Almost Optimal Bounds for Sublinear-Time Sampling of $k$-Cliquies: Sampling Cliques is Harder Than Counting

Talya Eden* Dana Ron† Will Rosenbaum‡

December 9, 2020

Abstract

Motivated by the need to analyze massive datasets efficiently, recent theoretical work has examined the problems of counting and sampling small subgraphs from graphs in sublinear time. 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 $\epsilon$ is a given approximation parameter. To this end it may perform the following types of queries on vertices: degree queries, neighbor queries, and pair queries.

We prove that the query complexity of this problem is $\Theta^*(\max\left\{\left(\frac{(n\alpha)^{k/2}}{n_k}\right)^{\frac{1}{k-1}}, \min\left\{n\alpha, \frac{n\alpha^{k-1}}{n_k}\right\}\right\})$. Interestingly, this establishes a separation between approximate counting and approximate uniform sampling in the sublinear regime. For example, if $k = 3$, $\alpha = O(1)$, and $n_3$ (the number of triangles) is $\Theta(n)$, then we get a lower bound of $\Omega(n^{1/4})$ (for constant $\epsilon$), while under these conditions, a $(1 \pm \epsilon)$-approximation of $n_3$ can be obtained by performing $\text{poly}(\log(n/\epsilon))$ queries (Eden, Ron and Seshadhri, SODA20).

Our lower bound follows from a construction of a family of graphs with arboricity $\alpha$ such that in each graph there are $n_k$ cliques (of size $k$), where one of these cliques is “hidden” and hence hard to sample. 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 $H_k$, where the challenge is simulate queries to $H_k$ while being given access only to $G$.

---

*CSAIL, MIT, Cambridge, MA, USA
†School of Electrical Engineering, Tel Aviv University, Tel Aviv, Israel
‡Amherst College, Amherst, MA, USA
1 Introduction

Counting and sampling are fundamental computational tasks in randomized algorithms, statistics, data science, and many other disciplines. Given a family $F$ of combinatorial objects—for example, $k$-cliques in a given graph, or satisfying assignments of a Boolean formula—the approximate counting problem asks to compute a number that is a $(1 \pm \varepsilon)$-multiplicative estimate of $N = |F|$. The almost uniform sampling problem is to produce a sample from $F$ such that each $x \in F$ is chosen with probability in the range $(1 \pm \varepsilon)/N$.

In a seminal work Jerrum, Valiant, and Vazirani [JVV86] showed that for a large family of combinatorial problems—self-reducible problems [Sch81]—approximate counting and almost uniform sampling are equivalent under polynomial-time reductions. In [DLM20], Dell, Lapinskas, and Meeks proved related results in the fine-grained setting.

Recently, Fichtenberger, Gao, and Peng [FGP20] asked if results analogous to [JVV86] hold for sublinear-time algorithms: In the sublinear-time regime, is almost uniform sampling ‘computationally comparable’ to approximate counting?

In [FGP20], the authors provide evidence for a positive answer to this question. Building upon previous results of Assadi, Kapralov, and Khanna [AKK18], Fichtenberger et al. describe algorithms for approximately counting and sampling arbitrary subgraphs of a graph $G$. In the case of counting and sampling $k$-cliques, both algorithms have expected run-time $O^*(m^{k/2}/n_k)$, where $m$ and $n_k$ denote the number of edges and $k$-cliques in $G$, respectively. These upper bounds are essentially optimal by a nearly-matching lower bound due to Eden and Rosenbaum [ER18a].

The algorithms of [FGP20] and [AKK18], however, require a non-standard “augmented” query model that allows the algorithm to sample a uniformly random edge from the graph as an atomic unit-cost query. This model is strictly stronger than the well-studied general graph model, which allows for only degree, neighbor, and pair queries. Indeed, Eden and Rosenbaum [ER18a] prove a lower bound of $\Omega \left( n/n_k^{1/k} + m^{k/2}/n_k \right)$ for the query complexity of approximately counting $k$-cliques in the general graph model, which is strictly greater than the upper bounds of [AKK18] and [FGP20] for some range of parameters. Thus, it is not clear that counting and sampling should have the same query complexities in the general graph model.

Another subtlety not addressed in [ERS20b, AKK18, FGP20] is that the complexity of approximate counting can vary dramatically for restricted families of input graphs. The algorithms of [ERS20b, AKK18, FGP20] are only optimal when considering the worst-case over all possible inputs. However for a rich family of graphs—namely the family of graphs with bounded arboricity—the lower bounds can be circumvented: In [ERS20a], Eden, Ron and Seshadhri prove that when given a bound $\alpha$ on the arboricity of the input graph $G$, the value of $n_k$ can be approximated up to a factor of 2, it is equivalent to the average degree of the densest subgraph in $G$.  

---

1 We refer to this notion of “almost uniform” sample as pointwise almost uniformity. Pointwise uniformity is the notion used, for example, in [JVV86], and is a strictly stronger requirement than approximate uniformity with respect to $L_2$ or total variation distance (TVD).

2 Namely, they prove “black box” results for turning algorithms which decide whether or not a witness exists into algorithms to approximately count the number of witnesses, or to sample from the set of witnesses approximately uniformly, with essentially the same running time.

3 We use $\Theta^*$ to suppress a dependence on functions $g(\log n, k, 1/\varepsilon)$, which are at most $(\log n/\varepsilon)^{O(k)}$.

4 We note that the upper bound for approximate counting of $k$-cliques was already known due to [ERS20b].

5 The arboricity of a graph $G$, denoted $\alpha(G)$, is the minimal number of forests required to cover its edge set. It is well known, that up to a factor of 2, it is equivalent to the average degree of the densest subgraph in $G$.  

---

1
in time

\[ O^* \left( \min \left\{ \frac{n}{n_k^{1/k}}, \frac{n \alpha^{k-1}}{n_k} \right\} + \frac{m \alpha^{k-2}}{n_k} \right) = O^* \left( \frac{n \alpha^{k-1}}{n_k} \right). \]  

(1)

For certain ranges of the parameters, this upper bound is exponentially smaller than the worst-case lower for general graphs. A natural question is whether a similar result can be established for sampling \(k\)-cliques almost uniformly in bounded arboricity graphs.

In [ERR19], Eden, Rosenbaum and Ron provide a positive answer to this question in the case of sampling edges (i.e., \(2\)-cliques) in a graph. Specifically, they prove that the query complexity of uniformly edge is \(\Theta^* (n \alpha/m) = \Theta^* (n \alpha/n^2)\). This complexity matches the complexity of (1) for \(k = 2\), thereby providing a positive answer to Fichtenberger et al.’s question in the case of counting and sampling edges. Moreover, the tight correspondence between the complexities of counting and sampling edges holds even when parameterized by the graph’s arboricity.

In the current work, we show that, surprisingly, the tight correspondence for the complexity of sampling and counting \(k\)-cliques does not generalize to \(k \geq 3\). Specifically, we prove the following theorem.

**Theorem 1.** Any almost uniform \(k\)-cliques sampling algorithm for graphs with arboricity at most \(\alpha\) requires

\[ \Omega \left( \max \left\{ \left( \frac{(n \alpha)^{k/2}}{k \cdot n_k} \right)^{\frac{1}{k}} , \min \left\{ n \alpha, \frac{n \alpha^{k-1}}{n_k} \right\} \right\} \right) \]

queries.

The second term in the lower bound follows directly from the lower bound of [ERS20a] for the approximate counting variant of the problem. The first term, however, might be significantly larger. For example, for the case of triangles \((k = 3)\), \(\alpha = O(1)\), and \(n_3 = \Theta(n)\), the first term translates to a lower bound of \(\Omega(n^{1/4})\) for approximately uniform sampling. This is in stark contrast to the counting variant which has complexity \(O^*(1)\), implying an exponential gap between the two tasks for certain regimes of parameters.

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

**Theorem 2.** There exists an almost uniform sampling algorithm for \(k\)-cliques in graphs with arboricity at most \(\alpha\). Given a constant factor estimate of \(n_k\), the query complexity of the algorithm is

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

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 [ERS20a] whose expected query complexity (1) is dominated by the runtime of Theorem 2.

\(^6\)Recall that for every graph \(G\) with arboricity at most \(\alpha\), \(m\) is always upper bounded by \(\alpha\).
Remarks on almost uniformity. In the results listed above, we measure “almost uniformity” with respect to pointwise distance between distributions. That is, we require that every $k$-clique is sampled with probability $(1 \pm \varepsilon)/n_k$. One could also consider the (strictly weaker) requirement that the distribution of sampled $k$-cliques is close to uniform with respect to total variation distance (TVD). That is, if $p_C$ is the probability that the algorithm returns a clique $C$, then $\sum_C |p_C - 1/n_k| \leq \varepsilon$. Interestingly, the complexity of sampling $k$-cliques almost uniformly with respect to TVD is different from the bounds prescribed in Theorems 1 and 2. The algorithm of Eden, Ron, and Seshadhri [ERS20a] can be adapted to sample a $k$-clique almost uniformly with respect to TVD using the same number of queries (stated in Equation (1)) as their approximate counting algorithm. In particular, combining this observation with Theorem 1 gives an exponential separation between the complexities of sampling $k$-cliques (1) pointwise almost uniformly, and (2) almost uniformly with respect to TVD, when $k \geq 3$. In contrast, for the case of edges ($k = 2$), TVD and pointwise almost uniform sampling can both have sample complexity $O^*(1)$ in bounded arboricity graphs [ERR19].

Sampling almost uniformly with respect to TVD may be sufficient in many contexts. However, there are scenarios in which we the stronger notion of pointwise almost uniform sampling is crucial, since we cannot allow to “give up” on sampling a small fraction of the domain elements, as might be the case when sampling with respect to TVD. See [ER18b] for further discussion.

We review additional related work in Section 5, and next turn to discuss the ideas behind our upper and lower bounds.

1.1 The high level ideas behind the clique-sampling algorithm

We start by briefly describing the ideas behind the edge sampling algorithm of [ERR19] for sampling edges almost uniformly, which we employ both as a subroutine and starting point for our sampling scheme for $k$-cliques. Throughout, we assume that an upper bound $\alpha$ on the arboricity of the input graph is known.

1.1.1 The edge sampling algorithm

Let $L_0$ be the set of all vertices in the graph with degree at most (roughly) $\alpha$ (so that almost all the vertices in the graph belong to $L_0$). The 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 $j$ 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 $\approx 1/\alpha^\alpha$.

1.1.2 A random walk on $(k - 1)$-cliques

In order to sample $k$-cliques in $G$, we present a variant of the edge-sampling algorithm described in Section 1.1.1, and apply it to an auxiliary graph $H_k$, defined as follows. 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 an edge in $H_k$ between...
a pair of nodes $v_Q, v_Q'$ corresponding to two of its $(k-1)$-cliques, $Q$ and $Q'$. We say that $C$ is assigned to $Q$ and $Q'$, where the assignment rule depends on the degrees (in $G$) of the vertices in $C$. In particular, when $k = 2$, this assignment is uniquely determined and we have $H_2 = G$. 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 $G$. A simple but important observation is that if $G$ has arboricity at most $\alpha$, then so does $H_k$.

The challenge is to simulate the edge sampling procedure of [ERR19] on the graph $H_k$, while only having query access to the graph $G$. The simulation is not straightforward since:

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 [ERR19]); and
3. in order to sample a random neighbor of a vertex $v_Q$ in $H_k$ we must sample a $k$-clique in $G$ that is assigned to $Q$. (In [ERR19] this could be achieved by a single neighbor query.)

Below, we outline how we address these challenges.

**Addressing the challenges.** In order to sample nodes in $H_k$ we recursively invoke our procedure for sampling $(k-1)$-cliques in $G$ almost uniformly. Given a sampled node $v_Q$ in $H_k$, we implement a procedure to check whether $v_Q \in L_0$, by trying to approximate the number of $k$-cliques that are assigned to $Q$ in $G$. To do so efficiently, we replace the threshold $\alpha$ used to define $L_0$ in [ERR19], by a value $\tau \geq \alpha$, where we will explain how $\tau$ is chosen later in the analysis.

It remains to explain how we simulate a random neighbor query for a vertex $v_Q$ in $H_k$ (so as to simulate a random walk on $H_k$). Let $A(Q)$ denote the set of $k$-cliques assigned to $Q$. Recall that sampling an edge incident to $v_Q$ translates to sampling a $k$-clique $C$ in $A(Q)$. The rule for assigning each $k$-clique to a pair of $(k-1)$-(sub)cliques is defined in such a way that we need only consider $k$-cliques containing higher-degree neighbors of the minimum degree vertex $u$ in $V_Q$. Let $d(Q) = d(u)$, where $d(u)$ denotes the degree of $u$ in $G$. If $d(Q) \leq \sqrt{n\alpha}$, then by performing a uniformly random neighbor query from $u$ in $G$, each such neighbor is sampled with probability $1/d(Q)$. Otherwise if $d(Q) > \sqrt{n\alpha}$, then for every higher degree neighbor $w$ of $u$, $d(w) > \sqrt{n\alpha}$. By invoking the edge-sampling procedure of [ERR19] on $G$, we can sample every vertex $y$ in $G$ with probability $d(w)/n\alpha$. By performing rejection sampling, we can get each such higher degree vertex $w$ with probability $d(w)/n\alpha \cdot \sqrt{n\alpha}/d(w) = \sqrt{n\alpha}/d(Q)$. Hence, for every $(k-1)$-clique $Q$, each $k$-clique in $A(Q)$ is sampled with probability max $\left\{1/d(Q), \frac{1}{\sqrt{n\alpha}} \right\} = \frac{1}{\min\{d(Q), \sqrt{n\alpha}\}}$.

To simulate a random neighbor query from a node $v_Q$ in $H_k$ such that $v_Q \notin L_0$ (so that the number of $k$-cliques in $A(Q)$ is at least $\tau$) we repeat the above sampling attempts $O^\tau\left(\frac{\min\{d(Q), \sqrt{n\alpha}\}}{\tau}\right)$ times. This process succeeds in obtaining a uniformly distributed $k$-clique in $A(Q)$ with high probability. For a node $v_Q$ in $L_0$, performing $O^\tau\left(\frac{\min\{d(Q), \sqrt{n\alpha}\}}{\tau}\right)$ many attempts implies that each $k$-clique in $A(Q)$ is obtained with probability $1/\tau$.

An inductive analysis shows that a single invocation of the above simulation of the random walk on $H_k$ returns each $k$-clique in $G$ with probability roughly $\frac{1}{\min\{d(Q), \sqrt{n\alpha}\}}$. 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 $\tau^{k-2}$ 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 $\frac{n_k}{n\alpha \tau^{k-2}}$. Hence, $O^\tau\left(\frac{n\alpha \tau^{k-2}}{n_k}\right)$ repetitions are sufficient so that, with high probability,
an almost uniformly distributed $k$-clique in $G$ is returned.

**Query complexity.** As discussed above, to sample a $k$-clique in $G$ with high probability, we perform $t = O^*(\frac{n^{\alpha + 2k - 2}}{n_k})$ repetitions of the random walk simulation on $H_k$. In each such simulation, 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 simulated on a node $v_T$ in $H_i$ for $i > 2$, $r = O^*(\frac{\min(d(T), \sqrt{n\alpha})}{\tau})$ queries are performed in $G$. Conditioned on $\tau$ being sufficiently larger than $\alpha$, we get that the expected number of queries in each such simulation is just $O^*(1)$ (while the maximum is $O^*(\frac{\sqrt{n\alpha}}{\tau})$). This implies that the expected total query complexity is $O^*(\frac{n^{\alpha + 2k - 2}}{n_k})$. As for the maximum running time, we can get an upper bound of $O^*(\frac{n^{\alpha + 2k - 2}}{n_k} + \frac{\sqrt{n\alpha}}{\tau})$ 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 $O^*(\frac{n^{\alpha + k - 1}}{n_k})$, as in the case of counting, while the maximum query complexity is $O^*(\frac{n^{\alpha + k - 1}}{n_k} + \sqrt{n/\alpha})$. The upper bound in Theorem 2 is derived by setting $\tau$ so that the two summands in the expression $O^*(\frac{n^{\alpha + 2k - 2}}{n_k} + \frac{\sqrt{n\alpha}}{\tau})$ are equal.

### 1.2 Overview of the lower bound

The second term in the lower bound of Theorem 1 follows directly from a lower bound of $\min \left\{ na, \frac{n(\alpha/k)^{k-1}}{n_k} \right\}$ by [ERS20a] for the counting variant. Hence, our main focus is on proving the first term (which, as noted previously, may be much larger than the second).

To obtain the first term in lower bound, we construct a family of graphs (with arboricity at most $\alpha$), 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 $\frac{(n\alpha)^{k/2}}{n^{k/2}} \frac{1}{\tau^c}$ (for a sufficiently large constant $c$) cannot sample this clique with probability $\Omega(1/n_k)$. This does not preclude the possibility that the expected complexity of the algorithm is smaller (as discussed in Subsection 1.1.2).

This 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. Other than the clique edges, there are no other edges between the high-degree vertices. 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.\(^7\) We prove that in each

---

\(^7\)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 are incident to all clique vertices), but this is not advantageous for the algorithm.
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 \(O(k/\ell)\).\(^8\)

The rough intuition for the upper bound \(O(k^2/\ell^2)\) on revealing the first edge is that the number of clique edges is \(\binom{\ell}{2}\), 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 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 \(T \cdot \frac{k^2}{\ell^2} \cdot (T \cdot \frac{k}{\ell})^{k-2}\). If we want this expression to be \(\Omega(1/n_k)\), the number of queries \(T\) must be \(\Omega\left(\left(\frac{(n\alpha)^{k/2}}{k^{k/2} n_k^{k/2}}\right)^{1/(k-1)}\right)\).

2 Preliminaries

Let \(G = (V, E)\) be a graph over \(n\) vertices and arboricity at most \(\alpha\). Each vertex \(v \in V\) has a unique id in \([n]\), denoted \(id(v)\). Let \(C_k\) denote the set of \(k\)-cliques of \(G\), and let \(n_k = |C_k|\). 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, \(\text{deg}(v)\), returns the degree \(d(v)\) of the vertex \(v\); (2) A neighbor queries, \(\text{nbr}(v, i)\) for \(i \in [d(v)]\), returns the \(i\)th neighbor of \(v\); (3) A pair query, \(\text{pair}(v, v')\), 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 (Clique’s degree). For any \(k\)-clique \(C\), we let \(d(C)\) denote the degree of the first vertex in \(C\) according to the ordering \(\prec\). We refer to it as the degree of \(C\). Letting \(v\) denote the first vertex in \(C\), we refer to \(\Gamma(v)\) as the set of neighbors of \(C\), and denote it by \(\Gamma(C)\).

Definition 2.3 (Clique’s id and an ordering of cliques). For a \(t\) 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 \(C, C'\), \(C < C'\) 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 \(C\) its two first \((k-1)\)-cliques according to \(\prec\). For every \((k-1)\) clique \(Q\), we denote its set of assigned \(k\)-cliques by \(A(Q)\), and let \(a(Q) = |A(Q)|\). We refer to \(a(Q)\) as the assigned cliques degree of \(Q\).

\(^8\)We note that whenever the term \(\frac{(n\alpha)^{k/2(k-1)}}{k^{k/2} n_k^{k/2}}\) dominates the second term in the lower bound of Theorem 1, it is smaller than \(\sqrt{n/\alpha}\).
1.1 A

3.3 We present the edge-sampling procedure. This procedure

\[ n \]

\[ n \]

\[ n \]

\[ n \]

\[ n \]

\[ n \]

\[ 2 \]

2.4

3.2

3.1

\[ d \]

\[ C \]

\[ \{ \]

\[ Q, u \]

\[ Q \cup \{ w \} \]

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

3 The algorithm for sampling \( k \)-cliques

As discuss in the introduction (Section 1.1), our algorithm for sampling \( k \)-cliques almost uniformly in a graph \( G \) works by simulating an edge-sampling procedure on an auxiliary graph, \( H_k \), which is defined based on \( G \). We begin by precisely defining \( H_k \), providing the clique-sampling algorithm and stating our main theorem. In Section 3.1 we present the edge-sampling procedure. This procedure is designed to run on a graph \( F \), which is defined based on a graph \( G \), where query access to \( F \) is implemented by subroutines that have query access to \( G \). In our case the edge-sampling procedure is applied to \( F = H_k \), and as one of the subroutines requires access to almost uniformly distributed cliques of smaller size, the procedure is also recursively applied to \( F = H_i \) for \( i = k - 1, \ldots, 2 \).

We give sufficient conditions on these subroutines based on which the correctness of the edge-sampling procedure can be established. We then turn (in Section 3.2) to present and analyze these subroutines for the \( H_i \)’s. We wrap things up with an inductive argument in Section 3.3.

Definition 3.1 (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 \( H_k = G \), and each edge (2-clique) in \( G \), is assigned to both its endpoints. More generally, the above definition implies a one-to-one correspondence between the set edges incident to a node \( v_Q \) in \( H_k \) and the set \( A(Q) \) of \( k \)-cliques assigned to \( Q \) in \( G \). Hence, sampling an edge \( e \in H_k \) is equivalent to sampling a \( k \)-clique in \( G \). In the rest of the paper we go back and forth between sampling edges in \( H_k \) and sampling \( k \)-cliques in \( G \).

We claim that for every \( k \), the graph \( H_k(G) \) has arboricity at most \( \alpha(G) \). The proof is deferred to Appendix A.

Claim 3.2. Let \( G \) be a graph of arboricity at most \( \alpha \). Then \( H_k(G) \) has arboricity at most \( \alpha \).

In addition to receiving as input \( n, \alpha, k \) and \( \epsilon \) (as well as being given query access to \( G \)), our algorithm receives two additional parameters. The first, \( \overline{n}_k \), is assumed to be a constant factor estimate of \( n_k \). Such an estimate can be obtained by running the algorithm of [ERS20a] without asymptotically increasing the expected complexity of our sampling algorithm. The second, \( \tau \), is a parameter that affects the complexity of our algorithm. In particular, as we show in Theorem 3, it introduces a certain tradeoff between the expected running time of the algorithm and the maximum running time. We then show how for an appropriate setting of \( \tau \) and by cutting-off the execution of the algorithm, we can obtain Theorem 2.

Theorem 3. Consider an invocation of the algorithm Sample-a-Clique\( (G, n, \alpha, k, \epsilon, \overline{n}_k, \tau) \) with an estimate \( \overline{n}_k \) of \( n_k \) such that \( \overline{n}_k \in [n_k, 2n_k] \), and with a parameter \( \tau \) such that \( \frac{\alpha}{\epsilon} \cdot (4k \log n)^k \leq \tau \leq \sqrt{\overline{n}_k \alpha} \). The algorithm Sample-a-Clique returns a \( k \)-clique \( C \) in \( C_k \), such that the resulting
Sample-a-Clique($G, n, \alpha, k, \varepsilon, \mathbf{\pi}_k, \tau$)

1. Let $\beta = \varepsilon/10k$.
2. Set $N_k = n\alpha^{k-1}$.
3. While the number of queries does not exceed $r = \max\left\{\sqrt{n\alpha}, \min\left\{n\alpha, \frac{n\alpha^{k-2}}{n_k}\right\}\right\} \cdot (k \log n/\varepsilon)^c$ for a sufficiently large constant $c$:
   (a) Invoke Sample-Edge($H_k, N_k, \beta, \tau, \mathbf{\bar{p}} = (G, n, k, \alpha)$), and if an edge in $H_k$ is returned, then return the corresponding $k$-clique in $G$.

distribution on $k$-cliques is pointwise $\varepsilon$-close to uniform on $C_k$. The expected query complexity of the algorithm is $O^*\left(\min\left\{n\alpha, \frac{n\alpha^{k-2}}{n_k}\right\}\right)$, and the maximum query complexity of the algorithm is $O^*\left(\max\left\{\sqrt{n\alpha}, \min\left\{n\alpha, \frac{n\alpha^{k-2}}{n_k}\right\}\right\}\right)$.

Theorem 2 is a corollary of the above theorem, when setting $\tau = \max\left\{\frac{\alpha}{\varepsilon} \cdot (4k \log n)^k, (\mathbf{\bar{p}}/\sqrt{n\alpha})^{1/(k-1)}\right\}$, for a given good estimate $\mathbf{\pi}_k$ of $n_k$.

Corollary 3.3 (Theorem 2, restated). There exists a pointwise $\varepsilon$-close to uniform sampling algorithm for $k$-cliques in graphs with arboricity at most $\alpha$. If the algorithm is given an estimate $\mathbf{\pi}_k$ of $n_k$ such that $\mathbf{\pi}_k \in [n_k, 2n_k]$, then the query complexity of the algorithm is

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

If no such estimate is given, then the above only holds in expectation.

Note that for a given estimate $\mathbf{\pi}_k$, we set $\tau = \max\left\{\frac{\alpha}{\varepsilon} \cdot (4k \log n)^k, (\mathbf{\bar{p}}/\sqrt{n\alpha})^{1/(k-1)}\right\}$, while Theorem 3 holds only for values $\tau$ such that $\frac{\alpha}{\varepsilon} \cdot (4k \log n)^k \leq \tau \leq \sqrt{n\alpha}$. This setting implies that $\tau$ is always lower bounded by $\frac{\alpha}{\varepsilon} \cdot (4k \log n)^k$, but is only upper bounded by $\sqrt{n\alpha}$ if $\alpha \leq \frac{\varepsilon\sqrt{n\alpha}}{(4k \log n)^k}$. If this condition does not hold, then $\alpha = \Theta(n)$ (recall that always $\alpha \leq n$), so it is more beneficial to invoke the $m^{k/2}/n_k$ algorithm of [FGP20], replacing each edge query by an invocation of the edge sampling algorithm of [ERR19].

We defer the proof of the theorem and corollary to Subsection 3.3, and first present and analyze the procedure Sample-Edge and the subroutines it uses.

### 3.1 The procedure Sample-Edge

In this subsection we present the procedure Sample-Edge for sampling edges almost uniformly in a graph $F$, given query access to a graph $G$ that defines $F$. In the case that $F = G$, we simply invoke the procedure referred to in the following theorem.

**Theorem 4** (Corollary 2.8 in [ERR19]). Let $G$ be a graph over $n$ vertices. There exists a procedure Sample-Basic-Edge that, given query access to $G$, a bound $\alpha$ on the arboricity of $G$ and a parameter $\beta \in (0, 1)$, returns each oriented edge in $G$ with probability $\frac{1+\beta}{n\alpha}\gamma$ for $\gamma = 4\log^2 n/\beta$ (and fails to return an edge with the remaining probability). The query complexity and running time of the algorithm are $O(\log n)$. 


For $F \neq G$, the procedure Sample-Edge makes calls to three subroutines: Define-$L_0$, Sample-$E(L_0)$ and Sample-a-Neighbor. These subroutines simulate queries to $F$ by performing queries to $G$. In particular, Define-$L_0$ returns whether a given node in $F$ belongs to a certain set of nodes $L_0$ and effectively determines $L_0$ by its answers. The subroutine Sample-$E(L_0)$ returns a random edge in $F$ that is incident to $L_0$ and, given a node $v$ in $F$ that does not belong to $L_0$, Sample-a-Neighbor returns a random neighbor of $v$. Using these subroutines, Sample-Edge performs a random walk of length logarithmic in $|V_F|$ starting from $L_0$, and returns the last edge traversed in the walk.

**Sample-Edge**($F, \overline{N}_F, \beta, \tau, \vec{p} = (G, n, \alpha, k)$)

1. If $F = G$, then invoke Sample-Basic-Edge($G, n, \beta, \vec{p}$) and if an edge $(u, v)$ is returned, then return it. Otherwise, FAIL.
2. Set $s = \lceil \log \overline{N}_F \rceil$ and set $\beta' = \beta/(2s + 2)$.
3. Choose $j \in [0, \ldots, s]$ uniformly at random.
4. Invoke Sample-$E(L_0)(F, \overline{N}_F, \beta', \tau, \vec{p})$, and let $e_0 = (v_0, v_1)$ be the returned edge if one was returned. Otherwise, return FAIL.
5. For $i = 1$ to $j$ do:
   (a) If Define-$L_0(F, \overline{N}_F, v_i, \delta = \frac{\beta'}{\overline{N}_F} \cdot \frac{\beta}{(k \log(n/\beta))}^{O(k)}, \beta', \tau, \vec{p}) = \text{YES}$ then return FAIL.
   (b) Invoke Sample-a-Neighbor($F, v_i, \beta', \tau, \vec{p}$) to sample an edge $(v_i, v_{i+1})$ in $F$.
6. Return $(v_j, v_{j+1})$.

Before analyzing Sample-Edge, we briefly discuss its application to sampling $k$-cliques. As described in the algorithm Sample-a-Clique, in order to sample $k$-cliques in $G$, we invoke the procedure Sample-Edge on the graph $H_k$ and implement the subroutines it calls (for $F = H_k$). Our main lemma regarding the correctness and complexity of Sample-Edge when invoked with $H_k$ (and for appropriate implementations of the subroutines) follows.

**Lemma 3.4.** Consider an invocation of Sample-Edge($H_k, \overline{N}_k, \beta, \tau, \vec{p} = (G, n, k, \alpha)$) where $\overline{N}_k \geq |V_{H_k}|$, and $(\alpha/\beta) \cdot O(4k \log n)^k \leq \tau \leq \sqrt{n\alpha}$.

The procedure returns an edge in $H_k$ with probability

$$\Omega \left( \frac{n_k}{n\alpha \gamma \cdot \tau^{k-2}} \cdot \left(\frac{\beta}{\log(n/\beta)}\right)^{O(k)} \right),$$

and conditioned on an edge being returned, each edge is returned with probability $\frac{1+\beta}{n_k}$.

Furthermore, the expected running time of a single invocation of the procedure is $O^*(1)$, and the maximum running time is $O^*(\sqrt{n\alpha}/\tau)$.

The proof of Lemma 3.4 appears in Section 3.3. There, we also prove Theorem 3, which follows almost directly from the lemma.

We next introduce several definitions that formalize what it means for the subroutines called by Sample-Edge to be “good simulators” for queries to $F$.

**Definition 3.5** (A good $L_0$ and $L_0$ oracle). For a graph $F$ and a parameter $\tau$, we say that a subset of vertices in $F$ is a $\tau$-good $L_0$ with respect to $F$ if the following conditions hold: (1) for every
vertex \(v \in V_F\) such that \(d_F(v) \leq \tau\) we have \(v \in L_0\), and (2) for every \(v \in V_F\) such that \(d_F(v) > 2\tau\) we have \(v \notin L_0\).

We say that a subroutine is a \((\delta, \tau)\)-good \(L_0\) oracle with respect to a graph \(F\), if with probability at least \(1 - \delta\) the subroutine defines a \(\tau\)-good \(L_0\).

**Definition 3.6** (A good \(E(L_0)\)-sampling subroutine.). We say that a subroutine is a \((\beta, X)\)-good \(E(L_0)\)-sampling subroutine with respect to a graph \(F\) it returns every oriented edge in \(E_F(L_0)\) with probability in \(\frac{1 + \beta}{X}\).

**Definition 3.7** (A good neighbor-sampling subroutine.). We say that a subroutine is a \((\beta, \tau)\)-good neighbor-sampling subroutine with respect to a graph \(F\) if the following holds. Given a vertex \(v\) such that \(d_F(v) > \tau\), each oriented edge incident to \(v\) in \(F\) is returned with probability in \(\frac{1 + \beta}{d_F(v)}\).

In order to state and apply the next lemma, it will also be useful to introduce the following definition.

**Definition 3.8** (A good edge-sampling procedure). We say that a procedure is a \((\beta, X)\)-good edge-sampling procedure with respect to a graph \(F\) if it returns every oriented edge in \(F\) with probability in \(\frac{1 + \beta}{X}\).

Recall that \(s = \lceil \log N_F \rceil\) and \(\beta' = \beta / (2s + 2)\).

**Lemma 3.9.** Consider an invocation of the procedure Sample-Edge with parameters \((F, N_F, \beta, \tau, \vec{p})\)

Assume that the following conditions hold for some \(X\).

1. The subroutine Define-\(L_0(F, N_F, v, \delta, \beta', \tau, \vec{p})\) is a \((\delta, \tau)\)-good \(L_0\) oracle with respect to \(F\) for \(\delta \leq \beta / X\).
2. If \(L_0\) determined by Define-\(L_0\) is \(\tau\)-good with respect to \(F\), then Sample-\(E(L_0)(F, N_F, \beta', \tau, \vec{p})\) invoked in Step 4 is a \((\beta', X)\)-good \(E(L_0)\)-sampling subroutine for \(L_0\).
3. The subroutine Sample-a-Neighbor\((F, v, \beta', \tau, \vec{p})\) invoked in Step 5b is a \(\beta'\)-good neighbor-sampling subroutine for \(F\).

Then Sample-Edge is a \((\beta, X')\)-good edge-sampling procedure for \(X' = X(s + 1)\).

The proof of Lemma 3.9 is similar to the proof of the correctness of the edge-sampling procedure of [ERR19], with careful adaptations due to the simulated queries, and is deferred to Appendix A.

We shall bound the complexity of Sample-Edge in Section 3.3, after analyzing the subroutines.

### 3.2 Simulating queries in \(H_t\) for \(t \leq k\): implementing the subroutines

In this section we present the subroutines Define-\(L_0\), Sample-\(E(L_0)\) and Sample-a-Neighbor. The subroutines are stated for \(H_k\), but can be applied to any \(H_t\) for \(t \in [3, k]\).

We will rely on the following simple claims from [ERS20a].

**Claim 3.10** (Claim 3.1 in [ERS20a]). For every \(k \geq 2\), \(\sum_{C \in \mathcal{C}_h} d(C) \leq n \cdot \alpha(G)^k\).

**Claim 3.11** (Claim 3.2 in [ERS20a]). For every \(k \geq 2\), \(n_k(G) \leq \frac{2\alpha(G)}{k} \cdot n_{k-1}(G)\).

The next simple claim is useful for estimating the probability of the recurring sampling attempts in the different subroutines. The proof is deferred to Appendix A.
Claim 3.12. For \( x > 0, y \geq 1 \) and \( x \cdot \lfloor y \rfloor < 1 \), it holds that
\[
xy(1 - 2xy) \leq 1 - (1 - x)\lfloor y \rfloor \leq xy(1 + 1/y).
\]

We start with presenting and analyzing the procedure for sampling oriented edges incident to nodes in \( L_0(H_k) \) (where \( L_0 \) will be determined by the subroutine \( \text{Define-} L_0 \) as discussed subsequently). Observe that this is the subroutine in which the recursive calls are invoked. Indeed, to sample edges in \( E(L_0(H_k)) \), it first samples (almost uniformly distributed) nodes in \( H_k \), which is equivalent to sampling (almost uniformly distributed) \((k - 1)\)-clique in \( G \). Therefore, in this subsection we analyze the correctness conditioned on the correctness of the recursive call to \( \text{Sample-Edge} \), and in the following section we shall prove the complete inductive argument.

| Sample-\( E(L_0) \)(\( H_k, \beta', \tau, \vec{p} = (G, n, k, \alpha) \)) |
| --- |
| 1. Let \( N_{k-1} = n\alpha^{k-3} \). |
| 2. Let \( \beta = \beta'/4 \). |
| 3. Invoke \( \text{Sample-Edge}(H_{k-1}, N_{k-1}, \beta, \tau, \vec{p}) \). Let \( Q \) be the returned \((k - 1)\)-clique if one was returned. Otherwise \( \text{FAIL} \). |
| 4. If \( \text{Define-} L_0(H_k, v_Q, \delta = \frac{\beta'}{N_F} \cdot \left( \frac{\beta}{k \log(n/\beta)} \right)^{O(k)}, \beta', \tau, \vec{p} \) = NO then \( \text{FAIL} \). |
| 5. If \( d(Q) \leq 40\tau/\beta'^2 \), then flip a coin with bias \( d(Q)/(40\tau/\beta'^2) \), and if it comes out Heads: |
| (a) Select a neighbor \( w \) of \( Q \) uniformly at random. |
| (b) Otherwise (\( d(Q) > 40\tau/\beta'^2 \)) then repeat the following \( \left\lceil \frac{3\min\{d(Q), \sqrt{n\alpha\gamma}\}}{\beta'\tau} \right\rceil \) times: |
| (i) Select a neighbor \( w \) of \( Q \) uniformly at random. |
| (ii) Otherwise (\( d(Q) > 40\tau/\beta'^2 \)) |
| (a) Invoke the procedure \( \text{Sample-Basic-Edge}(G, n, \beta/3, \vec{p}) \). Let \( (w, z) \) denote the returned edge if one was returned. |
| (b) If \( d(w) \geq d(Q) \), then keep \( w \) with probability \( \sqrt{n\alpha\gamma}/d(w) \). |
| (c) Check if \( \{Q, w\} \) is a \( k \)-clique assigned to \( Q \). If so, return \( C = \{Q, w\} \). |

Claim 3.13. Consider an invocation of \( \text{Sample-} E(L_0) \) with \( H_k, \beta, \tau \) and \( \vec{p} = (G, n, k, \alpha) \) such that \( \frac{4\alpha}{\beta} \leq \tau \leq \sqrt{n\alpha} \). Assume that \( L_0 \) is \( \tau \)-good, and that \( \text{Sample-Edge} \) invoked in Step 3 is a \( (\beta, X) \)-good edge-sampling procedure with respect to \( H_{k-1} \) for some \( X \) such that \( X\tau^2 \geq n\alpha^k \).

Then the subroutine \( \text{Sample-} E(L_0) \) is a \( (\beta', X') \)-good \( E(L_0) \)-sampling subroutine with respect to \( H_k \) for \( X' = 40X\tau/\beta'^2 \) and \( \beta' = 4\beta \) (recall Definition 3.6).

Furthermore, the expected query and time complexity of the subroutine are \( O(q_1 + \log n) \) and the maximum query and time complexity are \( O(q_2 + \frac{\log n}{\sqrt{m\gamma}}) \), where \( q_1 \) and \( q_2 \) are the expected and maximum, respectively, query and time complexity of \( \text{Sample-Edge}(H_{k-1}, N_{k-1}, \beta, \tau, \vec{p}) \).

Proof. Due to Step 4, if in Step 3 a \((k - 1)\)-clique \( Q \) is sampled such that \( v_Q \notin L_0(H_k) \), then the subroutine fails. Hence, only edges in \( E(L_0(F)) \) have non-zero probability of being returned. Consider an oriented edge \((v_Q, v_{Q'})\) for some \( v_Q \in L_0(H_k) \). By the definition of \( H_k \), the edge \((v_Q, v_{Q'})\) in \( H_k \) corresponds to a \( k \)-clique \( C \) in \( G \) that is assigned to the \((k - 1)\)-clique \( Q \). By the assumption that \( \text{Sample-a-Clique} \) is a \( (\beta, X) \)-good \((k - 1)\)-clique sampling subroutine, \( Q \) is sampled in Step 3 with probability \( \frac{1 + \beta^2}{X} \). Condition on this event happening.
First consider the case that \( d(Q) \leq 40\tau / \beta^2 \). Then \( w \) is sampled with probability \( \frac{d(Q)}{40\tau / \beta^2} \cdot \frac{1}{d(Q)} = \frac{\beta^2}{40\tau} \).

Next consider the case that \( d(Q) > 40\tau / \beta^2 \). Note that in this case an edge is sampled only if the coin toss at Step 6 Heads. We again consider two separate cases. If \( Q \) is such that \( d(Q) \leq \sqrt{n\alpha\gamma} \), then every clique \( \{Q, w\} \in A(Q) \) is sampled with probability \( \frac{\beta^2}{40\tau d(Q)} \), where the \( \beta^2/40 \) factor is due to the tossing of the coin. Otherwise \( d(Q) > \sqrt{n\alpha\gamma} \), and by Claim 2.5, \( d(w) \geq d(C') > \sqrt{n\alpha\gamma} \). Furthermore, by Theorem 4, an invocation of Sample-an-Edge(\( G, \alpha, \beta/3 \)) returns each oriented edge in the graph with probability \( \frac{1+\beta/3}{n\alpha\gamma} \) for \( \gamma = 4[\log^2 n]/\varepsilon \). Hence, each assigned clique \( \{Q, w\} \) in \( A(Q) \) is sampled and kept with probability \( \frac{(1+\beta/3)\beta d(w)}{n\alpha\gamma} \cdot \frac{1+\beta/3}{\sqrt{n\alpha\gamma}} = \frac{(1+\beta/3)\beta^2}{40\sqrt{n\alpha\gamma}} \). It follows that for every \( Q \) such that \( d(Q) > 40\tau / \beta^2 \), in each invocation of the loop in Step 6, every \( k \)-clique in \( A(Q) \) is sampled with almost equal probability \( \frac{(1+\beta/3)\beta^2}{40\min\{d(Q),\sqrt{n\alpha\gamma}\}} \). Hence, the success probability of a single invocation is \( \frac{1}{40\min\{d(Q),\sqrt{n\alpha\gamma}\}} \cdot \frac{1+\beta/3}{\beta} \). Let \( x \) denote this probability.

Let \( y \equiv \frac{3\min\{d(Q),\sqrt{n\alpha\gamma}\}}{\beta\tau} \), and \( E \) denote the event that a \( k \)-clique is returned in one of the invocations. Then

\[
\Pr[E] = 1 - (1 - x)^y \approx 1 - \left(1 - \frac{(1+\beta/3)\beta^2 a(Q)}{40\min\{d(Q),\sqrt{n\alpha\gamma}\}}\right)^y = 1 - \frac{3\min\{d(Q),\sqrt{n\alpha\gamma}\}}{\beta\tau} \cdot \frac{(1+\beta/3)\beta^2 a(Q)}{40\min\{d(Q),\sqrt{n\alpha\gamma}\}}.
\]

By the assumption that \( L_0 \) is \( \tau \)-good, and since \( \nu_Q \in L_0(H_k) \), it holds that \( a(Q) \leq 2\tau \), and thus \( x \cdot y < \frac{6(1+\beta/3)\beta^2 a(Q)}{40\tau} \). Therefore, by Claim 3.12, it holds that

\[ xy(1 - 2xy) \leq \Pr[E] \leq xy(1 + 1/y). \]

Since \( a(Q) < 2\tau \) and \( \beta < 1/2 \), \( 2xy \leq \frac{6(1+\beta/3)\beta^2 a(Q)}{40\tau} < \beta/3 \). Also, since \( d(Q) \geq 40\tau / \beta^2 \), and by the assumption \( \tau < \sqrt{n\alpha\gamma} \), it holds that \( 1/y = \frac{\beta\tau}{3\min\{d(Q),\sqrt{n\alpha\gamma}\}} \leq \beta/3 \). Hence, \( \Pr[E] \in xy \cdot (1 + \beta/3) \), implying

\[ \Pr[E] \in \frac{(1+\beta/3)\beta \cdot a(Q)}{40\tau} \leq \frac{1}{40\tau} \cdot \frac{(1+\beta)}{\beta \cdot a(Q)}. \]

Also, observe that conditioned on a \( k \)-clique being returned, each \( k \)-clique is returned with probability \( \frac{1}{(1+\beta/3)\beta a(Q)} \). Therefore, for the case \( d(Q) > 40\tau / \beta^2 \), it holds that for every specific assigned \( k \)-clique \( C \) of \( Q \),

\[ \Pr[C \text{ is returned}] = \frac{(1+\beta/3)^2 \cdot \beta^2 a(Q)}{40\tau} \cdot \frac{1}{(1+\beta/3)\beta a(Q)} \in \frac{(1+\beta)}{40\tau} \cdot \frac{\beta^2}{40}. \]

It follows that for every \( k \)-clique \( C \) corresponding to an oriented edge in \( E(L_0(H_k)) \),

\[ \Pr[C \text{ is returned}] \in \frac{1}{X} \cdot \frac{(1+\beta)\beta^2}{40\tau} = \frac{(1+\beta)\beta^2}{40X\tau} \]

(independently of the degree of the clique to which it is assigned).

We turn to analyze the query and time complexity of the procedure, which we shall denote by cost. Recall \( q_1 \) and \( q_2 \) denote the expected and maximum cost of a single invocation of the subroutine Sample-Edge(\( L_0 \)), respectively. Further note that by Theorem 4, each invocation of the
procedure Sample-Basic-Edge, takes $O(\log n)$ queries. Hence,

$$\mathbb{E}[\text{cost}] = O\left( q_1 + \frac{1}{X^2} \sum_{Q \in C_{k-1}} \frac{\beta^2 \cdot d(Q)}{40\tau} \cdot q_1 + \sum_{Q \in C_{k-1}} \frac{\beta^2 \log n}{X^2} \cdot \left\lceil \frac{\min\{d(Q), \sqrt{n\alpha\gamma}\}}{40\tau \beta} \right\rceil \right)$$

$$= O\left( q_1 + \frac{\beta \log n}{X^2} \sum_{Q' \in C_{k-1}} d(Q') \right) = O\left( q_1 + \frac{\beta \log n \cdot n\alpha^{-1}}{X^2} \right) = O(q_1 \log n).$$

where the second to last equality is due to Claim 3.10, and the last equality is by the assumption that $X^2 \geq n\alpha^{-1}$. Also, the maximum query complexity of the procedure is $O\left( q_2 + \frac{\log n \cdot \sqrt{n\alpha\gamma}}{\tau} \right)$. □

**Sample-a-Neighbor**($H_k, v_Q, \beta, \tau, \bar{p} = (G, n, k, \alpha)$)

1. Let $Q$ be the $(k-1)$-clique in $G$ corresponding to $v_Q$.
2. Repeat at most $r = \frac{\min\{d(Q), \sqrt{n\alpha\gamma}\}}{\tau} \cdot 2\ln(3/\beta)$ times:
   (a) If $d(Q) \leq \sqrt{n\alpha\gamma}$:
      i. Sample a neighbor $w$ of $Q$ uniformly at random.
   (b) Else, if $d(Q) > \sqrt{n\alpha\gamma}$:
      i. Invoke the procedure Sample-Basic-Edge($G, n, \beta/3, \bar{p}$). Let $(w, z)$ denote the returned edge, if one was returned. Otherwise, return to Step 2.
      ii. If $d(w) \geq d(Q)$, then keep $w$ with probability $\sqrt{n\alpha\gamma}/d(w)$. Otherwise, return to Step 2.
   (c) If $Q$ and $w$ form a $k$-clique $C$, and $C$ is assigned to $Q$, then return the edge in $F$ that corresponds to $C$.

**Claim 3.14.** The subroutine Sample-a-Neighbor is a $(\beta, \tau)$-good neighbor-sampling subroutine with respect to $H_k$, as defined in Definition 3.7. The query and time complexity of the subroutine are $O\left( \frac{\min\{d(Q), \sqrt{n\alpha\gamma}\}}{\tau} \cdot \log n \log(1/\beta) \right)$.

**Proof.** Let $v_Q$ be a node in $H_k$ such that $v_Q \notin L_0(H_k)$. We shall prove that every $k$-clique in $\mathcal{A}(Q)$ is returned with probability in $\frac{1+\beta}{d(Q)}$, implying that every incident edge of $v_Q$ in $H_k$ is returned with probability $\frac{1+\beta}{d_{H_k}(v)}$.

If $d(Q) \leq \sqrt{n\alpha\gamma}$, then for every assigned clique $C = \{Q, w\}$ of $Q$, $w$ is sampled with probability $1/d(Q)$. Otherwise $d(Q) > \sqrt{n\alpha\gamma}$, and by Claim 2.5, $d(w) \geq d(C) > \sqrt{n\alpha\gamma}$. Also, by Theorem 4, the invocation of Sample-Basic-Edge($G, n, \alpha, \beta/10, \bar{p}$) in Step 2(b)i returns each oriented edge in the graph $G$ with probability $\frac{1+\beta/10}{\sqrt{n\alpha\gamma}} \cdot \frac{\sqrt{n\alpha\gamma}}{d(w)} = \frac{1+\beta/10}{\sqrt{d(w)}}$. Hence, for every $Q$ (independently of its degree), each $k$-clique assigned to $Q$ is sampled with probability $\frac{1+\beta/10}{\min\{d(Q), \sqrt{n\alpha\gamma}\}}\cdot \frac{1+\beta/10}{\sqrt{n\alpha\gamma}}$.

Let $E$ denote the event that an assigned $k$-clique of $Q$ is sampled in one of the $r$ invocations of the loop in Step 2. Recall that by the assumptions that $v \notin L_0$ and that $L_0$ is $\tau$-good, it holds that

$$\mathbb{P}(E) = 1 - (1 - \frac{1+\beta}{d(Q)})^r \geq 1 - \exp\left(-\frac{r}{d(Q)}\right) \geq 1 - \exp\left(-\frac{\min\{d(Q), \sqrt{n\alpha\gamma}\}}{\tau} \cdot 2\ln(3/\beta)\right) = 1 - \exp\left(-\frac{\min\{d(Q), \sqrt{n\alpha\gamma}\}}{\tau} \cdot \ln(3/\beta)\right) \geq 1 - \frac{\min\{d(Q), \sqrt{n\alpha\gamma}\}}{\tau} \cdot \ln(3/\beta).$$

Hence, the query and time complexity of the subroutine are $O\left( \frac{\min\{d(Q), \sqrt{n\alpha\gamma}\}}{\tau} \cdot \log n \log(1/\beta) \right)$. □
Claim 3.15. Consider an invocation of the subroutine Define-\(L_0\) with parameters \(H_k, N_k, v_Q, \delta, \beta, \tau, \bar{p} = (G, n, k, \alpha)\). Let \(Q\) denote the \((k-1)\)-clique corresponding to the node \(v_Q\). With probability at least \(1 - \delta/\sqrt{N_k}\), the followings hold.

- If \(a(Q) \leq \tau\), then the subroutine returns YES.
\[ \text{If } a(Q) > 2\tau, \text{ then the subroutine returns NO.} \]

Furthermore, the query and time complexity of the subroutine are
\[ O \left( \frac{\sqrt{\min\{d(Q), \sqrt{\alpha\gamma}\}} \cdot k \log^2(n/\delta)}{\tau} \right). \]

**Proof.** If \( d(Q) \leq \sqrt{\alpha\gamma} \), then
\[ \mathbb{E}_{w \in \Gamma(Q)}[\chi_w] = \frac{1}{d(Q)} \sum_{w \in \Gamma(w)} \mathbb{1}_{\{Q,w\} \in A(Q)} = \frac{a(Q)}{d(Q)}. \]

Otherwise, if \( d(Q) > \sqrt{\alpha\gamma} \), then by Claim 2.5, for every \( w \) such that \( \{Q,w\} \in A(Q) \), \( d(w) \geq d(Q) > \sqrt{\alpha\gamma} \). By Theorem 4, an invocation of \textbf{Sample-Basic-Edge}(\( G, n, \beta, \bar{p} \)) returns each oriented edge in \( G \) with probability \( \frac{1+\beta}{\sqrt{\alpha\gamma}} \). Hence, each assigned clique \( \{Q,w\} \) of \( Q \) is sampled and kept with probability \( \frac{1+\beta}{d(w)} \cdot \frac{\sqrt{\alpha\gamma}}{d(w)} = \frac{1+\beta}{\sqrt{\alpha\gamma}} \). It follows that in the case that \( d(Q) > \sqrt{\alpha\gamma} \),
\[ \mathbb{E}_{w \in \Gamma(Q)}[\chi_w] = \frac{1}{d(Q)} \sum_{w \in \Gamma(w)} \mathbb{1}_{\{Q,w\} \in A(Q)} = \frac{(1+\beta)\alpha\gamma}{\sqrt{\alpha\gamma}}. \]

Hence, in general, \( \mathbb{E}[\chi_w] \in \left( \frac{(1+\beta)\alpha\gamma}{\min(d(Q), \sqrt{\alpha\gamma})} \right) \). Therefore, by the multiplicative Chernoff bound, as the \( \chi_w \) variables are \( \{0,1\} \) independent variables, it holds that
\[ \Pr \left[ \frac{1}{r} \sum_{i=1}^{r} \chi_{w_i} - \mathbb{E}[\chi_w] > \frac{1}{4} \mathbb{E}[\chi_w] \right] < \exp \left( -\frac{r \cdot \mathbb{E}[\chi_w]}{48} \right). \]

For \( Q \) such that \( a(Q) > 2\tau \), \( \mathbb{E}[\chi_w] > \frac{(1-\beta)2\tau}{d(Q)} \), so that by Equation (3.2) and the setting of \( r = \frac{\min(d(Q), \sqrt{\alpha\gamma})}{(1-\beta)2\tau} \cdot 50 \log(1/\delta') \), \( \Pr[\bar{a} < \frac{3}{4} \cdot \frac{(1-\beta)2\tau}{d(Q)} ] < \delta' \). Hence, with probability at least \( 1 - \delta' \), \( \bar{a} > 1.5 \cdot \frac{\tau}{d(Q)} \) and the subroutine will return NO (where we used \( \beta < 1/4 \)).

If \( d(Q) < \tau \), then consider a random variable \( \chi' \) for which \( \mathbb{E}[\chi'] = \frac{(1+\beta)\tau}{d(Q)} \). Then, by Equation (3.2) and the setting of \( r \) and \( \bar{a} \),
\[ \Pr \left[ \frac{1}{r} \sum_{i=1}^{r} \chi_{w_i} > \frac{5}{4} \cdot \frac{(1+\beta)\tau}{d(Q)} \right] \leq \Pr \left[ \frac{1}{r} \sum_{i=1}^{r} \chi'_{i} > \frac{5}{4} \cdot \frac{(1+\beta)\tau}{d(Q)} \right] < \delta'. \]

Hence, with probability at least \( 1 - \delta' \), \( \bar{a} < 1.5\tau/d(Q) \) the subroutine will return YES (where here too we used \( \beta < 1/4 \)).

Finally, by Theorem 4, the query complexity of each invocation of the loop is \( O(\log n) \) in case \textbf{Sample-Basic-Edge} is invoked, and \( O(1) \) otherwise. By the setting of \( \delta' = \delta/\sqrt{F} \), it follows that the query and time complexity of the subroutine are \( O(\tau) = O \left( \frac{\min(d(Q), \sqrt{\alpha\gamma}) \cdot \log^2(n/\delta)}{\tau} \right) \) as claimed. \( \square \)

**Corollary 3.16.** If the subroutine \textbf{Define-L}_0 is invoked with \( H_k, \sqrt{K}, \delta, \beta, \tau \) such that \( |H_k| \leq \sqrt{K} \), then it is a \((\delta, \tau)\)-good \( L_0 \) oracle with respect to \( H_k \), as defined in Definition 3.5. That is, with probability at least \( 1 - \delta \), the subroutine \textbf{Define-L}_0 defines a \( \tau \)-good set \( L_0 \) for the graph \( H_k \); for every \( v_Q \) such that \( a(Q) \leq \tau \), it returns YES, and for every \((k - 1)\)-clique \( Q \) such that \( a(Q) > 2\tau \), it returns NO.
Proof. Assume that each \((k - 1)\)-clique \(Q\) is assigned a unique random string \(R_Q\) so that whenever Define-\(L_0\) is invoked with \(v_Q\) the subroutine uses the same randomness \(R_Q\). Then we get that the with probability at least \(1 - \delta\), over the choice of these random strings, all \(Q\) cliques are correctly classified with probability at least \(1 - \frac{\delta}{\sqrt{n}} > 1 - \delta\). Hence, with probability at least \(1 - \delta\), Define-\(L_0\) is an oracle to a \(\tau\)-good set \(L_0\). \(\square\)

3.3 Proving the recursion

Given the statements regarding the query complexity of the subroutines Sample-\(E(L_0)\) Sample-a-Neighbor and Define-\(L_0\), we are now ready to prove the statement regarding the query complexity of the procedure Sample-\(E\).

Lemma 3.17 (Sample-\(E\) complexity). Consider an invocation of the procedure Sample-\(E\) with parameters \((F, N_F, \beta, \bar{p} = (G, n, k, \tau, \alpha))\). Let \(q_{n, \tau, \gamma}^{exp}(k - 1, \beta')\) and \(q_{n, \tau, \gamma}^{max}(k - 1, \beta')\) denotes the expected and maximum query complexities of the recursive invocation of the procedure Sample-\(E\) in Step 3 of Sample-\(E(L_0)\). Also, assume that the followings hold.

1. if Define-\(L_0\) is \(\tau\)-good with respect to \(H_k\) then Sample-\(E(L_0)(H_k, N_k, \beta', \bar{p})\) invoked in Step 4 is a \((\beta', X)\)-good \(E(L_0)\)-sampling subroutine with respect to \(H_k\) for \(X\) such that \(X \cdot \tau = \Omega^*(n\alpha^k)\).
2. Define-\(L_0\) invoked in Step 5a is a \((\delta, \tau)\)-good \(L_0\)-oracle with respect to \(H_k\) for \(\delta \leq \beta'/X\).
3. Sample-a-Neighbor\((H_k, v_Q, \beta', \bar{p})\) invoked in Step 5b is a \(\beta'\)-good neighbor-sampling subroutine with respect to \(H_k\).

Then for \(k = 2\) the expected and maximum query and time complexities of the procedure are \(O(\log n)\). For \(k \geq 3\), the expected query and time complexity of the procedure is \(O^*\left(q_{n, \tau, \gamma}^{exp}(k - 1, \beta/(2s + 2)) + 1\right)\), and the maximum query complexity is \(O^*\left(q_{n, \tau, \gamma}^{max}(k - 1, \beta/(2s + 2)) + \frac{\sqrt{n\alpha}}{\tau}\right)\).

\[\text{Pr}[(u, u') = (v_Q, v_{Q+1})] \leq \frac{1 + \beta}{X(s_k + 1)}.\]
Hence, the expected cost of the \(i\)th invocation of the loop is
\[
\mathbb{E}[\text{cost}] = O^*(\sum_{(v_Q,v_Q') \in E_{F}} P_i[(v_Q,v_Q')] \cdot \left[ \min\{d(Q),\sqrt{n\alpha}\gamma\} \middle/ \tau \right])
\]
\[
= O^*(\frac{1}{X(s+1) \cdot \tau} \left( \sum_{C \in \mathcal{C}_k, |d(C)| \leq \tau} 1 + \sum_{C \in \mathcal{C}_k} d(C) \right))
\]
\[
= O^*(\frac{n\alpha^k}{X\tau}) = O^*(1)
\]
where the last inequality is by the assumptions that \(X \cdot \tau = \Omega^*(n\alpha^k)\) and since \(s = O^*(1)\). Hence, together with the recursive call and since there are \(s\) invocations of the loop, the expected query complexity of the procedure is
\[
O^*(q_{n,\tau,\gamma}^\text{exp}(k-1, \beta/(2s + 2)) + 1).
\]
The maximum query complexity of the procedure is the maximum cost of the recursive call plus the maximum cost of the invocations of the subroutines \textbf{Sample-a-Neighbor} and \textbf{Define-\text{L}_0} which is bounded by \(O^*(\frac{\sqrt{n\alpha}}{\tau})\). Hence, the maximum cost is
\[
O^*(\frac{q_{n,\tau,\gamma}^\text{max}(k-1, \beta/(2s + 2)) + \sqrt{n\alpha}}{\tau}).
\]

\[
\square
\]

We now prove our main lemma, Lemma 3.4, from which Theorem 3 and Corollary 3.3 follow.

**Lemma 3.18** (Lemma 3.4, restated). Consider an invocation of \textbf{Sample-Edge}(\(H_k, \overline{V}_k, \beta, \bar{p} = (G, n, k, \tau, \alpha)\)) where \(\overline{V}_k \geq |V_{H_k}| \cdot \frac{\alpha}{4} \cdot (4k \log n)^k < \tau < \sqrt{n\alpha}\). Let \(\beta_i\) denote the value of \(\beta\) with which the recursive call \textbf{Sample-Edge} for \(H_i\) was invoked (so that in particular \(\beta_k = \beta\)).

The procedure returns every edge in \(H_k\) with probability in
\[
\frac{1 \pm \beta_2}{n\alpha\gamma \cdot \tau^{k-2}} \cdot \prod_{i=3}^{k} \frac{\beta_i^2 \cdot (1 \pm \beta_i)}{(40i(s_i+1) \cdot \log^2(n/\beta_i))}.
\]

Furthermore, the expected query complexity of the procedure is \(O^*(1)\), and the maximum query complexity is \(O^*(\frac{\sqrt{n\alpha}}{\tau})\).

**Proof.** We prove the lemma by induction on \(k\). If invoked with \(F = H_2\) and \(\beta_2\) then \(F = G\), and By Theorem 4, each oriented edge in \(H_2\) is returned with probability \(\frac{1 \pm \beta_2}{n\alpha\gamma\tau}\) for \(\gamma = 4[\log^2 n]/\varepsilon\).

Therefore, each edge is returned with probability \(\frac{2(1 \pm \beta_2)}{n\alpha\gamma\tau}\).

We now assume that the induction claim holds for \(k - 1 \geq 2\) and prove it for \(k\). That is, we assume that, if \textbf{Sample-Edge} is invoked with parameters \(H_{k-1} \leq \overline{V}_{k-1}, \tau\) and \(\beta_{k-1}\), such that \(4\alpha/\beta_{k-1} \leq \tau \leq \sqrt{n\alpha}\), then each \((k-1)\)-clique in \(H_{k-1}\) is sampled with probability \(\frac{1 \pm \beta_2}{n\alpha\gamma \cdot \tau^{k-3}} \cdot 2 \prod_{i=3}^{k-1} \frac{\beta_i^2 \cdot (1 \pm \beta_i)}{(40i(s_i+1) \cdot \log^2(N_i/\beta_i))}\).

We shall prove that \textbf{Sample-Edge}(\(H_k, \overline{V}_k, \beta, \bar{p} = (G, n, k, \tau, \alpha)\)) returns each \(k\)-clique with probability \(\frac{1 \pm \beta_2}{n\alpha\gamma \cdot \tau^{k-2}} \cdot 2 \prod_{i=3}^{k} \frac{\beta_i^2 \cdot (1 \pm \beta_i)}{(40i(s_i+1) \cdot \log^2(N_i/\beta_i))}\). By Lemma 3.9, it is sufficient to prove that
1. If Define-$L_0$ is $\tau$-good with respect to $H_k$ then Sample-$E(L_0)(F, \overline{N}_F, \beta', \tau, \overline{p})$ invoked in Step 4 is a $(\beta', X)$-good $E(L_0)$-sampling subroutine for $L_0$ that is determined by Define-$L_0$.

2. The subroutine Define-$L_0(H_k, \overline{N}_k, v, \delta, \beta, \tau, \overline{p})$ is a $(\delta, \tau)$-good $L_0$-oracle with respect to $H_k$ for $\delta' \leq \beta'/X$.

3. The subroutine Sample-a-Neighbor$(F, v, \beta', \tau, \overline{p})$ invoked in Step 5b is a $\beta'$-good neighbor-sampling subroutine for $F$.

For the first item, we would like to prove that the conditions of Claim 3.13 regarding Sample-$E(L_0)$ are met, building on the recursion hypothesis. First observe that for every $i$, $\beta_{i-1} = \beta_i/(2s_i + 2)$ for $s_i = |\overline{N}_i| = O(k \log n)$. Hence, by the assumption $\frac{\beta_{i-1}}{\beta_i} (4k \log n)^k \leq \tau$, for every $i$, $\beta_i \cdot \tau > 4\alpha$, and in particular for $i = k - 1$. Also, by Claim 3.11, for every $i$, $|H_i| \leq \frac{na^{i-1}}{n} = \overline{N}_i$, so that by Corollary 3.16, Define-$L_0$ is a $(\delta, \tau)$-good oracle with respect to $H_{k-1}$. Hence, the recursive invocation of Sample-Edge in Step 3 meets the assumptions in the current lemma, and therefore, by the induction hypothesis, it holds that Sample-Edge as invoked in Step 3 is a $(\beta_{k-1}, X_{k-1})$-good $(k - 1)$-clique sampling procedure for $X_{k-1}$ such that each $k$-clique is returned with probability

\[
\frac{1 + \beta_2}{n\alpha \gamma \cdot \tau^{k-3}} \cdot \frac{2^{k-1}}{\prod_{i=3}^{k-1} (40i(s_i + 1) \cdot \log^2(n/\beta_i))} = \Omega^*(\frac{1}{n\alpha \gamma \tau^{k-2}})
\]

Since $na^{k-2} < n\alpha \tau^{k-3}$, it holds that $X_{k-1} \tau^2 = \Omega^*(na^k)$. Therefore, in the case that $L_0$ is $\tau$-good then the conditions of Claim 3.13 hold, implying that the first item holds. The second item holds by Corollary 3.16, and the third item holds by Claim 3.14.

Hence, by Lemma 3.9, the procedure Sample-Edge returns every $k$-clique in $G$ with probability $\frac{1 + \beta_2}{n\alpha \gamma \cdot \tau^{k-3}}$ for $X' = X \cdot \tau (s_k + 1) \cdot 40k \log^2(n/\beta_k)$. Hence, each $k$-clique in $H_k$ is sampled with probability

\[
\frac{1 + \beta_2}{n\alpha \gamma \cdot \tau^{k-3}} \cdot \frac{2^{k-1}}{\prod_{i=3}^{k-1} (40i(s_i + 1) \cdot \log^2(n/\beta_i))} \cdot \frac{\beta_k(1 \pm \beta_k)}{40k(s_k + 1) \cdot \log^2(n/\beta_k) \cdot \tau}
\]

This concludes the first item of the claim.

We now turn to analyze the expected and maximum query and time complexities of the procedure. Let $q_k$ denote the expected query complexity of Sample-Edge when invoked on the graph $H_k$. Here too we prove the claim by induction on $k$, starting from $k = 2$. By Theorem 4, Sample-Basic-Edge has $O(\log n)$ query and time complexity, so that $q_2 = O(\log n)$ and the claim holds. We assume the claim holds for $k - 1$ and prove it for $k$. By Claim 3.13, each invocation of Sample-$E(L_0)$ has expected cost $O^*(q_{k-1} + 1)$. Therefore, by Lemma 3.17, the expected query complexity of Sample-Edge when invoked for $H_k$ is $O^*(q_1)$. Therefore, the query complexity and running time are bounded by $O^*(q_1)$. By Claim 3.17, the maximum query and time complexity of the $O(k)$ recursive invocations is $O^*(\log n + \frac{\sqrt{\alpha}}{\tau} \cdot k) = O^*(\frac{\sqrt{\alpha}}{\tau})$. This concludes the proof of the lemma. \qed

We next prove Theorem 3.

**Proof of Theorem 3.** Consider an invocation Sample-a-Clique$(G, k, \alpha, \varepsilon, \pi_k, \tau)$ with $\pi_k \in [n_k, 2n_k]$ and $\overline{N}_k \geq |V_{H_k}|$, $\alpha \cdot (4k \log n)^k < \tau < \sqrt{n\alpha \gamma}$ for $\gamma = 4[\log^2 n] / \varepsilon$. 

18
By Lemma 3.18, each invocation of Sample-Edge with $H_k$ returns a each $k$-clique with probability $\frac{1+\beta_2}{n\alpha\gamma\cdot \tau^{-k-2}} \cdot \prod_{i=3}^{k+1} \frac{\beta_i}{\beta_i \cdot (1+\beta_i)}$.

Since for every $i$, $\beta_{i-1} = \beta_i/(2s_i + 2)$ and $s_i = \log(n\alpha^i)$, it holds that $\beta_i = \Omega(\beta_k/(k^2 \log n)^i)$. Hence, by the setting of $\beta_k = \varepsilon/10k$, the overall success probability of outputting any clique in a single invocation is

$$\Omega\left( \frac{(1-\beta_k)^k}{n\alpha\gamma \cdot \tau^{-k-2}} \cdot \left( \frac{\beta_k}{k \log n} \right)^{O(k)} \right) = \frac{n_k}{n\alpha\gamma \cdot \tau^{-k-2}} \cdot \Omega\left( \left( \frac{\beta}{(k \log n)} \right)^{O(k)} \right).$$

Hence, the expected number of invocations until a $k$-clique is returned is $O\left( \frac{n\alpha \tau^{k-2}}{n_k} \right)$. (4$k \log n/\varepsilon)^O(k)$. By Lemma 3.18, the expected cost of each invocation is $O^*(1)$, so that it follows that the expected complexity of the algorithm is

$$O^*\left( \frac{n\alpha \tau^{k-2}}{n_k} \right).$$

Therefore the claim regarding the expected complexity holds.

It also follows that there exists a sufficiently large constant $c_1'$, such that if we perform at least $t_1' = \frac{n\alpha \tau^{k-2}}{n_k} \cdot (k \log n/\varepsilon)^{c_1' - k}$ invocations, then with probability at least $1 - \varepsilon/10n_k$, a $k$-clique is returned. Hence, if we allow $t_1'' = \frac{n\alpha \tau^{k-2}}{n_k} \cdot (k \log n/\varepsilon)^{c_1' - k}$ queries (in $G$) for a sufficiently large $c_1 > c_1'$, then with probability at least $1 - \varepsilon/10n_k$, a $k$-clique is returned. We note that if $t_1''$ exceeds $n\alpha$, then the algorithm may simply read the entire graph by querying the neighbors of all vertices, which takes at most $n\alpha$ queries. Hence, we can let $t_1 = \min\{t_1', n\alpha\}$.

The claim regarding the maximum query complexity holds by the setting of $r$ in Step 3 of the algorithm. Hence, it remains to prove that the resulting distribution on $k$-cliques is pointwise $\varepsilon$-close to uniform. We start by showing that with probability at least $1 - \varepsilon/10n_k$, the algorithm does not halt in Step 3.

Consider random variables $\chi_j$ such that $\chi_j$ is the running time of the $j$th invocation. By Lemma 3.18, for every $j$, $\mathbb{E}[\chi_j] = O^*(1)$, and $\max\{\chi_j\} = O^*(\sqrt{n\alpha}/\tau)$. Hence, by the multiplicative Chernoff bound, if we perform at most $t_2 = \sqrt{n\alpha} \cdot (k \log n/\varepsilon)^{c_2 - k}$ invocations for a sufficiently large constant $c_2$, then we have that, with probability at least $1 - \varepsilon/10n_k$, the number of queries does not exceed its expected value $O^*(t_2) = O^*(\sqrt{n\alpha}/\tau)$. Therefore, if we halt the algorithm after performing at most $t_3 = \max\{\min\{n\alpha, \frac{n\alpha \tau^{k-2}}{n_k} \cdot \frac{\sqrt{n\alpha}}{\tau}\}, (k \log n/\varepsilon)^{c_3 - k}$ queries for a sufficiently large constant $c_3$, it holds that with probability at least $1 - \varepsilon/10n_k$, a $k$-clique is returned. Hence, with probability at least $1 - \varepsilon/5n_k$, the algorithm does not fail in Step 3 (since $t_3 \geq t_2$) and it returns a $k$-clique (since $t_3 \geq t_1$).

Finally, conditioned on a $k$-clique being returned, each $C$ is returned with probability in $[\frac{1+\varepsilon/10}{1+\varepsilon/100n_k}, \frac{1+\varepsilon/100n_k}{n_k}]$. Hence, after $t$ invocations, each $k$-clique is returned with probability $\left[\frac{(1-\varepsilon/5)(1-\varepsilon/50n_k)}{1+\varepsilon/100n_k}, \frac{1+\varepsilon/100n_k}{n_k}\right] \in [\frac{1+\varepsilon}{n_k}, \frac{1+\varepsilon}{n_k}]$ so that the resulting distribution on $k$-cliques is $\varepsilon$-close to uniform. This concludes the proof of the theorem.

\textit{Proof of Corollary 3.3.} If the algorithm is not given the estimate $\overline{n}_k$ and a parameter $\tau$ then it proceeds as follows.
First, it invokes the \( k \)-clique counting algorithm by [ERS20a] with an approximation parameter \( \varepsilon = 1/4 \) for \( t = \log(10 n_k/\varepsilon) \) times. It then sets \( \pi_k \) to be twice the median of the returned estimates. Since each invocation of the counting algorithm returns an estimate \( \pi_k \in (1 \pm 1/4)n_k \) with probability at least \( 2/3 \), it holds that with probability at least \( 1 - \varepsilon/5n_k \), \( \pi_k \in [n_k, 2n_k] \). As the expected complexity of each such invocation is \( O^*(n\alpha^{k-1}/n_k) \), these invocations do not asymptotically affect the complexity of the algorithm. However, as the bound on the complexity of the counting algorithm is only on the expected value, and it is not guaranteed to hold with high probability, we must remove the condition of Step 3, so that our guarantee regarding the complexity of the algorithm only holds in expectation.

The algorithm then sets \( \tau = \max \left\{ \left( 1 - \varepsilon \right) \cdot (4k \log n)^k, \left( \pi_k / \sqrt{n\alpha} \right)^{1/(k-1)} \right\} \). Clearly for both cases \( \tau \geq \left( \frac{\alpha}{\varepsilon} \right) \cdot (4k \log n)^k \). Also, if \( \tau = \left( \frac{\alpha}{\varepsilon} \right) \cdot (4k \log n)^k \) then \( \tau \leq \sqrt{n\alpha} \). Otherwise, for \( \tau = \frac{\pi_k^{1/k}}{(n\alpha)^{(k/(k-1))}} \), it holds that \( \tau < \frac{1}{\sqrt{n\alpha}} \).

Finally it sets \( \varepsilon' = \varepsilon/3 \) and runs the algorithm as before (with the new \( \varepsilon' \) value) and without the halting condition in Step 3.

By the above settings, if \( \pi_k \in [n_k, 2n_k] \), then the conditions of Theorem 3 hold. Therefore, if \( \pi_k \in [n_k, 2n_k] \), it holds that the resulting distribution on \( k \)-cliques is pointwise \( (\varepsilon/3) \)-close to uniform. Since the condition holds with probability at least \( 1 - \varepsilon/5n_k \), it follows that the resulting distribution on \( k \)-cliques is pointwise \( \varepsilon \)-close to uniform. \( \Box \)

4 A Lower Bound for Sampling Cliques

In this section we prove a lower bound on sampling \( k \)-cliques from a distribution that is \( \varepsilon \)-pointwise close to uniform. We show that \( \Omega\left( \max \left\{ \left( \frac{(n\alpha)^{k/2}}{k^{k-n_k}} \right)^{1/k}, \frac{n\alpha^{k-1}}{n_k} \right\} \right) \) queries are necessary in order to ensure that each \( k \)-clique is sampled with probability at least \( 1/(2n_k) \).

The second term in this lower bound follows from a related lower bound on approximate counting of the number of \( k \)-cliques. We discuss this term briefly in Section 4.4. The main focus of this section is on the first term. Observe that as long as \( n_k \leq k^{1/2} \pi \alpha^{k-1/2} \), the second term is dominant. We hence establish the lower bound of \( \Omega(Q) \) for \( Q = \left( \frac{(n\alpha)^{k/2}}{k^{k-n_k}} \right)^{1/k} \), assuming that \( n_k > k^{1/2} \pi \alpha^{k-1/2} \), so that \( Q \) is smaller than \( \sqrt{n\alpha} < \sqrt{n\alpha} \) by a sufficiently large constant factor.

In a nutshell, the lower bound of \( \Omega(Q) \) is based on “hiding” a \( k \)-clique and proving that any algorithm that outputs this clique with probability \( \Omega(1/n_k) \), must perform \( \Omega(Q) \) queries. This complexity (almost) matches the upper bound of Theorem 2, thus proving the optimality of our algorithm (up to a \( (\log n/\varepsilon)^{O(k)} \) factor).

4.1 The lower bound construction

We define a family of graphs, denoted \( \mathcal{G} \), where all graphs in the family have the same basic structure, and only differ in the labeling of the vertices and edges. The graphs are over \( n \) vertices and have arboricity at most \( \alpha \). For a given integer \( \tilde{n}_k \), all graphs contain \( n_k = \Theta(\tilde{n}_k) \) \( k \)-cliques.

In what follows we assume for simplicity (and without loss of generality), that various expressions are integers (e.g., \( \sqrt{n\alpha} \)). We also assume that \( n \) is sufficiently large. The basic underlying graph structure, denoted \( G = (V, E) \), is as follows. The set of vertices, \( V \) is partitioned into four subsets:
A, B, C and D. For \( n' \) that satisfies \( n = 2n' + \sqrt{n'\alpha \cdot k} \), we have that \( |A| = \ell = \sqrt{n'\alpha} \), \( |B| = n' \), \( |C| = \ell \cdot (k - 1) \) and \( |D| = n' \). Furthermore, the vertices in \( A \) have labels in \( [\ell] = \{1, \ldots, \ell\} \), the vertices in \( B \) have labels in \( \{\ell + 1, \ldots, \ell + n'\} \), the vertices in \( C \) have labels in \( \{\ell + n' + 1, \ldots, \ell \cdot k + n'\} \) and the vertices in \( D \) have the remaining labels. Each edge \( \{u, v\} \) corresponds to two oriented edges, \((u, v)\) and \((v, u)\), where each has its own label (the label of \((u, v)\) is in \( [d(u)] \) and the label of \((v, u)\) is in \( [d(v)] \)).

The subgraph of \( \tilde{G} \) induced by \( D \) is a fixed subgraph that contains \( n_k - 1 = \Theta(n_k) \) \( k \)-cliques (and has arboricity at most \( \alpha \)). Each vertex in \( A \) has \( \ell \) neighbors in \( B \), and each vertex in \( B \) has \( \alpha \) neighbors in \( A \) (observe that indeed \( |A| \cdot \ell = \ell^2 = n'\alpha \) and \( |B| \cdot \alpha = n'\alpha \)). This bipartite subgraph is fixed, and furthermore, the label of each (oriented) edge \((v, u)\) for \( v \in B \) and \( u \in A \) is fixed as well. However, the labels of the edges \((u, v)\) (for \( u \in A \), \( v \in B \)) are not fixed, and vary between graphs in \( G \). The graphs in \( G \) also differ in the choice of a subset \( S \subset A \) of \( k \) special vertices, where the subgraph induces by \( S \) is a \( k \)-clique. Finally, each vertex \( u \in A \setminus S \) has \( k - 1 \) distinct neighbors in \( C \) (where the choice of these neighbors differs as well between graphs in \( G \)). Observe that all vertices in \( A \) have degree \( \tilde{d} = \ell + (k - 1) \), all vertices in \( B \) have degree \( \alpha \) and all vertices in \( C \) have degree 1. See Figure 4.1 for an illustration of a graph in \( G \) for the case of \( k = 4 \). As noted above, while all graphs in \( G \) have the same underlying structure, as labeled graphs they differ in the subgraph induced by \( A \cup C \) (and in particular in the choice of the special subset \( S \subset A \)), as well as in the labels of the edges \((u, v)\) for \( u \in A \) and \( v \in B \).

![Diagram](image.png)

Figure 1: The underlying structure of the graphs in \( G \) for \( k = 4 \).

4.2 The process \( \mathcal{P} \) and the knowledge graph

We now define a process \( \mathcal{P} \) that answers queries of a sampling algorithm \( \text{Alg} \) while constructing a uniformly selected graph \( \hat{G} \in G \) on the fly. We shall be interested in showing that if \( \text{Alg} \) performs less than \( Q/c \) queries, for a sufficiently large constant \( c \), then the probability that it outputs the
labels of the vertices in the set $S$ is less than $1/(2n_k)$. Given our assumption that the labels of these vertices and their neighbors belong to $[n' + \ell \cdot k]$, we may assume without loss of generality that $\text{ALG}$ does not perform queries with labels in $\{n' + \ell \cdot k, \ldots, 2n' + \ell \cdot k\}$ (i.e., labels of vertices in $D$).

It will be helpful to consider what we refer to as the knowledge graph of $\text{ALG}$, after it performed $t$ queries, denoted $G^{kn}_t = (V_t, E_t, \overline{E}_t, \tau_t, d_t)$. Here $V_t$ is the set of (labeled) vertices in the knowledge graph, $E_t$ is the set of (oriented) edges, $\overline{E}_t$ is the set of “non-edges” (corresponding to negative answers to pair queries), $\tau_t$ is the edge-labeling function of $G^{kn}_t$, and $d_t$ is the degree function of $G^{kn}_t$. More precisely, for each vertex $u$ that appeared either in one of the first $t$ queries of $\text{ALG}$ or was an answer to one of these queries, there is a vertex in $V_t$. If for $t' \leq t$ a neighbor query $q_{u'} = \text{nbr}(u, i)$ was answered by $v$, then there is an edge $(u, v) \in E_t$ labeled by $i$ and $\tau_t((u, v)) = i$. We shall assume that $\mathcal{P}$ also provides $\text{ALG}$ with the label $j$ of the edge $(v, u)$, so that $(v, u) \in E_t$ as well, and $\tau_t((v, u)) = j$. If for $t' \leq t$ a pair query $q_{uv} = \text{pair}(u, v)$ was answered positively, then $(u, v), (v, u) \in E_t$, and if it was answered negatively, then $(u, v), (v, u) \in \overline{E}_t$. In the former case we assume that $\text{ALG}$ is provided with the labels of the edges $(u, v)$ and $(v, u)$, so that $\tau_t((u, v))$ and $\tau_t((v, u))$ are set appropriately. Finally, for each $u \in V_t$, $d_t(u)$ is the degree of $u$ in $G^{kn}_t$. Note that $d_t(u) \leq d(u)$ where $d(u)$ is the degree of the vertex $u$ in the final graph $\hat{G}$ (e.g., $d(u) = \hat{d}$ for each $u \in A$).

For $X \subseteq \{A, B, C\}$ let $V_{t, X} \subseteq V_t$ denote the subset of vertices $u$ in $G^{kn}_t$ that belong to $X$. For each $u \in V_t$, let $\Gamma_{t, X}(u)$ denote its set of neighbors in $G^{kn}_t$, and for $X \subseteq \{A, B, C\}$ let $\Gamma_{t, X}(u) = \Gamma_{t}(u) \cap V_{t, X}$. We use $\overline{\Gamma}_{t}(u)$ to denote the subset of “non-edges” $(u, v) \in \overline{E}_t$ and $\overline{d}_{t}(u) = |\overline{\Gamma}_{t}(u)|$. Finally, let $S_t \subseteq A$ denote the subset of vertices $u \in A$ such that $|\Gamma_{t, A}(u)| > 0$, and let $\overline{S}_t$ denote the subset of vertices $u \in A$ such that $|\Gamma_{t, C}(u)| > 0$. Hence, if $u \in S_t$, then it is one of the $k$ vertices in the hidden $k$-clique, and if $u \in \overline{S}_t$, then it does not participate in any $k$-clique (so $u \in A \setminus S$ for the special subset $S$).

We refer to the edges between $A$ and $C$ as informative edges (since they indicate whether a vertex in $A$ belongs to the special set $S$ or to $A \setminus S$), and to the edges between two vertices of $A$ as witness edges.

### 4.3 Details on how $\mathcal{P}$ answers queries

The process $\mathcal{P}$ answers the queries of $\text{ALG}$ as follows. To answer a query $q_t$, $\mathcal{P}$ considers the subset of (labeled) graphs in $G$ that are consistent with the knowledge graph $G^{kn}_{t-1}$, which we denote by $G_{t-1}$. It then selects a graph $\hat{G}_t \in G_{t-1}$ uniformly at random and answers the query according to $\hat{G}_t$. Finally, it updates $G^{kn}_{t-1}$, to obtain $G^{kn}_t$ by incorporating the new information obtained from the answer to this query (including the additional information as described previously). Once all $T$ queries are performed, $\mathcal{P}$ uniformly selects a graph $\hat{G}$ in $G_T$ and this is the resulting graph. Note that the graphs $\hat{G}_t$ for $t < T$ are only used as a tool to aid the description of the process, so that each $\hat{G}_t$ can be viewed as being “discarded” after answering $q_t$. Observe that this process generates a uniformly distributed graph in $G$.

We shall say that $\text{ALG}$ succeeds, if, after performing $T$ queries, it outputs a subset $\hat{S}$, such that $\hat{S}$ equals the special subset $S$ (the hidden clique) of the final graph $\hat{G}$. In particular this is the case if $|S_T| = k$. However, the algorithm may output the special subset even if $|S_T| < k$. To address this possibility, it will be convenient to assume (without loss of generality), that in order to select the final graph $\hat{G}$, the process first determines whether each of the vertices in $\hat{S} \setminus S_T$ belongs to the hidden clique (in the same manner that it answers queries of the algorithm).
In order to analyze the success probability of any algorithm ALG, we introduce the following central notions.

**Definition 4.1** (Witness answers). We say that an answer \( a_t \) to a query \( q_t \) (given the knowledge graph \( G_{t-1}^{kn} \)) is an edge-witness answer, if two new vertices of the hidden clique are discovered. Namely, \( |S_t \setminus S_{t-1}| = 2 \). This event is denoted by \( \mathcal{E}_t^2 \). Similarly, we say that an answer \( a_t \) is a vertex-witness answer if one such new vertex is discovered, that is, \( |S_t \setminus S_{t-1}| = 1 \). This event is denoted by \( \mathcal{E}_t^1 \).

Observe that if \( \mathcal{E}_t^1 \) occurs then necessarily the event \( \mathcal{E}_t^2 \) occurred for some \( t' < t \).

We shall prove that the following holds for any algorithm ALG that performs \( T \leq \ell/4 \) queries: (1) each answer has probability at most \( 8k^2/\ell^2 \) to be an edge-witness answer, and (2) following the first edge-witness answer (the first edge in the hidden \( k \)-clique), each answer has probability at most \( 8k/\ell \) to be a vertex-witness answer. We may assume without loss of generality that ALG does not perform pair queries that include a vertex \( v \in C \), so that \( \overline{E}_t \) never contains such a pair. This is the case since such a pair query can be replaced by a neighbor query \((v,1)\), which returns the specific single neighbor \( u \in A \) of \( v \), and as a consequence also determines that there is no edge between \( v \) and any other vertex in \( A \).

**Claim 4.2.** For every \( t \leq \ell/4 \), every knowledge graph \( G_{t-1}^{kn} \) and every query \( q_t \),

\[
\Pr[\mathcal{E}_t^2] \leq \frac{8k^2}{\ell^2}.
\]

**Proof.** First consider the case that \( q_t = \text{pair}(u,u') \) for some \( u,u' \in A \). Note that unless \( u,u' \in A \setminus (S_{t-1} \cup \overline{S}_{t-1}) \) (and \( (u,u') \notin \overline{E}_{t-1} \)), we have that \( \Pr[\mathcal{E}_t^2] = 0 \). This holds since \( u,u' \notin S_{t-1} \) by the definition of \( \mathcal{E}_t^2 \), and if either \( u \) or \( u' \) belongs to \( \overline{S}_{t-1} \), then clearly the response cannot correspond to a witness edge.

In order to prove the claim (for pair queries), we shall bound the fraction of graphs in \( G_{t-1} \) such that \( u \) and \( u' \) belong to the hidden clique. Refer to such graphs as witness graphs, and to all other graphs in \( G_{t-1} \) as non-witness graphs. We denote the set of witness graphs (with the witness pair \((u,u')\)) by \( \mathcal{W}_t = \mathcal{W}_t(u,u') \), and of the set of non-witness graphs by \( \overline{\mathcal{W}}_t = \overline{\mathcal{W}}_t(u,u') \). (Here we use the subscript \( t \), since while these graphs belong to \( G_{t-1} \), they also depend on \( q_t = \text{pair}(u,u',1) \).) In order to bound the fraction of witness graphs in \( G_{t-1} \), we define the following auxiliary graph \( H_t \).

The graph \( H_t \) is a bipartite graph over the sets of nodes \( W \) and \( \overline{W} \), where in \( W \) there is a node for every graph in \( \mathcal{W}_t \) and in \( \overline{W} \) there is a node for every graph in \( \overline{\mathcal{W}}_t \).

Let \( G \) be a witness graph in \( \mathcal{W}_t \) with the special set \( S = \{u_0 = u, u_1 = u', r_1, \ldots, r_{k-2}\} \), and let \( w \) be the node it corresponds to in \( H_t \). For each two vertices \( y_0, y_1 \in A \setminus (S \cup \overline{S}_{t-1}) \) such that \( (y_0, y_1) \notin \overline{E}_{t-1} \) and \( (y_b, r_p) \notin \overline{E}_{t-1} \) for every \( b \in \{0,1\} \) and \( p \in [k-2] \), there is a neighbor \( \overline{w} \) of \( w \) in \( H_t \), corresponding to the graph \( \overline{G}' \in \overline{\mathcal{W}}_t \) that results from \( G \) by performing the following operations.

For \( b \in \{0,1\} \), let \( v_{b,1}, \ldots, v_{b,k-1} \) be the neighbors of \( y_b \) in \( C \). First, we remove the edge \((u_0, u_1)\) and add the edge \((y_0, y_1)\). Next, for each \( b \in \{0,1\} \) we remove the edge \((u_b, r_p)\) for each \( p \in [k-2] \), and the edge \((y_b, v_{b,p})\) for each \( p \in [k-1] \). Finally, for each \( b \in \{0,1\} \), we add the edges \((y_b, r_p)\) for each \( p \in [k-2] \) and the edges \((u_b, v_{b,p})\) for each \( p \in [k-1] \). (Recall that we assumed without loss of generality that there are no pairs in \( \overline{E}_{t-1} \) that contain a vertex in \( C \), so that such an addition is consistent with \( G_{t-1}^{kn} \).) See Figure 4.3 for an illustration.
It remains to specify the labels of all new edges. The new (oriented) edge \((y_0, y_1)\) is given the label of the removed edge \((y_0, v_{b,k-1})\), and \((y_1, y_0)\) is given the label of \((y_1, v_{1,k-1})\). For \(b \in \{0, 1\}\), each new edge \((y_b, r_p)\) for \(p \in [k-2]\) is given the label of the removed edge \((y_b, v_{b,p})\), each new edge \((u_b, v_{b,p})\) for \(p \in [k-2]\) is given the label of the removed edge \((u_b, r_p)\), and \((u_b, v_{b,k-1})\) is given the label of \((u_b, u_{1-b})\).

By the description of the process \(\mathcal{P}\), for any pair \((u, u')\) (where \(u, u' \in A \setminus (S_{t-1} \cup \overline{S}_{t-1})\) (and \((u, u') \notin \overline{E}_{t-1}\) )

\[
\Pr[\mathcal{E}_{t}^2 \mid q_t = \text{pair}(u, u')] = \frac{|W_{\ell}|}{|W_{\ell}|} = \frac{|W|}{|W|}.
\]

Let \(d_W\) and \(d_{\overline{W}}\) denote the average degree of the nodes in \(W\) and \(\overline{W}\), respectively. Since \(H_t\) is a bipartite graph, \(|W| = \frac{|\overline{W}|}{d_W}\). Hence, to bound \(\Pr[\mathcal{E}_{t}^2]\), we will be interested in bounding \(|W|\) by lower bounding \(d_W\) and upper bounding \(d_{\overline{W}}\).

By the definition of the set of neighbors of each node \(w\) in \(W\), the number of “eligible” pairs of vertices \((y_0, y_1)\) that can be used to define a neighbor \(\bar{w}\) of \(w\) is lower bounded by \(|[A] - |\overline{S}_{t-1} - \overline{E}_{t-1}|\). Since \(|S_{t-1}| + |\overline{E}_{t-1}| \leq \ell/4\), there are at least \(\ell^2/8\) such pairs. Therefore, \(d_W \geq \ell^2/8\).

Now consider a non-witness graph \(\overline{G}' \in \overline{W}\). Since \(\overline{G}'\) is a non-witness graph, \(u\) and \(u'\) do not belong to the special set \(S' = \{r_1, \ldots, r_k\}\) of \(\overline{G}'\). By the definition of \(H_t\), each neighbor in \(H_t\) of the node \(w_{\overline{G}_t}\) corresponding to \(\overline{G}'\) must correspond to a graph \(\overline{G} \in W_{\ell}\) that contains a \(k\)-clique over \(u, u'\) and \(k-2\) of the vertices in \(S'\). Therefore, \(w_{\overline{G}_t}\) has at most \(\binom{k}{2}\) neighbors in \(H_t\). Since this is true for every node \(\bar{w} \in \overline{W}\), we have that \(d_{\overline{W}} < k^2\). Therefore, \(|W| \leq \frac{\overline{W}k^2}{\ell/8}\) and so

\[
\Pr[\mathcal{E}_{t}^2 \mid q_t = \text{pair}(u, u')] = \frac{|W|}{|W|} \leq \frac{|W|}{|W|} \leq \frac{8k^2}{\ell^2}.
\]

The analysis of the case that \(q_t\) is a neighbor query \(\text{nbr}(u, i)\) is essentially the same. If \(a_t = u'\) is the returned neighbor, then, here too, \((u, u')\) may be a witness edge only if \(u, u' \in A \setminus (S_{t-1} \cup \overline{S}_{t-1})\) (and \((u, u') \notin \overline{E}_{t-1}\) ). The set of witness and non-witness graphs is defined for each possible answer \(u'\), where in each witness graph, the label of \((u, u')\) is \(i\). By the same reasoning as above (for every \(u, i\) and \(u'\), \(\Pr[\mathcal{E}_{t}^2 \mid q_t = \text{nbr}(u, i) \& a_t = u' \leq \frac{8k^2}{\ell^2}\), and the claim is established.

**Claim 4.3.** For every \(t \leq \ell/4\), every knowledge graph \(G^k_{t-1}\) and every query \(q_t\),

\[
\Pr[\mathcal{E}_{t}^1] \leq \frac{8k}{\ell}.
\]

**Proof.** The proof of this claim is similar to the proof of Claim 4.2, with some small changes to the definition of the auxiliary graph \(H_t\). We analyze the case that \(q_t\) is a pair query, and the case the \(q_t\) is a neighbor query is essentially the same. If \(q = \text{pair}(u, u')\), then \(\mathcal{E}_{t}^1\) may occur only if \(u \in S_{t-1}\) and \(u' \in A \setminus (S_{t-1} \cup \overline{S}_{t-1})\), or vice versa (and \((u, u') \notin \overline{E}_{t-1}\)).

Here too we define an auxiliary bipartite graph \(H_t\) over a set of nodes \(W \cup \overline{W}\), where there is a node in \(W\) for each graph in \(W_{\ell}\), and a node in \(\overline{W}\) for each graph in \(\overline{W}_{\ell}\). There is an edge between a node in \(W\) and a node in \(\overline{W}\) only if the corresponding graphs differ on a single vertex in the hidden clique (rather than two vertices, as was the case in the proof of Claim 4.2). More precise details follow.
Let $\hat{G}$ be a witness graph in $\mathcal{W}_t$ with the special set $S = \{u, u', r_1, \ldots, r_{k-2}\}$ where $u \in S_{t-1}$ and $u' \in A \setminus (S_{t-1} \cup \overline{S}_{t-1})$, and let $w$ be the node it corresponds to in $H_t$. For each vertex $y' \in A \setminus (S \cup \overline{S}_{t-1})$ such that $(u, y') \notin E_{t-1}$ and $(y', r_p) \notin \overline{E}_{t-1}$ for every $p \in [k-2]$, there is a neighbor $\bar{w}$ of $w$ in $H_t$, corresponding to the graph $\hat{G}' \in \overline{\mathcal{W}_t}$ that results from $\hat{G}$ by performing the following operations.

Let $v_1, \ldots, v_{k-1}$ be the neighbors of $y'$ in $C$. First, we remove the edge $(u, u')$ and add the edge $(y, y')$. Next, we remove the edge $(u', r_p)$ for each $p \in [k-2]$, and the edge $(y', v_p)$ for each $p \in [k-2]$. Finally, we add the edges $(y', r_p)$ for each $p \in [k-2]$ and the edges $(u', v_p)$ for each $p \in [k-2]$.

It remains to specify the labels of all new edges. The new (oriented) edge $(u, y')$ is given the label of the removed edge $(u, u')$, and $(y', u)$ is given the label of $(y', v_{k-1})$. Each new edge $(y', r_p)$ for $p \in [k-2]$ is given the label of the removed edge $(y', v_p)$, each new edge $(u', v_p)$ for $p \in [k-2]$ is given the label of the removed edge $(u', r_p)$, and $(u', v_{k-1})$ is given the label of $(u', u)$.

As before, by the description of the process $\mathcal{P}$, it holds that

$$\Pr[E^t | q_t = \text{pair}(u, u')] = \frac{|\mathcal{W}_t|}{|\hat{G}_t|} = \frac{|W|}{|W \cup \overline{W}|}.$$ 

Let $d_w, d_{\overline{W}}$ denote the average degrees of the nodes in $W$ and $\overline{W}$, respectively. Since $H_t$ is a bipartite graph, $|W| = \frac{|\mathcal{W}| d_w}{d_w}$. Hence, to bound $\Pr[a_t = W_2]$, we will be interested in bounding $|W|$ by lower bounding $d_w$ and upper bounding $d_{\overline{W}}$.

By the definition of the set of neighbors of each node $w$ in $W$, the number of “eligible” vertices $y'$ that can be used to replace $u'$ in the hidden clique and hence define a neighbor $\bar{w}$ of $w$ is lower bounded by $|A| - k - |S_{t-1}| - |\overline{E}_{t-1}|$. Since $|S_{t-1}| + |\overline{E}_{t-1}| \leq \ell/4$, there are at least $\ell/8$ such vertices. Therefore, $d_W \geq \ell/8$. 

![Figure 2: Two neighboring graphs in $H_t$ for $k = 5$.](image)
Now consider a non-witness graph \( \hat{G}' \in \overline{W}_t \). Since \( \hat{G}' \) is a non-witness graph, its special set \( S' = \{u, r_1, \ldots, r_{k-1}\} \) of \( \hat{G}' \) contains \( u \) but not \( u' \). By the definition of \( H_t \), each neighbor in \( H_t \) of the node \( \hat{w}_G \) corresponding to \( \hat{G}' \) must correspond to a graph \( \tilde{G} \in W_t \) that contains a \( k \)-clique over \( u, u' \) and \( k - 2 \) of the vertices among \( \{r_1, \ldots, r_{k-1}\} \). Therefore, \( \tilde{w}_G \) has at most \( k - 1 \) neighbors in \( H_t \). Since this is true for every node \( \tilde{w} \in \overline{W} \), we have that \( d_{\overline{W}} < k \).

Therefore, \( |W| \leq 8|\overline{W}| \cdot k/\ell \), and so

\[
\Pr[\mathcal{E}_t^1 \mid q_t = \text{pair}(u, u')] = \frac{|W|}{|W \cup \overline{W}|} = \frac{|W|}{|W|} \leq \frac{8k}{\ell}.
\]

As noted above (and similarly to the proof of Claim 4.2) showing that \( \Pr[\mathcal{E}_t^1 \mid q_t = \text{nbr}(u, i) \& a_t = u'] \leq 8k/\ell \) is the same, except that in each witness graph, the edge \((u, u')\) must be labeled \( i \).

We are finally ready to prove our main lower bound theorem.

**Proof of Theorem 1.** Recall that we assume for the sake of the analysis, that if the algorithm completes its execution after \( T \) queries and \( |S_T| < k \), then in order to select the final \( \hat{G} \in \mathcal{G}_T \), the process first determines whether the vertices in \( \hat{S} \setminus S_T \) belong to the hidden clique \( S(\hat{G}) \) of \( \hat{G} \) (in the same manner that it answers queries). Recall that by the definition of the witness events, for any \( t' \), unless the event \( \mathcal{E}_t^2 \) has occurred for some \( t < t' \), the event \( \mathcal{E}_{t'} \) cannot occur.

Therefore, for the algorithm to succeed, it must hold that for some \( k_2 \geq 1 \) and \( k_1 \geq 0 \) such that \( 2k_2 + k_1 = k \), there are \( k_2 \) indices \( t_1, \ldots, t_{k_2} \) and \( k_1 \) indices \( t'_1, \ldots, t'_{k_1} \), such that the events \( \mathcal{E}_{t_j}^2 \) for \( j \in [k_2] \) all hold and the events for \( j \in [k_1] \) \( \mathcal{E}_{t_j}^1 \) all hold. By Claim 4.2 and Claim 4.3 (and since \( \frac{T \cdot k^2}{\ell^2} < (\frac{T \cdot k}{\ell})^2 \) for \( x > 1 \))

\[
\Pr[\hat{S} = S(\hat{G})] \leq \frac{8T \cdot k^2}{\ell^2} \cdot \left( \frac{8T \cdot k}{\ell} \right)^{k-2} = \frac{(8T)^{k-1} \cdot k^k}{\ell^k}.
\]

By the setting of \( \ell = \Theta(\sqrt{n\alpha}) \), in order to have \( \Pr[\hat{S} = S(\hat{G})] = \Omega(1/n_k) \), it must hold that \( T = \Omega\left( \frac{(n\alpha)^{k/2}}{k^{k-n_k}} \right)^{\frac{1}{k-1}} \). \( \square \)

### 4.4 The counting-based lower bound

The second term in the lower bound of Theorem 1 follows directly from a lower bound of

\[
\Omega \left( \min \left\{ n\alpha, \frac{n(\alpha/k)^{k-1}}{n_k} \right\} \right)
\]

by [ERS20a] for \( k \)-clique counting. They prove that any algorithm that performs fewer queries than (2) cannot distinguish between two families of graphs with high probability: one with \( n_k \) \( k \)-clique, and one with no \( k \)-cliques. Since any \( k \)-clique sampling algorithm could distinguish between the two families (by returning a \( k \)-clique if a the graph belongs to the former family), any uniform sampling algorithm cannot perform fewer queries.
5 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. The 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 [Fei06] and Goldreich and Ron [GR08], who presented algorithms for approximately counting the number of $k$-cliques in a graph for $k = 2$ (edges).\(^9\) Later, Gonen, Ron and Shavitt [GRS11] gave essentially optimal bounds for the problem of approximately counting the number of stars in a graph. In [ELRS15, ERS20b] Eden, Levi, Ron and Seshadhri and Eden, Ron and Seshadhri presented essentially optimal bounds for the problems of approximately counting triangles and $k$-cliques.

Augmented model. In [ABG\(^+\)18], 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 [AKK18] considered the problem of approximate counting of arbitrary subgraphs $H$. The expected query complexity of their algorithm is $m^{ρ(H)}_{n_H}$, where $ρ(H)$ is the fractional edge cover of $H$\(^10\), and $n_H$ is the number of copies of $H$ in $G$. Their result is optimal for the case of $k$-clique and odd-cycle counting.

In [ER18a], 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.

Set query model. In [BHPR\(^+\)17], Beame, Har-Peled, Ramamoorth 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 [CLW19] who settled the complexity of the problem to $Θ^*(n/√m)$. In [BBGM19], Bhattacharya, Bishnu, Ghosh, and Mishra later have generalized the BIS model to tripartite set queries, where they considered the problem of triangle counting.

Uniform sampling. In [ER18b], 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 Têtek [Têtek20], so that the new algorithm allos sampling from the exact uniform distribution. In [FGP20], Fichtenberger, Gao and Peng proved that in the augmented edge model, exact uniform sampling of arbitrary subgraphs can be performed in $O\left(\frac{m^ρ(H)}{n_H}\right)$ time. This matches the upper bound of [AKK18] for the counting variant.

---

\(^9\)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 $(1 + ε)$-factor approximation algorithm with the same complexity as the previous one (as well as a matching lower bound).

\(^10\)The fractional edge cover of a graph $H = (V_H, E_H)$ is a mapping $ψ : E_H → [0, 1]$ such that for each vertex $a ∈ V_H$, $ε(a) ≥ 1$. The fractional edge-cover number $ρ(H)$ of $H$ is the minimum value of $Σ e ∈ E_H ψ(e)$ among all fractional edge covers $ψ$. 

27
Graphs $G$ with bounded arboricity In [ERS19, ERS20a], 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 [ERR19] presented an improved algorithm for almost uniform sampling of edges in bounded arboricity graphs, in the general graph query model.

References

[ABG+18] 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.

[AKK18] 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.

[BBGM19] 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.

[BHPR+17] Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge estimation with independent set oracles. arXiv preprint arXiv:1711.07567, 2017.

[CLW19] Xi Chen, Amit Levi, and Erik Waingarten. Nearly optimal edge estimation with independent set queries. CoRR, abs/1907.04381, 2019.

[DLM20] Holger Dell, John Lapinskas, and Kitty Meeks. Approximately counting and sampling small witnesses using a colourful decision oracle. In Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms, pages 2201—-2180, 2020.

[ELRS15] 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.

[ER18a] 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.

[ER18b] Talya Eden and Will Rosenbaum. On sampling edges almost uniformly. In 1st Symposium on Simplicity in Algorithms, SODA 2018, January 7-10, 2018, New Orleans, LA, USA, pages 7:1–7:9, 2018.

[ERR19] 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.
A Missing proofs from Section 3

Proof of Claim 3.2. By [NW64], for any graph \( F \), \( \alpha(F) = \max_{F' \subseteq F} \left\lfloor \frac{m_{F'}}{n_{F'}^{\alpha - 1}} \right\rfloor \), where \( n_{F'} \) and \( m_{F'} \) denote the number of vertices and edges in the subgraph \( F' \). Assume towards contradiction that \( H_k \) has arboricity \( \alpha' > \alpha \), and let \( H' \) be a subgraph of \( H_k \) such \( \alpha(H') = \left\lfloor \frac{m_{H'}}{n_{H'}^{\alpha' - 1}} \right\rfloor \). Let \( A \) be the set of all vertices of \( G \) that participate in the nodes of \( H' \). Then the subgraph \( G[A] \) has \( n_{k-1}(G[A]) = n_{H'}(k-1) \)-cliques, and

\[
2n_k(G[A]) \geq 2m_{H'} > \alpha' \cdot n_{H'} > \alpha \cdot n_{k-1}(G[A])
\]
In order to prove Lemma 3.9 we first describe a decomposition of a graph’s vertices into disjoint layers $L_0, \ldots, L_s$. Layers $L_1, \ldots, L_s$ are determined given the zeroth layer $L_0$ and a parameter $\beta \in (0, 1)$. This decomposition is essentially the same as the decomposition of [ERR19], except that here we do not predetermine $L_0$.

**Definition A.1** (Layering of a graph). Let $F = (V_F, E_F)$ be a graph, let $L_0 \subseteq V_F$ be a subset of vertices, and let $\beta \in (0, 1)$ be a parameter. Starting from the given $L_0$, we define an $(L_0, \beta)$-layering of the vertices of $F$ into a series of non-empty disjoint layers $L_0, L_1, \ldots, L_s$, defined iteratively as follows. $L_0$ is as given, and for $j \geq 1$:

$$L_j = \{ v \in V_F : |\Gamma(v) \cap (L_0 \cup \ldots \cup L_{j-1})| > (1 - \beta)|\Gamma(v)| \} .$$

That is, $L_j$ is the set of vertices for which $(1 - \beta)$ of their neighbors reside in $L_0 \cup \ldots \cup L_{j-1}$. We say that a graph $F = (V_F, E_F)$ admits an $(L_0, \beta)$-layering of depth $s$ if $V_F = L_0 \cup \ldots \cup L_s$.

**Notation A.2.** For a graph $F$ and a decomposition as above, let $E_F(L_j)$ denote the set of oriented edges incident to the vertices of $L_j$, $E_F(L_j) = \{(v, u) \mid v \in L_j \}$. We sometimes omit the subscript $F$ when the graph at question is clear from context.

We claim that for any choice of parameter $\beta$ and an appropriate choice of the layer $L_0$, the decomposition in the above definition has depth $s = \lceil \log |V_F| \rceil$.

**Claim A.3.** Let $F$ be a graph of arboricity $\alpha$, and let $\beta \in (0, 1)$ be a parameter. Suppose that for $\tau$ that satisfies $\tau \cdot \beta \geq 4\alpha$, we have that for every $v$ such that $d_F(v) \leq \tau$ it holds that $v \in L_0$. Then the graph $F$ admits an $(L_0, \beta)$-layering of depth $s = \lceil \log |V_F| \rceil$ as defined in Definition A.1.

**Proof.** For each $i$, let $W_i = V_F \setminus (L_0 \cup L_1 \cup \ldots \cup L_{i-1})$ be the set of vertices not in levels $0, 1, \ldots, i-1$. Let $m(W_i)$ denote the number of edges in the subgraph of $F$ induced by $W_i$. For any fixed $i$ and $v \in W_{i+1}$, we have $d_{\leq i}(v) < (1 - \beta)d(v)$ because $v \notin L_{\leq i}$. Therefore, $v$ has at least $\beta d(v) > \beta \tau$ neighbors in $W_i$ (recall that if $v \notin L_0$ then by the promise that $L_0$ contains all vertices with degree at most $\tau$, $d(v) \geq \tau$). Summing over vertices $v \in W_{i+1}$ gives

$$m(W_i) = \frac{1}{2} \sum_{v \in W_i} d_{\geq i}(v) \geq \frac{1}{2} \sum_{v \in W_{i+1}} d_{\geq i}(v) > \frac{1}{2} |W_{i+1}| \cdot \beta \tau .$$

On the other hand, since $G$ has arboricity at most $\alpha$, Theorem 3.11 implies that $m(W_i) \leq \alpha |W_i|$. Combining the above upper bound with Equation (4), it follows that $\frac{|W_{i+1}|}{|W_i|} \leq \frac{2\alpha}{\beta \tau} \leq \frac{1}{2}$, where the last inequality is by the assumption that of $\beta \cdot \tau \geq 4\alpha$. Therefore, $s \leq \lceil \log |V_F| \rceil$, as required.

We shall use the following notation in the proof of Lemma 3.9.

**Notation A.4.** For a graph $F = (V_F, E_F)$ and an edge $e \in E_F$, let $P_j[e]$ denote the probability that Sample-Edge returns $e$ when the chosen index in Step 3 is $j$. Further let $P_{\leq j}[e] \overset{\text{def}}{=} \sum_{i=0}^j P_i[e]$ and similarly for $P_{\geq j}[e]$.
Proof of Lemma 3.9. By the assumption that Define-$L_0$ is a $(\delta, \tau)$-good $L_0$-oracle for $\delta \leq \beta'/X$, it holds that with probability at least $1 - \delta \geq 1 - \beta'/X$, Define-$L_0$ determines a $\tau$-good $L_0$, (see Definition 3.5). Denote this event by $\mathcal{E}$, and condition on it holding.

To prove the claim we shall prove the following two items.

- For every $j \in [s]$, $e \in E_F(L_j)$ and $\ell \in [j, s]$, it holds that $P_{\leq \ell}[e] \geq \frac{(1 - \beta)^{2\ell + 1}}{X}$.

- For every $e \in E_F$, $P_{\leq s}[e] \leq \frac{(1 + \beta)^{s + 1}}{X}$.

We start with the first item which we prove by induction on $j$, starting with $j = 0$. Let $e = (u, u')$ be an edge in $E_F(L_0)$ (so that $u \in L_0(F)$). By the assumption that Sample-$E(L_0)$ is a good $L_0$-sampling subroutine for $F$, it holds that the invocation Sample-$E(L_0)(F, N_F, \beta', \tau, \vec{p})$ in Step 4 returns $e$ with probability $\frac{1 + \beta}{X}$. Hence,

$$P_0[e] \geq \frac{1 - \beta'}{X}. \tag{5}$$

Hence, for $j = 0$ and $0 < \ell \leq s$,

$$P_{\leq \ell}[e] = \sum_{i=0}^{\ell} P_i[e] = P_0[e] + \sum_{i=1}^{\ell} P_i[e] \geq \frac{(1 - \beta')}{X}. \tag{6}$$

We now assume that the first item in the claim holds for all $i \leq j - 1$ and $\ell \in [i, s]$, and prove that it holds for $j$ and for every $\ell \in [j, s]$. By the induction hypothesis, for every $i \leq j - 1$, $(v, u) \in E_F(V_i)$, $P_{\leq j-1}[v, u] \geq \frac{(1 - \beta')^{i+2}}{X}$. Moreover, by the assumption that Sample-a-Neighbor is a $\beta'$-good neighbor-sampling subroutine with respect to $F$, for every $u' \in \Gamma_F(u)$, the probability that $u'$ is returned when Sample-a-Neighbor is invoked with $u$ is at least $\frac{1 - \beta'}{d(u)}$. Therefore, for every $e = (u, u') \in E(V_j)$,

$$P_{\leq \ell}[e] \geq P_{\leq j}[e] = \sum_{(v, u) \in E_F} P_{\leq j-1}[(v, u)] \cdot \Pr[\text{Sample-a-Neighbor returns } u']$$

$$\geq \sum_{(v, u) \in E_F} P_{\leq j-1}[(v, u)] \cdot \frac{1 - \beta'}{d(u)} \geq \sum_{i=0}^{j-1} \sum_{(v, u) \in E_F(V_i)} P_{\leq j-1}[(v, u)] \cdot \frac{1 - \beta'}{d(u)}$$

$$\geq \sum_{i=0}^{j-1} \sum_{(v, u) \in E_F(V_i)} \frac{(1 - \beta')^{2i+1}}{X} \cdot \frac{1 - \beta'}{d(u)} \geq \frac{(1 - \beta')^{2j-1} \cdot d_{\leq j-1}(u)}{X} \cdot \frac{1 - \beta'}{d(u)}$$

$$\geq \frac{(1 - \beta')^{2j+1}}{X},$$

where we used the decomposition property of Definition A.1, that $d_{\leq j-1}(u) > (1 - \beta')d(u)$. Hence, the first item of the claim holds for every $j \in [s]$ for every $\ell \in [j, s]$.

We continue to prove the second item. We prove by induction on $j$ that for every $j \in [s]$, $P_{\leq j}[e] \leq \frac{(1 + \beta)^{j+1}}{X}$. First for $e \in E_F(L_0)$, by the assumption that Sample-$E(L_0)$ is a good $L_0$-sampling subroutine, it holds that $P_0[e] \leq \frac{1 + \beta}{X}$. Furthermore, by the condition on $\mathcal{E}$, Sample-$E(L_0)$ is a $\tau$-good $L_0$-sampling subroutine, only edges in $E_F(L_0)$ are returned in Step 4. Hence, for every
\( e \notin E_F(L_0) \), \( P_j[e] = 0 \). Therefore, the claim holds for every \( e \) and \( j = 0 \). We now assume that for every \( e \), for every \( i \leq j-1 \), \( P_{\leq i}[e] \leq \frac{(1+\beta')^{i+1}}{X} \), and prove it for \( j \). First, for \( e \in L_0 \), due to Step 5a, \( e \) can only be returned if \( \text{Sample-Edge} \) is invoked with \( j = 0 \), in which case, \( e \) is returned with probability at most \( \frac{1+\beta'}{X} \). Hence,

\[
P_{\leq j}[e] = \sum_{i=0}^{j} P_j[e] = P_0[e] \leq \frac{1+\beta'}{X}.
\]

Now consider an edge \( e = (u, u') \notin E_F(L_0) \). We deal separately with \( P_0[e] \) and \( P_{\leq j}[e] - P_0[e] \). By the assumption that \( \text{Sample-E}(L_0) \) is a \((\beta', \tau)\)-good \( L_0 \)-sampler, \( P_0[e] = \frac{\beta'}{X} \). Also, by the assumption that \( \text{Sample-a-Neighbor} \) is a \( \beta' \)-good neighbor-sampling subroutine, if \( \text{Sample-a-Neighbor} \) is invoked with a node \( u \in V_F \), then for every \( u' \in \Gamma_F(u) \), the probability that it is returned by the subroutine is at most \( \frac{1+\beta'}{d(u)} \). Hence,

\[
P_{\leq j}[e] = \frac{\beta'}{X} + \sum_{i=1}^{j} P_i[e] \leq \sum_{i=1}^{j} \sum_{(v, u) \in E_F} P_{i-1}[(v, u)] \cdot \frac{1+\beta'}{d(u)} = \sum_{(v, u) \in E_F} \frac{1+\beta'}{d(u)} \sum_{i=0}^{j-1} P_i[(v, u)]
\]

\[
= \sum_{(v, u) \in E_F} \frac{1+\beta'}{d(u)} \cdot P_{j-1}[(v, u)] \leq \frac{(1+\beta')^j}{d(u)} \cdot \frac{1+\beta'}{X} \cdot d(u) = \frac{(1+\beta')^j+1}{X}.
\]

Therefore, \( P_{\leq j}[e] \leq \frac{(1+\beta')^{j+1}}{X} \). This concludes the proof of the second item.

For every edge \( e \in F \), let \( P_i[e \mid \mathcal{E}] (P_i[e \mid \overline{\mathcal{E}}]) \) denote the events that the edge \( e \) is returned in the \( i \)-th invocation, conditioned on the event \( \mathcal{E} (\overline{\mathcal{E}}) \). Then

\[
P_{\leq s}[e] = P_{\leq s}[e \mid \mathcal{E}] \cdot \Pr[\mathcal{E}] + P_{\leq s}[e \mid \overline{\mathcal{E}}] \cdot \Pr[\overline{\mathcal{E}}].
\]

Hence, for \( \delta = \beta'/X \), \( P_{\leq s}[e] \geq (1-\delta) \frac{(1-\beta')^{2s+1}}{X} \geq \frac{(1-\beta')^{2s+2}}{X} \) and \( P_{\leq s}[e] \leq \frac{(1+\beta')^{s+1} + \delta}{X} \leq \frac{(1+\beta')^{s+2}}{X} \).

Therefore, for every edge \( e \in F \), \( P_{\leq s}[e] \in \left[ \frac{(1-\beta')^{2s+1}}{X}, \frac{(1+\beta')^{s+2}}{X} \right] \). Consider a specific edge \( e \in F \), and assume that \( v \in L_i \) for some \( i \in [0, \ldots, s] \), \( s = \lfloor \log N_F \rfloor \). By the above,

\[
\Pr[e \text{ is returned}] = \sum_{i=0}^{s} \Pr[j = i] \cdot \Pr[e \text{ is returned} \mid j = i]
\]

\[
= \sum_{i=0}^{s} \frac{1}{s+1} \cdot P_i[e] = \frac{P_{\leq s}[e]}{s+1} \in \left[ \frac{(1-\beta')^{2s+1}}{X}, \frac{1+\beta'}{s+1} \right] \cdot \frac{1}{X}.
\]

Therefore, by the setting of \( \beta' = \frac{\beta}{(2s+2)} \), for every edge \( e \), \( \Pr[e \text{ is returned}] \in \frac{1+\beta}{X \cdot (s+1)} \), as claimed. \( \square \)

**Proof of Claim 3.12.** By the Taylor expansion of the function \( (1-x)^y \), it holds that

\[
1 - x[y] \leq (1-x)[y] \leq 1 - x[y] + (x[y])^2.
\]

Therefore,

\[
1 - (1-x)[y] \leq x[y] \leq x(y+1) = xy(1+1/y).
\]

And

\[
1 - (1-x)[y] \geq x[y] - x^2(y)[y] - 1 \geq xy - 2x^2y^2 = xy(1-2xy).
\]

\( \square \)