in H_k such that $v_Q \notin L_0$ (so that $|\mathcal{A}(Q)| \geq \tau$), we repeat the above sampling attempts $O^*\left(\left\lceil \frac{\min\{d(Q),\sqrt{n\alpha}\}}{\tau} \right\rceil\right)$ times. This process succeeds in obtaining a uniformly distributed k-clique in $\mathcal{A}(Q)$ with high probability. For a node v_Q in L_0 (where we don't have a lower bound on $|\mathcal{A}(Q)|$), performing this number of attempts implies that each k-clique in $\mathcal{A}(Q)$ is obtained with probability $1/\tau$.

An inductive analysis shows that a single invocation of the above emulation of the random walk on H_k returns each k-clique in G with probability roughly $\frac{1}{n\alpha \cdot \tau^{k-2}}$. The $(n\alpha)$ term in the denominator comes from the base of the induction, i.e., sampling a uniform 2-clique (edge) in G, and the term τ^{k-2} stems from the $k-2$ recursive calls, where in each level of recursion, we "lose" a factor of $1/\tau$. Therefore, the overall success probability of a single attempt to sample an edge in H_k is roughly $\frac{n_k}{n\alpha \cdot \tau^{k-2}}$. Hence, $O^*(\frac{n\alpha \cdot \tau^{k-2}}{n_k})$ repetitions are sufficient so that, with high probability, an almost uniformly distributed k-clique in G is returned.

Challenge 4: Proving correctness. Given the above approach, we are able to emulate the basic algorithm of [23] on the auxiliary graph H_k. Hence, to prove correctness we follow the ideas of [23]. However, since the emulation on H_k results in "noisy" answers to node, degree and neighbor queries, so that we cannot immediately rely on the correctness of the algorithm of [23]. Hence, we must (re-)prove that the emulation algorithm induces a distribution on the edges (of H_k) that is close to uniform. This is done by carefully keeping track of the divergence from uniformity that is caused due to the noisy answers to queries throughout the execution of the algorithm. We note that the main challenge lies not in the proof of correctness, but rather in bounding the complexity of the clique-sampling algorithm, as discussed next.

Challenge 5: Bounding the query complexity. As discussed above, to sample a k-clique in G with high probability, we perform $t = O^*\left(\frac{n\alpha \cdot \tau^{k-2}}{n_k}\right)$ repetitions of the random walk emulation on H_k. In each such emulation, there is a sequence of $k-1$ recursive calls to sample i-cliques for $i \in [2, \ldots, k]$ by performing a random walk on the graph H_i. Whenever a random neighbor query is emulated on a node in H_i for $i > 2$, $r = O^*\left(\frac{\min\{d(T), \sqrt{n\alpha}\}}{\tau}\right)$ queries are performed in G. Conditioned on τ being sufficiently larger than α, we get that the expected number of queries in each such emulation is just $O^*(1)$ (while the maximum is $O^*\left(\frac{\sqrt{n\alpha}}{\tau}\right)$). This implies that the expected total query complexity is $O^*\left(\frac{n\alpha \cdot \tau^{k-2}}{n_k}\right)$. As for the maximum running time, we can get an upper bound of $O^*\left(\frac{n\alpha \cdot \tau^{k-2}}{n_k} + \frac{\sqrt{n\alpha}}{\tau}\right)$ by aborting the algorithm if it performs a larger number of queries, while still obtaining an output distribution as desired.

Hence, we get a certain tradeoff between the expected query complexity and the maximum one (for "hard to sample" cliques). In particular, if we set $\tau = \Theta^*(\alpha)$, we get that the expected query complexity is $O^*\left(\frac{n\alpha^{k-1}}{n_k}\right)$, as in the case of counting, while the maximum query complexity is $O^*\left(\frac{n\alpha^{k-1}}{n_k} + \sqrt{n/\alpha}\right)$. The upper bound in Theorem 1.1 is derived by setting τ so that the two summands in the expression $O^*\left(\frac{n\alpha \cdot \tau^{k-2}}{n_k} + \frac{\sqrt{n\alpha}}{\tau}\right)$ are equal.

1.3 A discussion of related random-walk based sampling algorithms

The idea of sampling k-subgraphs (i.e., subgraphs of size k) from a graph G using random walks on an auxiliary graph (in which nodes and edges correspond to subgraphs), is not new, see, e.g., [7, 47, 42, 12, 11]. However, our approach, and in particular our definition of the auxiliary graph H_k, differs from previous ones in several ways, which are crucial for sampling k-cliques according to a distribution that is ε-pointwise close to uniform, with sublinear query complexity. Below we discuss the main aspects of difference between our approach and the aforementioned previous works.

1. The task. We start by noting that in the aforementioned random-walk based works, the focus was on sampling from a distribution whose support is *all* connected k-subgraphs, while we focus on sampling from a distribution whose support is the set of k-cliques. An algorithm for sampling k-subgraphs can be directly adapted to output only k-cliques using rejection sampling, however this could significantly increase the complexity.
2. The distance measure. In most previous works, the considered distance measure is the total variation distance, while our result considers the strictly stronger pointwise distance measure.[8]
3. The definition of the auxiliary graph. The auxiliary graph considered in the aforementioned works, which we denote here by H'_k, is defined as follows. There is a node in H'_k for each subgraph of size $k-1$ in G, and there is an edge between two nodes in H'_k if the two corresponding $(k-1)$-subgraphs differ by a single vertex. Hence, similarly to our auxiliary graph, H_k, edges in the auxiliary graph correspond to the objects that we would like to sample (k-subgraphs in previous works, and k-cliques in ours). However, the way we define the edge-set of H_k is pivotal to the analysis of our algorithm. In particular, we put an edge between two nodes in H_k not only if the union between the corresponding $(k-1)$ cliques is a k-clique, but also if this k-clique is assigned to the two $(k-1)$-cliques. Our assignment rule is tailored to bound the query complexity of the algorithm (based on the degrees of vertices in the cliques). Also note that if the original graph G is connected, then so is H'_k, while H_k is typically not connected.
4. Performing random walks on the auxiliary graph. Recall that in our context, where we are given only query access to G and would like to minimize the number of queries, we have to overcome several challenges in the emulation of random walks on H_k. These do not arise in previous works: The random walk starts from an arbitrary node in H'_k (an arbitrary $(k-1)$-subgraph), and each step in the walk is simply implemented by selecting a random neighbor of one of the vertices in the current $(k-1)$-subgraph.
5. The complexity of the sampling algorithm. As noted above, our focus is on bounding the query complexity of the algorithm, and indeed we get an almost tight bound based on our definition of H_k and the details of the emulation of random walks on H_k given query access to G. In the aforementioned works, the complexity of the algorithms was shown to depend on the mixing time of H'_k, and one of the main challenges of these works is in analyzing it. Indeed, in [11] it is proved that even if the mixing time of G is relatively small, the mixing time of H'_k may be a factor of $\rho(G)^{k-2}$ larger, where $\rho(G)$ is the ratio between the maximum and minimum degree in G (and hence may be large (e.g., $\Omega(n)$) even in bounded-arboricity graphs).

1.4 Overview of the lower bound

The first and last terms in the lower bound of Theorem 1.2 follow directly from a lower bound of $\Omega^*\left(\min\left\{n\alpha, \frac{n\alpha^{k-1}}{n_k}\right\}\right)$ by [26] for the task of approximately counting the number of k-cliques. They prove that any algorithm that performs fewer queries than the above expression cannot distinguish with high probability between two families of graphs, one with n_k k-cliques, and one with no k-cliques. It follows that any uniform sampling algorithm cannot perform fewer queries. Therefore, our main focus is on proving the term $\Omega^*\left(\left(\frac{(n\alpha)^{k/2}}{n_k}\right)^{\frac{1}{k-1}}\right)$ (which, as noted previously, may be much larger than the $\Omega^*\left(\frac{n\alpha^{k-1}}{n_k}\right)$ term).

[8] In [11], Bressan combined a random-walk based approach with a (linear in m) preprocessing of the graph, in order to obtain an exact uniform sampler.

To this end, we construct a family of graphs (with arboricity at most α), such that in each graph, among the n_k k-cliques that it contains, there is one "hidden" k-clique. This clique is hidden in the sense that any algorithm that (always) performs less than $\left(\frac{(n\alpha)^{k/2}}{n_k}\right)^{\frac{1}{k-1}}/c$ queries (for a sufficiently large constant c) cannot sample this clique with probability $\Omega(1/n_k)$.[9]

The above idea is formalized by defining a process that answers the queries of a sampling algorithm "on the fly" while constructing a random graph in the family. All graphs in the family have the same underlying structure, and they differ in the choice of clique vertices and in the labeling of (part of) the edges. Here we give the high-level idea of the underlying structure, and the intuition for the lower-bound expression.

In each graph in the family, the hidden clique is over a subset S of k vertices that all have (high) degree $\Theta(\ell)$ where $\ell = \sqrt{n\alpha}$. The total number of high-degree vertices is $\Theta(\ell)$ as well (so that all graphs graphs in the family have $\Theta(\ell^2) = \Theta(n\alpha)$ edges[10]). Other than the clique edges, there are no other edges between the high-degree vertices. See Figure 1 for an illustration. Intuitively, in order to reveal the hidden clique, the algorithm must first reveal one edge (u, u') in the clique and then reveal $k-2$ additional edges between u and the other edges in the clique.[11] We prove that in each query, the probability of revealing the first edge of the clique is $O(k^2/\ell^2)$, and the probability of revealing any consecutive edge is $O(k/\ell)$.[12]

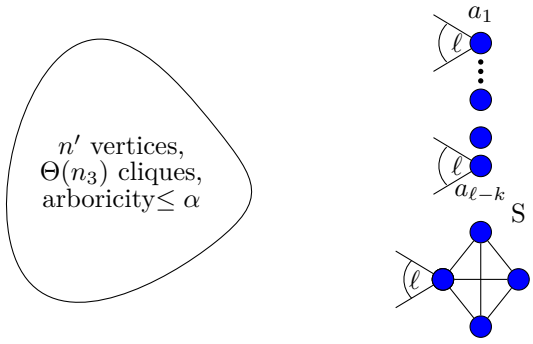

Figure 1 The underlying structure of the graphs in the lower bound construction for $k = 4$. For a more detailed description and figure, see the full version of this paper [24].

The intuition for the upper bound $O(k^2/\ell^2)$ on the probability of revealing the first edge is that the number of clique edges is $\binom{k}{2}$, while the total number of edges is $\Theta(\ell^2)$. Similarly, the rough intuition for the upper bound of $O(k/\ell)$ on revealing each additional edge in the clique is that each clique vertex has $k-1$ neighbors in the clique and a total of $\Theta(\ell)$ neighbors. In order to provide a formal argument, we define an auxiliary bipartite graph

[9] We note that this does not preclude the possibility that the expected complexity of the algorithm is smaller (as discusses in Subsection 1.2.2).

[10] As stated in the introduction, we note that since all of the graphs have $n\alpha$ edges, our lower bound does not exclude the possibility of a refined upper bound that also depends on the number of edges m. (And indeed, it is possible to replace some of the $n\alpha$ terms in the upper bound with m terms. However, we chose not to further complicate the exposition of the algorithm and therefore we also present the lower bound in terms of worst-case m.)

[11] The algorithm may alternatively try to reveal $k/2$ edges in the clique that do not have common endpoints (or some other combination of edges that together reveals all clique vertices), but this is not advantageous for the algorithm.

[12] We note that whenever the term $\frac{(n\alpha)^{k/(2(k-1))}}{kn_k^{1/(k-1)}}$ dominates the last term in the lower bound of Theorem 1.2, it is smaller than $\sqrt{n/\alpha}$.

whose nodes correspond to graphs that are consistent with all previous queries (and answers) and either contain a "witness" clique edge that corresponds to the query of the algorithm (one side of the graph), or do not (the other side). The edges of the bipartite graph are defined by certain transformations from witness graphs to non-witness graphs. By analyzing the degrees of nodes on both sides of this auxiliary graph, we obtain the aforementioned bounds on the probability of revealing edges in the hidden clique.

Given these probability upper bounds, if an algorithm performs T queries, then the probability that it reveals the hidden clique is upper bounded by $T \cdot \frac{k^2}{\ell^2} \cdot \left(T \cdot \frac{k}{\ell}\right)^{k-2}$. If we want this expression to be $\Omega(1/n_k)$, the number of queries T must be $\Omega^*\left(\left(\frac{(n\alpha)^{k/2}}{n_k}\right)^{1/(k-1)}\right)$.

1.5 Related Work

The works most related to ours were mentioned in earlier subsections of the introduction. In Appendix A, we give a broader view of recent advances on sublinear-time approximate counting and uniform sampling algorithms.

1.6 Organization

We start with some preliminaries in Section 2. Due to the page limitation, in this extended abstract we only describe the algorithm and a sketch of the analysis for the case of $k = 3$ (triangles) – see Section 3.

The full algorithm and analysis for the general case, as well as the lower bound, can be found in the full version of this paper [24].

2 Preliminaries

Let $G = (V, E)$ be a graph over n vertices and arboricity at most α. Each vertex $v \in V$ has a unique id in $[n]$, denoted $id(v)$. Let $\mathcal{C}_k$ denote the set of k-cliques of G, and let $n_k = |\mathcal{C}_k|$. For a vertex v, let $\Gamma(v) = \Gamma_G(v)$ denote its set of neighbors and let $d(v) = d_G(v) = |\Gamma(v)|$. We sometimes refer to edges as oriented, meaning that we consider each edge from both its endpoints.

Access to G is given via the following types of queries: (1) A degree query, $\mathsf{deg}(v)$, returns the degree $d(v)$ of the vertex v; (2) A neighbor query, $\mathsf{nbr}(v, i)$ for $i \in [d(v)]$, returns the i^{th} neighbor of v; (3) A pair query, $\mathsf{pair}(v, v')$, returns whether $(v, v') \in E$.

▶ **Definition 2.1** (Ordering of the vertices.). *We define an ordering on the graph's vertices, where $u \prec v$ if $d(u) < d(v)$ or if $d(u) = d(v)$ and $id(u) < id(v)$.*

▶ **Definition 2.2** (Cliques' degree and neighbors). *For a k-clique C, let v be the minimal vertex in v_C according to $\prec$. We define the degree of C in G to be $d(C) = d(v)$. We define the neighbor-set of C, denoted $\Gamma(C) = \Gamma(v)$, to be the set of v's neighbors in G.*

▶ **Definition 2.3** (Cliques id and an ordering of cliques). *For a k-clique C, let its id, $id(C)$ be a concatenation of its vertices ordered by $\prec$. We extend the order $\prec$ to cliques, so that for two k cliques C, C', $C \prec C'$ if $d(C) < d(C')$ or if $d(C) = d(C')$ and $id(C) < id(C')$.*

▶ **Definition 2.4** (Assignment of k-cliques to $(k-1)$-cliques). *We* assign *each k-clique C to its two first $(k-1)$-cliques according to $\prec$. For every $(k-1)$ clique Q, we denote its set of assigned k-cliques by $\mathcal{A}(Q)$, and let $a(Q) = |\mathcal{A}(Q)|$. We refer to $a(Q)$ as the assigned cliques degree of Q.*

Note that for every $k \geq 3$, $a(Q) \leq d(Q)$.

▶ **Observation 2.5.** *By Definition 2.4, for $k \geq 3$, if Q and Q' are assigned a k-clique C, then $d(Q) = d(Q') = d(C)$. Hence, if a k-clique C is assigned to a $(k-1)$-cliques Q such that $C = Q \cup \{w\}$, then $d(Q) = d(C) \leq d(w)$.*

We shall sometimes abuse notation and let $\{Q, u\}$ denote $Q \cup \{w\}$.

We are now ready to define the auxiliary graph H_k, which is central to our algorithm.

▶ **Definition 2.6** (The graph H_k)**.** *Given a graph G, we define the graph $H_k(G) = H_k = (V_{H_k}, E_{H_k})$ as follows. For every $(k-1)$-clique Q in G there is a node v_Q in V_{H_k}. For every k-clique C in G, there is an edge in H_k between the two $(k-1)$-cliques that C is assigned to, as defined in Definition 2.4.*

For the sake of clarity, throughout the paper, we refer to the vertices of H_k as nodes. Note that for the special case of $k = 2$, we have that $H_2(G) = G$, and each edge (2-clique) in G, is assigned to both its endpoints. More generally, Definition 2.6 implies a one-to-one correspondence between the set of edges incident to a node v_Q in $H_k(G)$ and the set $\mathcal{A}(Q)$ of k-cliques assigned to Q in G. This in turn implies that the degree of a node v_Q in $H_k(G)$ equals the assigned cliques degree of Q, $a(Q)$. By the comment following Definition 2.4, the degree of v_Q in $H_k(G)$ is upper bounded by the degree of Q in G.

The last claim in this section concerns the arboricity of $H_k(G)$.

▷ Claim 2.7. Let G be a graph of arboricity at most α. Then $H_k(G)$ has arboricity at most α.

3 The case of $k = 3$: sampling triangles

As a warmup, in this section we describe our algorithm for the case of sampling triangles and provide the structure of its analysis. To ease readability, some of the claims we present are loosely stated (the precise and general claims appear (and are proved) in the next section). Since the graph G is fixed throughout the presentation, we use the shorthand H_3 for $H_3(G)$.

In addition to receiving as input n, α, and ϵ, as well as being given query access to G, the sampling algorithm, Sample-Triangle, receives the parameter $\overline{n}_3$, which is assumed to be a constant factor estimate of the number of triangles, n_3. Such an estimate can be obtained by running the algorithm of [26], without asymptotically increasing the expected complexity of our algorithm.

To sample a triangle in G, the algorithm Sample-Triangle repeatedly invokes the procedure Sample-Edge-Auxiliary-Tri on the graph H_3, while ensuring that the number of queries does not exceed a certain threshold. For the sake of conciseness, from this point on we view the parameters that Sample-Triangle receives, as global variables for all other procedures.

Sample-Triangle$(n, \alpha, \varepsilon, \overline{n}_3)$

1. Set $\tau = \max\left\{\frac{\sqrt{\overline{n}_3}}{(n\alpha)^{1/4}}, \alpha\right\}$.
2. While the number of queries does not exceed $r = c \cdot \min\left\{n\alpha, \max\left\{\frac{\sqrt{n\alpha}}{\tau}, \frac{n\alpha\tau}{\overline{n}_3}\right\}\right\}$ for a sufficiently large constant c:
   a. Invoke Sample-Edge-Auxiliary-Tri(τ), and if an edge in H_3 is returned, then **return** the corresponding 3-clique (triangle) in G.

▶ **Theorem 3.1** (loosely stated)**.** *The algorithm* Sample-Triangle *is a pointwise ε-close to uniform sampling algorithm for triangles (3-cliques) in graphs with arboricity at most α. The query complexity of the algorithm is $O^*\left(\min\left\{n\alpha, \max\left\{\left(\frac{(n\alpha)^{3/4}}{\sqrt{n_3}}\right), \frac{n\alpha^2}{n_3}\right\}\right\}\right)$.*

We defer the proof of the theorem to the end of this section, and continue to describe the procedure Sample-Edge-Auxiliary-Tri. This procedures (tries to) return an (almost) uniformly distributed edge in the auxiliary graph H_3 (corresponding to a triangle in G), so that each edge is returned with probability (roughly) $\frac{1\pm\Theta(\varepsilon)}{n\alpha\tau}$, and it is the main procedure used in order to prove Theorem 3.1.

As will be explained in more detail momentarily, the setting of τ in Sample-Triangle (together with the random coins of the algorithm) determines a set $L_0(H_3)$ of nodes in H_3. Recall that the degree of a node v_Q in H_3 (where Q is a 2-clique, i.e., an edge in G) is the number of triangles (3-cliques) that are assigned to Q according to the assignment rule of Definition 2.4 (denoted $a(Q)$). With high probability over the randomness of the algorithm, all nodes $v_Q \in H_3$ whose degree (in H_3) at most τ belong to $L_0(H_3)$, and all nodes $v_Q \in H_3$ with degree greater than 2τ do not belong to $L_0(H_3)$ (the rest of the nodes can belong to either set). We use $E(L_0(H_3))$ to denote the edges in H_3 that are incident to nodes in $L_0(H_3)$.

Sample-Edge-Auxiliary-Tri first invokes the subroutine Sample-Edge-L_0-Auxiliary-Tri that either returns a (close to) uniform edge (v_0, v_1) among the edges of $E(L_0(H_3))$ or returns FAIL. The procedure then chooses an index j in $[0, \ldots, \log(n\alpha)]$ uniformly at random, and performs a random walk of length j on H_3, by traversing at each step to a uniformly selected neighbor of the last visited node. This is done by invoking the procedure Sample-Neighbor-Auxiliary-Tri. If at any point the last visited node belongs to $L_0(H_3)$ (which is verified by the procedure Define-L_0-Auxiliary-Tri), then the procedure fails. Otherwise, the last traversed edge in the walk is returned.

Sample-Edge-Auxiliary-Tri(τ).

1. Set $s = \log(n\alpha)$ and set $\beta = \varepsilon/(2s+2)$.
2. Invoke Sample-Edge-L_0-Auxiliary-Tri(β, τ), and let $e_0 = (v_{Q_0}, v_{Q_1})$ be the returned edge if one was returned. Otherwise, **return** FAIL.
3. Choose $j \in [0, \ldots, s]$ uniformly at random.
4. For $i = 1$ to j do:
   a. If Define-L_0-Auxiliary-Tri($v_{Q_i}, \delta = \beta/\overline{n}_3, \beta, \tau$)=YES, then **return** FAIL.
   b. Invoke Sample-Neighbor-Auxiliary-Tri(v_{Q_i}, β, τ) to sample an edge $(v_{Q_i}, v_{Q_{i+1}})$ in H_3.
5. **Return** $(v_{Q_j}, v_{Q_{j+1}})$.

▶ **Lemma 3.2** (loosely stated). *The procedure* Sample-Edge-Auxiliary-Tri *returns an edge in H_3 so that each edge is returned with probability (roughly) $\frac{(1\pm\varepsilon)}{n\alpha\tau}$. Furthermore, the expected running time of a single invocation of the procedure is $O^*(1)$, and the maximum running time is $O^*(\sqrt{n\alpha}/\tau)$.*

Before presenting the subroutines Sample-Edge-L_0-Auxiliary-Tri, Sample-Neighbor-Auxiliary-Tri and Define-L_0-Auxiliary-Tri invoked in Sample-Edge-Auxiliary-Tri, we describe a simple procedure, Samp-High-Deg-Nbr, used by these subroutines. The procedure gets a 2-clique (edge) Q as input, and tries to sample a higher degree neighbor of Q in G, so that each such neighbor is returned with probability $1/\min\{d(Q), \sqrt{n\alpha}\}$. As mentioned in the introduction, we shall make use of the procedure Sample-Basic-Edge by [29], that returns every edge in G with probability (roughly) $(1 \pm \beta)/(n\alpha)$, given an approximation parameter β.

Samp-High-Deg-Nbr(Q, β).

1. Let u be the min degree vertex of Q, and query $d(u)(= d(Q))$.
2. If $d(Q) \leq \sqrt{n\alpha}$, then query the i^{th} neighbor of u in G, for a uniformly selected index $i \in d(u)$. If $d(w) \geq d(Q)$, then return w.
3. If $d(Q) > \sqrt{n\alpha}$, then:
   a. Invoke Sample-Basic-Edge(β) and if an edge is returned then denote it (w, x). Otherwise, return FAIL.
   b. Query $d(w)$ and if $d(w) > d(Q)$, then return the endpoint w with probability $d(w)/\sqrt{n\alpha}$. Otherwise, return FAIL.

▷ Claim 3.3 (loosely stated). Let Q be a 2-clique (edge) in G. The procedure Samp-High-Deg-Nbr either returns a neighbor of Q in G (that is, a neighbor in G of the min-degree vertex of Q), or fails. Each $w \in \Gamma(Q)$ such that $d(w) \geq d(Q)$ is returned with (roughly) equal probability $(1 \pm \beta)/\min\{d(Q), \sqrt{n\alpha}\}$. The query and time complexity of the procedure are $O^*(\log n)$.

We turn to present the subroutines used by Sample-Edge-Auxiliary-Tri, starting with the subroutine Define-L_0-Auxiliary-Tri, that defines the aforementioned set $L_0(H_3) \subseteq V(H_3)$. Namely, $L_0(H_3)$ is determined according to the output of the subroutine, so that

$$L_0(H_3) = \{v_Q \in V_{H_3} \ : \ \text{Define-}L_0\text{-Auxiliary-Tri}(v_Q, \delta, \beta, \tau) = \ \text{YES} \ \}$$

(where we assume that the randomness of the subroutine is uniquely determined for each v_Q). Hence Define-L_0-Auxiliary-Tri can be thought of as an *oracle* that returns whether a given v_Q belongs to $L_0(H_3)$ or not.

Define-L_0-Auxiliary-Tri$(v_Q, \delta, \beta, \tau)$.

1. Let Q be the 2-clique (edge) in G corresponding to v_Q.
2. For $i = 1$ to $r = \frac{\min\{d(Q), \sqrt{n\alpha}\}}{\tau} \cdot \ln(1/\delta)$ do:
   a. Invoke Samp-High-Deg-Nbr$(Q, \beta/10)$ to (try and) sample a neighbor w_i of Q.
   b. If Q and w_i form a triangle C, and C is assigned to Q, then let $\chi_{w_i} = 1$.
3. Let $\tilde{a} = \frac{1}{r} \sum_{i=1}^{r} \chi_{w_i}$.
4. If $\tilde{a} < 1.5\tau/d(Q)$ then return YES. Otherwise, return NO.

Recall that $\mathcal{A}(Q)$ is the set of cliques assigned to Q, and $a(Q) = |\mathcal{A}(Q)|$.

▷ Claim 3.4 (loosely stated). With high probability Define-L_0-Auxiliary-Tri determines a set $L_0(H_3)$ so that the following holds for every $v_Q \in H_3$.

- If $a(Q) \leq \tau$, then $v_Q \in L_0(H_3)$ and the subroutine returns YES.
- If $a(Q) > 2\tau$, then $v_Q \notin L_0(H_3)$ and the subroutine returns NO.

Furthermore, the query and time complexities of the subroutine are $O^*\left(\frac{\min\{d(Q), \sqrt{n\alpha}\}}{\tau}\right)$.

Given Claim 3.4, from here on we assume that for every $Q \in L_0(H_3)$, $dt(Q) \leq 2\tau$, and for every $Q \notin L_0(H_3), a(Q) > \tau$.

We now present the subroutine Sample-Edge-L_0-Auxiliary-Tri that is used for sampling edges in H_3 that are incident to nodes in $L_0(H_3)$. In order to sample a uniform edge in $E(L_0(H_3))$, we first need to sample a node in $L_0(H_3)$. Recall that the nodes of H_3 correspond to edges in G. Hence, to sample a node in $L_0(H_3)$, the procedure first invokes Sample-Basic-Edge to sample an edge in G (almost) uniformly at random, and then checks if the corresponding node is in $L_0(H_3)$ (by invoking Define-L_0-Auxiliary-Tri).

Sample-Edge-L_0-Auxiliary-Tri(β, τ).

1. Invoke Sample-Basic-Edge$(\beta/4)$. Let Q be the returned 2-clique if one was returned. Otherwise FAIL.
2. If Define-L_0-Auxiliary-Tri$(v_Q, \delta, \beta/4, \tau)$=NO then FAIL.
3. Repeat at most $r = O^*\left(\frac{\min\{d(Q), \sqrt{n\alpha}\}}{2\tau}\right)$ times or until a neighbor is sampled:
   a. Invoke Samp-High-Deg-Nbr$(Q, \beta/10)$ to (try and) sample a neighbor w of Q.
4. If no neighbor is sampled, then return FAIL.
5. Check if $\{Q, w\}$ is a triangle assigned to Q. If so, return $C = \{Q, w\}$.

▷ Claim 3.5. Consider an invocation of Sample-Edge-L_0-Auxiliary-Tri with parameters β and τ. The subroutine Sample-Edge-L_0-Auxiliary-Tri returns every edge in $E(L_0(H_3))$ with probability (roughly) $(1 \pm \beta)/(2n\alpha\tau)$. The expected query and time complexity of the subroutine are $O^*(1)$ and the maximum query and time complexity are $O^*\left(\frac{\sqrt{n\alpha}}{\tau}\right)$.

Finally, the subroutine Sample-Neighbor-Auxiliary-Tri (tries to) sample a neighbor of a node v_Q in the auxiliary graph H_3. That is, it tries to sample a triangle $C \in \mathcal{A}(Q)$.

Sample-Neighbor-Auxiliary-Tri(v_Q, β).

1. Let Q be the 2-clique (edge) in G corresponding to v_Q.
2. Repeat at most $r = \frac{\min\{d(Q), \sqrt{n\alpha}\}}{\tau} \cdot \ln(1/\beta)$ times:
   a. Invoke Samp-High-Deg-Nbr$(Q, \beta/10)$ to (try and) sample a neighbor w of Q.
   b. If Q and w form a triangle C, and C is assigned to Q, then **return** the edge $(v_Q, v_{Q'})$ in H_3 that corresponds to C.

▷ Claim 3.6 (loosely stated). For a given node $v_Q \in V(H_3)$ such that $Q \notin L_0(H_3)$, with probability at least $1 - \beta$, the subroutine Sample-Neighbor-Auxiliary-Tri returns a neighbor of v_Q in H_3, so that each neighbor of v_Q is returned with probability (roughly) $(1 \pm \beta)/a(Q)$. The query and time complexity of the subroutine are $O^*\left(\frac{\min\{d(Q), \sqrt{n\alpha}\}}{\tau}\right)$.

Finally, we sketch the proof of (the loosely stated) Theorem 3.1.

Proof sketch of Theorem 3.1. By Lemma 3.2, Sample-Edge-Auxiliary-Tri returns every specific edge in H_3 with probability (roughly) $(1 \pm \varepsilon)/(n\alpha\tau)$. Since there is a one-to-one correspondence between edges in H_3 to triangles in G, this implies that each triangle is returned with probability (roughly) $(1 \pm \varepsilon)/(n\alpha\tau)$, and that with probability (roughly) $n_3/(n\alpha\tau)$, some triangle is returned. Hence, the expected number of invocations of the loop is $O(n\alpha\tau/n_3)$. Furthermore, by Claim 3.2, the expected complexity of each invocation is $O^*(1)$. It can be proven using standard concentration bounds, that with high probability, a triangle will be returned before the number of allowed queries r is exceeded. Therefore, with high probability, a triangle is returned, and since all triangle are almost equally likely to be the returned one, it holds that each triangle is returned with probability (roughly) $1/n_3$.

Furthermore, since the expected complexity of each invocation of Sample-Edge-Auxiliary is $O^*(1)$ and the maximum is $O^*(\sqrt{n\alpha}/\tau)$, it follows that the query and time complexity of Sample-Triangle is $O^*(r + \sqrt{n\alpha}/\tau) = O^*\left(\min\left\{n\alpha, \max\left\{\frac{\sqrt{n\alpha}}{\tau}, \frac{n\alpha\tau}{\overline{n}_3}\right\}\right\}\right)$. Plugging $\tau = \max\left\{\frac{\sqrt{\overline{n}_3}}{(n\alpha)^{1/4}}, \alpha\right\}$, we get that the query complexity of Sample-Triangle is bounded by

$$O^*\left(\min\left\{n\alpha, \max\left\{\frac{(n\alpha)^{3/4}}{\sqrt{n_3}}, \frac{n\alpha^2}{n_3}\right\}\right\}\right),$$

as claimed. ◀

References

1. Balázs Adamcsek, Gergely Palla, Illés J Farkas, Imre Derényi, and Tamás Vicsek. Cfinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22(8):1021–1023, 2006.
2. Maryam Aliakbarpour, Amartya Shankha Biswas, Themis Gouleakis, John Peebles, Ronitt Rubinfeld, and Anak Yodpinyanee. Sublinear-time algorithms for counting star subgraphs via edge sampling. *Algorithmica*, 80(2):668–697, 2018.
3. Uri Alon. *An introduction to systems biology: design principles of biological circuits*. Chapman and Hall/CRC, 2006.
4. Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A Simple Sublinear-Time Algorithm for Counting Arbitrary Subgraphs via Edge Sampling. In Avrim 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, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2019.6.
5. Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge estimation with independent set oracles. *ACM Transactions on Algorithms (TALG)*, 16(4):1–27, 2020.
6. Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Triangle estimation using tripartite independent set queries. In *30th International Symposium on Algorithms and Computation (ISAAC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.
7. Mansurul A Bhuiyan, Mahmudur Rahman, Mahmuda Rahman, and Mohammad Al Hasan. Guise: Uniform sampling of graphlets for large graph analysis. In *2012 IEEE 12th International Conference on Data Mining*, pages 91–100. IEEE, 2012.
8. Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Bipartite independent set oracles and beyond: Can it even count triangles in polylogarithmic queries? *arXiv preprint*, 2021. arXiv:2110.03836.
9. Amartya Shankha Biswas, Talya Eden, Quanquan C. Liu, Slobodan Mitrovic, and Ronitt Rubinfeld. Parallel algorithms for small subgraph counting. *CoRR*, abs/2002.08299, 2020. arXiv:2002.08299.
10. Amartya Shankha Biswas, Talya Eden, and Ronitt Rubinfeld. Towards a decomposition-optimal algorithm for counting and sampling arbitrary motifs in sublinear time. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 55:1–55:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.55.
11. Marco Bressan. Efficient and near-optimal algorithms for sampling connected subgraphs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 1132–1143, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451042.
12. Marco Bressan, Flavio Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. Motif counting beyond five nodes. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(4):1–25, 2018.
13. Broto Chakrabarty and Nita Parekh. Naps: network analysis of protein structures. *Nucleic acids research*, 44(W1):W375–W382, 2016.
14. Jianer Chen, Xiuzhen Huang, Iyad A Kanj, and Ge Xia. Strong computational lower bounds via parameterized complexity. *Journal of Computer and System Sciences*, 72(8):1346–1367, 2006.
15. Xi Chen, Amit Levi, and Erik Waingarten. Nearly optimal edge estimation with independent set queries. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2916–2935. SIAM, 2020.

16 Xiaowei Chen, Yongkun Li, Pinghui Wang, and John CS Lui. A general framework for estimating graphlet statistics via random walk. *Proceedings of the VLDB Endowment*, 10(3), 2016.

17 Maximilien Danisch, Oana Balalau, and Mauro Sozio. Listing k-cliques in sparse real-world graphs. In *Proceedings of the 2018 World Wide Web Conference*, pages 589–598, 2018.

18 Dhruba Deb, Saraswathi Vishveshwara, and Smitha Vishveshwara. Understanding protein structure from a percolation perspective. *Biophysical journal*, 97(6):1787–1794, 2009.

19 Nurcan Durak, Ali Pinar, Tamara G. Kolda, and C. Seshadhri. Degree relations of triangles in real-world networks and graph models. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1712–1716, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2396761.2398503.

20 Talya Eden, Amit Levi, Dana Ron, and C Seshadhri. Approximately counting triangles in sublinear time. In *Foundations of Computer Science , 2015 IEEE 56th Annual Symposium on*, pages 614–633. IEEE, 2015.

21 Talya Eden, Saleet Mossel, and Dana Ron. Approximating the arboricity in sublinear time. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 2404–2425. SIAM, 2022. doi:10.1137/1.9781611977073.96.

22 Talya Eden, Saleet Mossel, and Ronitt Rubinfeld. Sampling multiple edges efficiently. In Mary Wootters and Laura Sanità, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, August 16-18, 2021, University of Washington, Seattle, Washington, USA (Virtual Conference)*, volume 207 of *LIPIcs*, pages 51:1–51:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.APPROX/RANDOM.2021.51.

23 Talya Eden, Dana Ron, and Will Rosenbaum. The arboricity captures the complexity of sampling edges. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece.*, pages 52:1–52:14, 2019. doi:10.4230/LIPIcs.ICALP.2019.52.

24 Talya Eden, Dana Ron, and Will Rosenbaum. Almost optimal bounds for sublinear-time sampling of k-cliques: Sampling cliques is harder than counting. *CoRR*, abs/2012.04090, 2020. arXiv:2012.04090.

25 Talya Eden, Dana Ron, and C. Seshadhri. Sublinear time estimation of degree distribution moments: The arboricity connection. *SIAM J. Discrete Math.*, 33(4):2267–2285, 2019. doi:10.1137/17M1159014.

26 Talya Eden, Dana Ron, and C. Seshadhri. Faster sublinear approximation of the number of k-cliques in low-arboricity graphs. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1467–1478, 2020. doi:10.1137/1.9781611975994.89.

27 Talya Eden, Dana Ron, and C. Seshadhri. On approximating the number of k-cliques in sublinear time. *SIAM Journal on Computing*, 49(4):747–771, 2020. doi:10.1137/18M1176701.

28 Talya Eden and Will Rosenbaum. Lower bounds for approximating graph parameters via communication complexity. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques 2018*, pages 11:1–11:18, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.11.

29 Talya Eden and Will Rosenbaum. On sampling edges almost uniformly. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, pages 7:1–7:9, 2018. doi:10.4230/OASIcs.SOSA.2018.7.

30 David Eppstein and Darren Strash. Listing all maximal cliques in large sparse real-world graphs. In *International Symposium on Experimental Algorithms*, pages 364–375. Springer, 2011.

31 Uriel 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.

32 Hendrik Fichtenberger, Mingze Gao, and Pan Peng. Sampling arbitrary subgraphs exactly uniformly in sublinear time. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, pages 45:1–45:13, 2020. doi:10.4230/LIPIcs.ICALP.2020.45.

33 Irene Finocchi, Marco Finocchi, and Emanuele G Fusco. Clique counting in mapreduce: Algorithms and experiments. *Journal of Experimental Algorithmics (JEA)*, 20:1–20, 2015.

34 Gaurav Goel and Jens Gustedt. Bounded arboricity to determine the local structure of sparse graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 159–167. Springer, 2006.

35 Oded Goldreich and Dana Ron. Approximating average parameters of graphs. *Random Structures & Algorithms*, 32(4):473–493, 2008. doi:10.1002/rsa.20203.

36 Mira Gonen, Dana Ron, and Yuval Shavitt. Counting stars and other small subgraphs in sublinear-time. *SIAM Journal on Discrete Mathematics*, 25(3):1365–1411, 2011.

37 Shweta Jain and C Seshadhri. A fast and provable method for estimating clique counts using turán's theorem. In *Proceedings of the 26th international conference on world wide web*, pages 441–449, 2017.

38 Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 2004.

39 Idit Kosti, Predrag Radivojac, and Yael Mandel-Gutfreund. An integrated regulatory network reveals pervasive cross-regulation among transcription and splicing factors. *PLoS computational biology*, 8(7):e1002603, 2012.

40 Andrew McGregor, David Tench, Sofya Vorotnikova, and Hoa T. Vu. Densest subgraph in dynamic graph streams. In Giuseppe F. Italiano, Giovanni Pighizzini, and Donald Sannella, editors, *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part II*, volume 9235 of *Lecture Notes in Computer Science*, pages 472–482. Springer, 2015. doi:10.1007/978-3-662-48054-0_39.

41 Michael Mitzenmacher, Jakub Pachocki, Richard Peng, Charalampos Tsourakakis, and Shen Chen Xu. Scalable large near-clique detection in large-scale networks via sampling. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 815–824, 2015.

42 Tanay Kumar Saha and Mohammad Al Hasan. Finding network motifs using mcmc sampling. In *Complex Networks VI*, pages 13–24. Springer, 2015.

43 Kijung Shin, Tina Eliassi-Rad, and Christos Faloutsos. Patterns and anomalies in k-cores of real-world graphs with applications. *Knowledge and Information Systems*, 54(3):677–710, 2018.

44 Jakub Tetek. Approximate triangle counting via sampling and fast matrix multiplication. *CoRR*, abs/2104.08501, 2021. To appear in ICALP 2022. arXiv:2104.08501.

45 Jakub Tětek and Mikkel Thorup. Edge sampling and graph parameter estimation via vertex neighborhood accesses. *CoRR*, 2021. To appear in STOC 2022. arXiv:2107.03821.

46 Virginia Vassilevska. Efficient algorithms for clique problems. *Information Processing Letters*, 109(4):254–257, 2009.

47 Pinghui Wang, John CS Lui, Bruno Ribeiro, Don Towsley, Junzhou Zhao, and Xiaohong Guan. Efficiently estimating motif statistics of large networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 9(2):1–27, 2014.

A Related Work

We note that some of the works were mentioned before, but we repeat them here for the sake of completeness. In recent years, there has been an increasing interest in the questions of subgraph approximate counting and uniform sampling in sublinear-time. These works differ by the query model, graph class of G and the subgraph H at question.

The general graph query model. The first works on estimating the number of subgraph counts were by Feige [31] and Goldreich and Ron [35], who presented algorithms for approximately counting the number of k-cliques in a graph for $k = 2$ (edges).[13] Later, Gonen, Ron and Shavitt [36] gave essentially optimal bounds for the problem of approximately counting the number of stars in a graph. In [20, 27], Eden, Levi, Ron and Seshadhri and Eden, Ron and Seshadhri presented essentially optimal query complexity bounds for the problems of approximately counting triangles and k-cliques. In [44], Tětek gave improved running time bounds for the regime where the query complexity is linear in previous works.

In [28], Eden and Rosenbaum presented a framework for proving subgraph counting lower bounds using reduction from communication complexity, which allowed them to reprove the lower bounds for all of the variants listed above.

Augmented model. In [2], Aliakbarpour, Biswas, Gouleakis, Peebles, and Rubinfeld and Yodpinyanee suggested a model that also allows for uniform edge samples. In that model they presented improved bounds for the approximate star counting problem. In that model, Assadi, Kapralov and Khanna [4] considered the problem of approximate counting of arbitrary subgraphs H. The expected query complexity of their algorithm is $\tilde{O}\left(\min\left\{m, \frac{m^{\rho(H)}}{n_H}\right\}\right)$, where $\rho(H)$ is the fractional edge cover of H,[14] and n_H is the number of copies of H in G. In particular, for the case of k-clique (and odd-cycle counting) the complexity of their algorithm is $O(m^{k/2})$, and this is optimal. In [10], Biswas, Eden and Rubinfeld have refined the complexity of approximating and uniformly sampling arbitrary motifs to $O^*\left(\min\left\{m, \mathsf{decomp\text{-}cost}(G, H, D^*(H))\right\}\right)$, where $D^*(H)$ is an optimal decomposition of H, and $\mathsf{decomp\text{-}cost}$ is the decomposition cost of H in G.

Set query model. In [5], Beame, Har-Peled, Ramamoorthy and Sinha suggested two new models that allow what they refer to as *independent set* (IS) and *bipartite independent set* (BIS) queries. They considered the problem of estimating the number of edges and gave $O^*(n^{2/3})$ and $O^*(1)$ algorithms for this problem using IS and BIS queries, respectively. The first result was later improved by Chen, Levi and Waingarten [15] who settled the complexity of the problem to $\Theta^*(n/\sqrt{m})$. In [6], Bhattacharya, Bishnu, Ghosh, and Mishra later have generalized the BIS model to tripartite set queries, where they considered the problem of triangle counting, and in [8], Bishnu, Ghosh, and Mishra settled the complexity of approximately counting triangles using BIS queries to $\Theta^*\left(\min\left\{\frac{m}{\sqrt{T}}, \frac{m^{3/2}}{T}\right\}\right)$, where T denotes the number of triangles.

[13] Feige considered a model that only allows for degree queries, and presented a factor 2 approximation algorithm, and also proved that with no additional queries this approximation factor cannot be improved in sublinear time. Goldreich and Ron then considered this question allowing also for neighbor queries. In this model the proved an $(1 \pm \varepsilon)$-factor approximation algorithm with the same complexity as the previous one (as well as a matching lower bound).

[14] The fractional edge cover of a graph $H = (V_H, E_H)$ is a mapping $\psi : E_H \to [0, 1]$ such that for each vertex $a \in V_H$, $\sum_{e \in E_H, a \in e} \psi(e) \geq 1$. The fractional edge-cover number $\rho(H)$ of H is the minimum value of $\sum_{e \in E_H} \psi(e)$ among all fractional edge covers ψ.

Uniform sampling. In [29], Eden and Rosenbaum initiated the study of sampling subgraphs (almost) uniformly at random. They considered the general graph query model, and presented upper and matching lower bounds for the problem of sampling edges almost uniformly. Their algorithm matches the complexity of the counting variant of the problem. Their algorithm's dependency on ε was later improved by Tětek and Thorup [45] to $\log(1/\varepsilon)$, so that for all practical purposes the new algorithm allows to sample from essentially the uniform distribution. They also present an algorithm that works in a stronger model that allows for hash-based neighbor queries, and outputs an edge from *exactly* the uniform distribution.

In [32], Fichtenberger, Gao and Peng proved that in the augmented edge model, *exact* uniform sampling of arbitrary subgraphs can be performed in $O\left(\frac{m^{\rho(H)}}{n_H}\right)$ time. This matches the upper bound of [4] for the counting variant. The aforementioned bound of [10], also holds for this setting, refining over the complexity of [32].

In [22], Eden, Mossel and Rubinfeld presented an algorithm for sampling multiple edges efficiently. Their algorithm was later shown to be optimal by Tětek and Thorup [45].

Graphs G with bounded arboricity. In [25, 26], Eden, Ron and Seshadhri first studied the problem of sublinear approximate counting in bounded arboricity graphs. They presented improved algorithm for edges, star and k-clique counting in the general graph model, parameterized by the arboricity. In [23], Eden Ron and Rosenbaum presented an improved algorithm for almost uniform sampling of edges in bounded arboricity graphs, in the general graph query model.

Approximating the arboricity in sublinear time. In [21], Eden, Mossel and Ron presented an algorithm for approximating the arboricity in $\tilde{O}(n/\alpha)$ time. Their algorithm returns a value $\hat{\alpha}$ such that, with high probability, $\hat{\alpha} \in [\alpha, \alpha \cdot c \log^2 n]$, for some constant c. It can also be shown that the algorithm of McGregor, Tench, Vorotnikova and Vu [40] can be adapted to the adjacency list query model,[15] resulting in a $(1 \pm \varepsilon)$-multiplicative approximation in $\tilde{O}(m/\alpha)$ complexity. The output of these algorithms can be used as input to our algorithm (as well as all aforementioned sublinear-time algorithms that rely on getting an upper bound on the arboricity as input).

[15] This requires some care, as their algorithm relies on sampling edges uniformly at random, which is not immediately implementable in the adjacency list model. However, it can be shown that only edges with high degree endpoints (roughly ones with degree above α) are of interest, and these can be sampled uniformly with an additive overhead of $\tilde{O}(m/\alpha)$.