On Approximating the Number of k-cliques in Sublinear Time

Talya Eden * Dana Ron † C. Seshadhri ‡

Abstract

We study the problem of approximating the number of k-cliques in a graph when given query access to the graph. We consider the standard query model for general graphs via (1) degree queries, (2) neighbor queries and (3) pair queries. Let n denote the number of vertices in the graph, m the number of edges, and C_k the number of k-cliques. We design an algorithm that outputs a $(1+\epsilon)$-approximation (with high probability) for C_k, whose expected query complexity and running time are $O\left(\frac{n}{C_k^{1/k}} + \frac{m^{k/2}}{C_k}\right)\text{poly}(\log n, 1/\epsilon, k)$. Hence, the complexity of the algorithm is sublinear in the size of the graph for $C_k = \omega(m^{k/2-1})$. Furthermore, we prove a lower bound showing that the query complexity of our algorithm is essentially optimal (up to the dependence on $\log n$, $1/\epsilon$ and k).

The previous results in this vein are by Feige (SICOMP 06) and Goldreich-Ron (RSA 08) for edge counting ($k = 2$) and by Eden-Levi-Ron-Seshadhri (FOCS 2015) for triangle counting ($k = 3$). Our result matches the complexities of these results.

The heart of our algorithm is a procedure for sampling each k-clique incident to a given set S of vertices with approximately equal probability. This is done by first sampling an edge (u, v) incident to S uniformly at random, and then trying to extend the edge to a k-clique. We prove that by considering only v's higher-degree neighbors, we can sample a k-clique incident to S almost uniformly, and show that each such neighbor can be sampled with almost uniform and sufficiently high probability.

*Tel Aviv University, *talyaa01@gmail.com*. This research was partially supported by a grant from the Blavatnik fund. The author is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship.

†Tel Aviv University, *danaron@tau.ac.il*. This research was partially supported by the Israel Science Foundation grant No. 671/13 and by a grant from the Blavatnik fund.

‡University of California, Santa Cruz, *sesh@ucsc.edu*

1 Introduction

Counting the number of k-cliques in a graph is a classic problem in theoretical computer science, and the special case of $k = 3$ (triangle counting) is itself an important problem. In practice, clique counting has received much attention due to its significance for analyzing real-world graphs [26, 11, 39, 14, 34, 6, 4, 22, 5, 42, 27, 19, 47, 21, 31]. From a theoretical standpoint, the best exact algorithms use matrix multiplication [35, 18]. Better bounds for sparse graphs can be obtained by combinatorial methods [9, 50].

There is a long line of algorithms for approximately counting the number of cliques (especially triangles) in various computational models, including distributed and streaming settings [9, 41, 40, 28, 46, 48, 3, 30, 10, 44, 49, 2, 29, 43, 45]. All these algorithms begin by reading their entire input graph, and hence must run in at least linear time. Recently, Eden et al. [15] gave the first sublinear-time algorithm for triangle counting. The query model used is the standard model for sublinear algorithms on general graphs. We assume that the vertex set is $V = [n]$. Algorithms can make the following queries. (1) Degree queries: given $v \in V$, get the degree $d(v)$. (2) Neighbor queries, given $v \in V$ and $i \leq d(v)$ get the i^{th} neighbor of v. (3) Pair queries: given vertices u, v, determine if (u, v) is an edge.

We show that there is a sublinear-time algorithm for approximating the number of k-cliques in this model, for any given k, subsuming the result of [15] for $k = 3$.

1.1 Results

Let $G = (V, E)$ be a graph over n vertices and m edges, where we view edges as ordered pairs, so that m equals the sum of the degrees of all vertices. Let C_k denote that number of cliques of size k in G.

Our main theorem follows.

Theorem 1. *There exists an algorithm that, given n, k, an approximation parameter $0 < \epsilon < 1$, and query access to a graph G, outputs an estimate $\widehat{C}_k$, such that with high constant probability (over the randomness of the algorithm),*

$$(1 - \epsilon) \cdot C_k \leq \widehat{C}_k \leq (1 + \epsilon) \cdot C_k.$$

The expected query complexity of the algorithm is

$$O\left(\frac{n}{C_k^{1/k}} + \min\left\{m, \frac{m^{k/2}}{C_k}\right\}\right) \cdot \text{poly}(\log n, 1/\epsilon, k) \ ,$$

and the expected running time is $O\left(\frac{n}{C_k^{1/k}} + \frac{m^{k/2}}{C_k}\right) \cdot \text{poly}(\log n, 1/\epsilon, k)$.

We show that this result is almost optimal by presenting a nearly matching lower bound.

Theorem 2. *The query complexity of any multiplicative-approximation algorithm for C_k is*

$$\Omega\left(\frac{n}{C_k^{1/k}} + \min\left\{m, \frac{m^{k/2}}{C_k \cdot (c \cdot k)^k}\right\}\right) \ ,$$

for a (small) constant c.

Hence, our algorithm has sublinear complexity when the number of k-cliques in the graph is above a certain threshold, and there is no sublinear approximation algorithm for the problem if this condition does not hold.

A high-level discussion of the bound in Theorem 1: The complexity stated in Theorem 1 is a sum of two terms. For any $t \geq k$ consider a graph G consisting of a path of length $n - t$ and a clique of size t, so that $C_k = \Theta(t^k)$. The first term, $n/C_k^{1/k} = \Theta(n/t)$, is the expected number of random vertices required to hit the k-clique. Turning to the second term, there is a (simple) combinatorial $O(n + m^{k/2})$ algorithm for enumerating k-cliques [9]. The second term in our bound is $m^{k/2}/C_k$, essentially dividing the complexity of [9] by the number of k-cliques.

1.2 Main ideas and techniques

In all that follows when we refer to a random vertex we mean a vertex selected uniformly at random. Our algorithm for approximating the number of k-cliques builds on a few ideas of Eden et al. [15] (henceforth ELRS) for approximately counting the number of triangles (i.e., $k = 3$). But the heart of our algorithm, a process for sampling k-cliques, is conceptually and technically different. We begin with a high-level description of our application of the ELRS basic framework, after which we explain at what point the triangle counting ideas do not work for $k > 3$. We then turn to describe our approach, which is based on (almost-)uniform sampling of k-cliques that are incident to a subset of vertices. For the sake of simplicity, assume ϵ is a constant. It is convenient to assume that constant factor estimates of C_k and m are known.[1]

The ELRS framework: vertex sampling and clique assignment. Our algorithm starts by uniformly and independently selecting a (multi-)set S of vertices of size roughly $n/C_k^{1/k}$. Denoting the number of k-cliques incident to a vertex u by $c_k(u)$, a natural estimate for C_k is $\frac{n}{k|S|}\sum_{u\in S} c_k(u)$. Unfortunately, for a random u, $c_k(u)$ can have extremely large variance. ELRS reduce the variance by considering triangles only from endpoints u where $c_3(u)$ is not too large. In our setting, we formalize this by defining "sociable" vertices. A vertex is sociable if it participates in a number of k-cliques that is above a certain threshold τ or if its degree is above another threshold τ'. A k-clique with vertices $\{u_1, \ldots, u_k\}$ is *assigned* to the smallest degree vertex u_j that is not sociable. We set the parameters τ, τ' to ensure that the number of k-cliques that are not assigned to any vertex is at most $\epsilon \cdot C_k$. Let $\alpha(u)$ be the number of k-cliques assigned to u. We can prove that if $|S|$ is roughly $n/C_k^{1/k}$, then $\frac{n}{|S|}\sum_{u\in S} \alpha(u)$ is a $(1+\epsilon)$-approximation of C_k with high probability. The problem of approximating C_k is now reduced to approximating $\alpha(S) = \sum_{u\in S} \alpha(u)$.

Where the ELRS approach fails to generalize. ELRS tackle the task of approximating $\alpha(S)$ as follows. First, they "transfer" the assignment of triangles from vertices to edges. Letting $\alpha(u, v)$ denote the number of triangles assigned to the edge (u, v) we have that $\alpha(S) = \sum_{(u,v)\in E(S)} \alpha(u, v)$ where $E(S)$ is the (multi-)set of edges incident to the vertices of S. ELRS approximate the average value of $\alpha(u, v)$ (taken over edges in $E(S)$) by uniformly sampling edges from $E(S)$ and approximating the number of triangles that are assigned to the sampled edges. For an edge $(u, v) \in E(S)$, let u' be the vertex among u, v with lower degree, and assume for simplicity that $u' = u$. Suppose we sample a random neighbor w of u, check if the triple (u, v, w) forms a triangle and if so, decide whether it is assigned to (u, v). (We put aside for now the question of how the latter task is

[1] The assumption on m can be removed by applying [23], and the assumption on C_k can be removed by a geometric search (for details see subsection 3.7).

performed.) While the probability that a triangle assigned to (u, v) is observed increases linearly with $\alpha(u, v)$, it also decreases with $1/d(u)$, thus creating a degree-dependent bias. ELRS overcome this bias by repeating the above procedure with probability proportional to $d(u)$. They show that this sampling dependency on the degree of the vertices can be amortized to yield the final desired query complexity. Unfortunately, for general k, the number of samples scales as $d(u)^{k-2}$ and the sampling bound amortization fails for $k > 3$.

Sampling k-cliques incident to S almost uniformly. We now describe our approach for approximating $\alpha(S)$. Let $\mathcal{A}(S)$ denote the (multi-)set of k-cliques assigned to vertices in S. We estimate $\alpha(S) = |\mathcal{A}(S)|$ by sampling k-cliques incident to S, and checking whether they belong to $\mathcal{A}(S)$. The key is to be able to sample each k-clique incident to S (and in particular in $\mathcal{A}(S)$) with (approximately) *the same probability*. Specifically, as explained next, we'll do so with probability proportional to $\frac{1}{m(S)\cdot m^{k/2}}$, where $m(S) = |E(S)|$.

To this end, each k-clique that contains a vertex $u \in S$ is associated with one of the edges incident to u in the clique. For reasons that will become clear shortly, the other endpoint of this edge is selected to be the lowest degree vertex among the vertices in the clique (excluding u, and breaking ties arbitrarily but consistently). The procedure for sampling k-cliques starts by sampling an edge (u, v) uniformly in $E(S)$. The procedure next attempts to extend this single edge to a k-clique, and furthermore, to a clique in which all other $k-2$ vertices have degree higher than v. This is done in one of two different ways, depending on the degree of v, where in either way, the algorithm may fail to output any k-clique. For the sake of simplicity of the introduction, in what follows we say that a vertex is a low-degree vertex if its degree is at most $\sqrt{m}$ (in the technical part of the paper we use a slightly different definition). Otherwise we say it is a high-degree vertex.

The "low case": If v is a low-degree vertex, then we can basically sample $k-2$ random neighbors of v and check whether we obtained a k-clique in which v is the lowest degree vertex other than u. In order to ensure that all k-cliques incident to S are output with the same probability, we apply rejection sampling and keep each sampled neighbor w of v with probability $d(v)/\sqrt{m}$ (and conditioned on $d(v) \leq d(w)$). Hence, each such clique (i.e., in which the lowest degree vertex other than $u \in S$ is a low- degree vertex) is output with equal probability $\frac{(k-2)!}{m(S)\cdot\sqrt{m}^{k-2}}$. (The $(k-2)!$ factor is due to the fact that we may obtain the $k-2$ vertices in the clique other than u and v in any order.)

The "high case": The challenging case is when v is a high-degree vertex (which is related to the difficulty encountered when attempting to extend the ELRS algorithm). In this case we can no longer perform rejection sampling as described in the low case, unless we increase the rejection probability (which in turn will require to increase the query complexity). However, observe that we are interested only in sampling neighbors of v with degree higher than v and that the number of vertices with high degree is at most $\sqrt{m}$. Therefore, if we had a way to efficiently sample each high-degree vertex with probability (approximately) $1/\sqrt{m}$, we would obtain the same probability over cliques incident to S as in the low case. We next show how this can be done.

Consider selecting a random multi-set T of roughly $t = \frac{n}{\log n/\sqrt{m}}$ vertices. The setting of t is such that with high probability, for every high-degree vertex w, the number of neighbors that w has in T is close to its expected value, that is, $d(w) \cdot \frac{t}{n}$. This implies that if we select an edge (x, y) uniformly at random in $E(T)$ (the (ordered) edges incident to T, whose number is $m(T)$), then the probability that $y = w$ for a fixed high-degree vertex w, is approximately $\frac{d(w)\cdot(t/n)}{m(T)}$.

Assume that $m(T)$ is not much larger than its expected value, $m \cdot \frac{t}{n}$ (which can be ensured with high probability). Let $p(w) = \frac{m(T)}{d(w)\cdot(t/n)\cdot\sqrt{m}}$, so that under this assumption on $m(T)$, and the fact

that w is a high-degree vertex, $p \in (0,1]$. If we now keep w with probability $p(w)$, then we have a subroutine that samples each high-degree vertex with approximately equal probability $1/\sqrt{m}$. This in turn implies that we can select any fixed subset of $k-2$ high-degree vertices with probability very close to $\frac{(k-2)!}{\sqrt{m}^{k-2}}$. We then check whether we obtained a clique (and that the clique is associated with (u,v)).

We now have a procedure that outputs each clique incident to S with (roughly) the same probability, $\frac{(k-2)!}{m(S)\cdot\sqrt{m}^{k-2}}$. In the next paragraph, we discuss a procedure for deciding whether a k-clique belongs to $\mathcal{A}(S)$. Given this decision procedure, we can estimate $|\mathcal{A}(S)|$ by performing $\frac{m(S)\cdot(2\sqrt{m})^{k-2}}{(k-2)!\cdot|\mathcal{A}(S)|} = O\left(\frac{m^{k/2}}{C_k}\right)$ calls to the k-clique sampling procedure. (We assume that $|\mathcal{A}(S)|$ is close to its expected value, $C_k \cdot \frac{s}{n}$ and that $m(S)$ is not much larger than its expected value, $m \cdot \frac{s}{n}$).

Deciding whether a k-clique is in $\mathcal{A}(S)$. Deciding whether a k-clique incident to a vertex $u \in S$ should be assigned to u, requires to determine which of the clique vertices are sociable. Recall that a vertex is considered sociable if the number of k-cliques it participates in is more than τ or if its degree is above τ' (for appropriate settings of τ and τ'). The second condition can be easily verified by a single degree query. As for the first condition, given a vertex u, we verify whether the number of k-cliques that it participates in is more that τ by running the k-clique sampling procedure with $S = \{u\}$ a sufficient number of times (roughly $\frac{d(u)\cdot(2\sqrt{m})^{k-2}}{(k-2)!\cdot\tau} \leq \frac{\tau'\cdot(2\sqrt{m})^{k-2}}{(k-2)!\cdot\tau}$ where $\frac{\tau'}{\tau} = O\left(\frac{m}{C_k}\right)$). The procedure may err on "almost sociable" vertices, but the analysis can be modified to deal with this. While the procedure has high query complexity, it is only invoked when the clique-sampling procedure returns a clique. The frequency of the latter can be bounded appropriately to get the final query complexity.

The lower bound. The lower bound is obtained by building on the lower-bound construction of ELRS for triangles ($k = 3$). Roughly speaking, we reduce the problem of approximating the number of k-cliques for $k > 3$ to the problem of approximating the number of triangles (where the dependence on C_k for the graph in question is translated to a dependence on C_3 for a different graph).

1.3 Related work

A significant portion of the work on clique counting focuses on triangle counting. Because our focus is on general k, we avoid a detailed discussion of results for triangle counting. We point the interested reader to [15].

Nešetřil and Poljak give the first non-trivial algorithm for k-clique counting by reducing to matrix multiplication [35]. Specifically, their algorithm runs in $O(n^{\omega\lfloor k/3\rfloor+k(\mathrm{mod}3)})$. Eisenbrand and Grandoni refine this bound for certain values of k by careful reductions to rectangular matrix multiplication [18]. They also give better dependencies on m for sparse graphs. The general dependence of the form $n^{\omega k/3}$ is believed to be optimal. Recent work by Abboud et al. builds on this conjecture to prove hardness for various parsing algorithms [1]. More relevant to our work, Chiba and Nishizeki give an algorithm for k-clique enumeration, based on the arboricity of the graph, from which the $O(n + m^{k/2})$ bound for general graphs follows immediately. These ideas have also been used in practical algorithms for clique counting on distributed systems [21].

In the context of sublinear algorithms, our work follows a line of results on sublinear estimation of subgraph counts. Our analysis builds on several techniques developed in these results. The starting point is the average degree estimation results of Feige [20] and Goldreich and Ron [23].

Gonen et al. greatly generalize these techniques to estimate the count of k-stars in sublinear time [24]. Eden et al. [16] further extended and simplified all these results, and show connections between this problem and the graph degeneracy. They also build on the basic ELRS framework. Eden and Rosenbaum [17] provide an algorithm for sampling edges almost uniformly, and our clique sampler uses some of their ideas to sample high-degree vertices.

Other work on sublinear algorithms for estimating graph parameters include results on the minimum weight spanning tree [7, 13, 12], maximum matching [36, 53] and minimum vertex cover [38, 36, 32, 53, 25, 37].

2 Preliminaries

We consider simple undirected graphs over a set V of n vertices. It is convenient to think of the graph edges as ordered pairs, so that every edge is considered from both endpoints. We say that the ordered edge (u, v) originates from the vertex u. We denote the set of all ordered edges by E and let $m \triangleq |E|$. We use the following notations.

- $d(u)$: the degree of a vertex u (the number of edges originating from u). Note that $\sum_{u \in V} d(u) = m$.
- $E(S)$, $m(S)$: $E(S) \triangleq \{(u, v) \mid u \in S\}$ and $m(S) \triangleq |E(S)| = \sum_{u \in S} d(u)$.
- $d_S(u)$: for any vertex u and set of vertices S, $d_S(u)$ is the number of neighbors of u in S.
- C_k, $c_k(u)$: C_k is the number of k-cliques in the given graph. For $u \in V$, $c_k(u)$ is the number of k-cliques that u participates in. Note that $C_k = \frac{1}{k} \cdot \sum_{u \in V} c_k(u)$.

We use $\prec$ to denote a total order over the graph vertices such that for every two vertices u and v, if $d(u) < d(v)$, then $u \prec v$, and if $d(u) = d(v)$, then the order between u and v is determined in an arbitrary but fixed manner (e.g., by vertex id).

Let $[r] \triangleq \{1, \ldots, r\}$ and let $(1 \pm \alpha)^t \cdot x$ denote the interval $\left[(1-\alpha)^t \cdot x, (1+\alpha)^t \cdot x\right]$.

We make use of the following version of Chernoff's inequality [8]. Let χ_i for $i = 1, \ldots, m$ be a random variables taking values in $[0, B]$, such that for every i, $\mathrm{Exp}[\chi_i] = \mu$. Then

$$\Pr\left[\frac{1}{m}\sum_{i=1}^{m}\chi_i > (1+\gamma)p\right] < \exp\left(-\frac{\gamma^2 pm}{3B}\right) \quad \text{and} \quad \Pr\left[\frac{1}{m}\sum_{i=1}^{m}\chi_i < (1-\gamma)p\right] < \exp\left(-\frac{\gamma^2 pm}{2B}\right) .$$

The proof of the following claim is similar to the proof of [9] for their exact clique enumeration algorithm (and we include it here for the sake of completeness).

Claim 3. *For every graph G with m (ordered) edges and C_k k-cliques, $C_k \leq m \cdot \binom{\sqrt{m}}{k-2}$.*

Proof. Let D be the DAG obtained by orienting edges in G according to $\prec$. Let $d^+(v)$ be the out-degree of vertex v in D. Observe that $\max_v\{d^+(v)\} \leq \sqrt{m}$. (All $d^+(v)$ out-neighbors of v have degree at least $d(v) \geq d^+(v)$. Thus, $d^+(v) \leq \sqrt{m}$.) The number of k-cliques where v is the lowest vertex according to $\prec$ is at most $\binom{d^+(v)}{k-1}$. Thus, $C_k \leq \sum_v \binom{d^+(v)}{k-1} \leq \binom{\sqrt{m}}{k-2} \sum_v d^+(v) = m\binom{\sqrt{m}}{k-2}$. □

3 The algorithm

The main algorithm for approximating the number of k-cliques is presented in Subsection 3.2 and named Approximate-cliques. It takes the following parameters.

- $\overline{m}$: This is assumed to be a fairly precise estimate of the number of (ordered) edges m, and can be obtained using [23] (in expected time $O\left(\frac{n}{\sqrt{m}}\right) \cdot \mathrm{poly}(\log n, 1/\epsilon)$).

- $\overline{C}_k$: This is assumed to be a constant-factor estimate of C_k, which is obtained by geometric search (as shown in Subsection 3.7).
- ϵ: The main approximation parameter. We set $\overline{\epsilon} = \epsilon/5$.
- δ: The failure parameter. We set $\overline{\delta} = \delta/4$.

3.1 Sociable vertices and the assignment of cliques to vertices

The notion of *sociable* vertices, defined next, is critical in reducing the variance of the output of our algorithm.

Definition 4 (Sociable and shy vertices)**.** *We say that a vertex u is* sociable *if $c_k(u) > k \cdot (50\overline{C}_k)^{1-1/k}/\overline{\epsilon}^{1/k}$ or if $d(u) > 4\overline{m}/(\overline{\epsilon}\overline{C}_k)^{1/k}$. If $c_k(u) \leq \frac{1}{4}k \cdot (50\overline{C}_k)^{1-1/k}/\overline{\epsilon}^{1/k}$ and $d(u) \leq 4\overline{m}/(\overline{\epsilon}\overline{C}_k)^{1/k}$, then we say that u is* shy*.*

Note that a vertex u may be neither sociable nor shy. This is the case if $d(u) \leq 4\overline{m}/(\overline{\epsilon}\overline{C}_k)^{1/k}$ and $\frac{1}{4}k \cdot (50\overline{C}_k)^{1-1/k}/\overline{\epsilon}^{1/k} < c_k(u) \leq k \cdot (50\overline{C}_k)^{1-1/k}/\overline{\epsilon}^{1/k}$.

The following claim, whose proof follows directly from Definition 4, shows that we can ignore cliques that do not contain shy vertices.

Claim 5. *If $\overline{m} \in [(1-\overline{\epsilon})m, m]$ and $\overline{C}_k > C_k/4$, then at most $\overline{\epsilon}C_k$ k-cliques consist solely of vertices that are not shy.*

Definition 6 (An appropriate partition)**.** *We say that a partition $P = (V_0, V_1)$ of V is* appropriate *if every shy vertex (as defined in Definition 4) is in V_0 and every sociable vertex is in V_1 (and any other vertex can be either in V_0 or V_1).*

We next specify the assignment of cliques to vertices.

Definition 7 (Assigning cliques)**.** *Fix a partition $P = (V_0, V_1)$.*

- *Assignment of cliques: We* assign *each k-clique $K = \{u_1, \ldots, u_k\}$ to the vertex u_i that is the first (according to $\prec$) vertex of K in V_0. If all of K's vertices are in V_1, then K is* not assigned *to any vertex.*
- *$\alpha_P(u), \alpha_P(S)$: We denote the number of k-cliques assigned to u (for this P) by $\alpha_P(u)$. For a set S of vertices, $\alpha_P(S) = \sum_{u \in S} \alpha_P(u)$.*

The following is a corollary of Claim 5, Definition 6 and Definition 7.

Corollary 8. *For every partition $P = (V_0, V_1)$ of V it holds that $\alpha_P(V) \leq C_k$. Furthermore, if $P = (V_0, V_1)$ is appropriate, $\overline{m} \geq (1-\epsilon)m$ and $\overline{C}_k \geq C_k/4$, then $\alpha_P(V) \in [(1-\overline{\epsilon})C_k, C_k]$.*

Another distinction between types of vertices that will play a central role in our analysis is the following.

Definition 9 (High-degree and low-degree vertices)**.** *We say that a vertex u is a* high-degree *vertex if $d(u) > 2\sqrt{\overline{m}}$ and otherwise we say it is a* low-degree *vertex.*

3.2 The main algorithm and the procedures it uses

In this subsection we present our main algorithm and the corresponding main theorem. Our algorithm invokes several procedures, which are provided in the following subsections. Here we shortly describe all procedures and state the main claim regarding each of them. Building on these claims we give a proof sketch of the main theorem (the complete proof appears in Subsection 3.6).

Approximate-cliques. This is the main algorithm, and it is provided in Figure 1. The algorithm begins by constructing two random multisets, S and T. The multiset S is obtained by simply selecting vertices uniformly (independently) at random. The multiset T is constructed by a procedure Sample-degrees-typical. We show that with high probability, S and T have certain desired properties (where the correctness of subsequent steps of the algorithm relies on these properties).

In Step 5, the algorithm calls two procedures: Sample-a-clique and Is-sociable. The heart of the algorithm is the procedure Sample-a-clique that either returns a k-clique $\{u, v, w_1, \ldots, w_{k-2}\}$ where $u \in S$ or returns `fail`. The procedure Is-sociable distinguishes between sociable and shy vertices (as defined in Definition 4). It is used in order to decide for each k-clique that is output in the previous step, whether it is assigned to u (as defined in Definition 7).

Note that if the sample size s (defined in Step 2) is larger than n, then the algorithm can simply set $S = V$. Similarly, if q (the number of iterations in Step 5) is larger than $\overline{m}$, then the algorithm can query upfront all edges incident to S and their neighbors so that it never performs more than $\min\{m, \overline{m}\}$ queries. (If it views more than $\overline{m}$ edges, then it can abort.) Finally, we may assume that $\epsilon > 1/\overline{m}^{k/2}$, since otherwise we are required to output the exact number of k-cliques in the graph (recall that by Claim 3, $C_k < m^{k/2}$), and thus can simply invoke the exact enumeration algorithm of [9].

Approximate-cliques $(n, k, \overline{m}, \overline{C}_k, \epsilon, \delta)$

1. Let $\overline{\epsilon} = \epsilon/5$ and $\overline{\delta} = \delta/4$.
2. Let S be a multiset of $s = \frac{700 \cdot k \cdot n \cdot \ln(1/\overline{\delta})}{\overline{\epsilon}^{2+1/k} \cdot \overline{C}_k^{1/k}}$ vertices chosen uniformly at random.
3. Query the degree of each vertex in S and set up a data structure $D(S)$ that supports sampling a uniform edge in $E(S)$ in constant time.
4. Invoke Sample-degrees-typical$(n, k, \overline{m}, \overline{C}_k, \overline{\epsilon}, \overline{\delta})$. If the procedure returned `fail`, then **return** `fail`. Otherwise, let $(T, m(T), D(T))$ be its output.
5. For $i = 1$ to $q = \frac{m(S) \cdot (2\sqrt{\overline{m}})^{k-2}}{(1-\overline{\epsilon})^3 \cdot (k-2)! \cdot \overline{C}_k \cdot (s/n)} \cdot \frac{10 \ln(1/\overline{\delta})}{\overline{\epsilon}^2}$ do:
   (a) Invoke Sample-a-clique$(S, T, m(S), m(T), D(S), D(T), k, \overline{m})$.
   (b) If the procedure returned `fail` then set $\chi_i = 0$. Otherwise, let $K_i = (u_i, v_i, w_{i,1}, \ldots, w_{i,k-2})$ be the k-tuple returned and do the following.
      i. Query the degree and invoke Is-sociable$(x, T, m(T), D(T), \overline{m}, \overline{C}_k, n, k, \overline{\epsilon}, \overline{\delta})$ on each vertex $x \in K_i$.
      ii. If u_i is the first vertex (according to $\prec$) in K_i for which Is-sociable returned `shy`, then set $\chi_i = 1$. Otherwise, set $\chi_i = 0$.
6. **Return** $\widehat{C}_k = \frac{m(S)(2\sqrt{\overline{m}})^{k-2}}{(k-2)! \cdot (s/n)} \cdot \frac{1}{q} \sum_{i=1}^{q} \chi_i$.

Figure 1: The main algorithm for computing a $(1 \pm \epsilon)$-estimate of the number of k-cliques in a graph (given a constant factor estimate of this number).

The main theorem of our paper is the following (where the second item in the theorem is used by the geometric search algorithm for C_k).

Theorem 10. *Consider an invocation of Algorithm* Approximate-cliques$(n, k, \overline{m}, \overline{C}_k, \epsilon, \delta)$.

1. *If* $\overline{m} \in [(1-\overline{\epsilon})m, m]$ *and* $\overline{C}_k \in [C_k/4, C_k]$, *then with probability at least* $1 - \delta$, Approximate-cliques *returns a value* $\widehat{C}_k$ *such that* $\widehat{C}_k \in (1 \pm \epsilon) \cdot C_k$.
2. *If* $\overline{m} \in [(1-\overline{\epsilon})m, m]$ *and* $\overline{C}_k > C_k$, *then with probability at least* $\epsilon/4$, Approximate-cliques *returns a value* $\widehat{C}_k$ *such* $\widehat{C}_k \leq (1+\epsilon) \cdot C_k$.

3. *If* $\overline{C}_k \leq \overline{m}^{k/2}$*, then the expected query complexity and running time of* Approximate-cliques *are* $O\left(\frac{n}{\overline{C}_k^{1/k}} + \frac{\max\{m,\overline{m}\}\cdot\overline{m}^{(k-2)/2}}{\overline{C}_k} \cdot \frac{C_k}{\overline{C}_k}\right) \cdot \mathrm{poly}(\log n, 1/\epsilon, \log(1/\delta), k)$.[2] *The number of queries is always upper bounded by* $O\left(\frac{n}{\overline{C}_k^{1/k}}\right) \cdot \mathrm{poly}(\log n, 1/\epsilon, \log(1/\delta), k) + \min\{m, \overline{m}\}$.

Sample-degrees-typical. This procedure is described in Figure 4. Its goal is to output a degrees-typical multiset, which is defined below. The procedure itself is quite simple; it repeats the process of sampling a uniform random multiset for a sufficient number of times in order to achieve this condition with high probability.

Definition 11. *We say that a multiset* T *of size* t *is* degrees-typical *if* $m(T) \leq \frac{t}{n} \cdot 4\overline{m}$ *and for every high-degree vertex* $w \in V$*, it holds that* $d_T(w) \in (1 \pm (\overline{\epsilon}/k)) \cdot \frac{t}{n} \cdot d(w)$.

In Subsection 3.5 we prove the following lemma regarding the correctness and running time of the procedure Sample-degrees-typical.

Lemma 12. *Consider an invocation of* Sample-degrees-typical$(n, k, \overline{m}, \overline{C}_k, \overline{\epsilon}, \overline{\delta})$*. The procedure either returns* `fail`*, or returns a multiset* T *together with* $m(T)$ *and a data structure* $D(T)$ *that supports selecting a uniform edge in* $E(T)$ *in time* $O(1)$.

Let $\gamma = \min\{1/(4\overline{m}^{k/2}), \overline{\delta}\}$*. If* $\overline{m} \in [(1-\overline{\epsilon})m, m]$*, then with probability at least* $1-\gamma$*, the procedure returns a triple* $(T, m(T), D(T))$ *such that the multiset* T *is degrees-typical.*

The running time of Sample-degrees-typical *is* $O\left(\frac{n}{\sqrt{m}} \cdot \frac{k^2 \cdot \log(n/\gamma) \cdot \log(1/\gamma)}{\overline{\epsilon}^2}\right)$.

Sample-a-clique. This procedure is given in Figure 2. As mentioned earlier, this is the most important and novel aspect of our algorithm. Given any multiset of vertices S, the procedure Sample-a-clique produces cliques incident to S with roughly uniform probability.

We introduce the following notation, to precisely state the properties of the output of Sample-a-clique.

Definition 13. *Let* $\mathcal{C}(S)$ *denote the set of* k*-tuples* $(u, v, w_1, \ldots, w_{k-2})$ *that have the following properties: (1) the subgraph induced by* $\{u, v, w_1, \ldots, w_{k-2}\}$ *is a* k*-clique; (2)* $u \in S$*; (3)* $v \prec w_j$ *for every* $j \in [k-2]$.

By the above definition, each clique containing a vertex $u \in S$ is associated with the edge (u, v) in the clique such that $v \prec w$ for every other vertex $w \neq u$ in the clique, and the clique has exactly $(k-2)!$ corresponding tuples in $\mathcal{C}(S)$. In Subsection 3.3 we prove the following lemma regarding the correctness and running time of the procedure Sample-a-clique.

Lemma 14. *Let* T *be a degrees-typical multiset and let* S *be any multiset. For any fixed* k*-tuple* $K \in \mathcal{C}(S)$*, the probability that an invocation of* Sample-a-clique$(S, T, m(S), m(T), D(S), D(T), k, \overline{m})$ *returns* K *is in* $(1 \pm \overline{\epsilon}) \cdot \frac{1}{m(S)\cdot(2\sqrt{\overline{m}})^{k-2}}$*. The running time of* Sample-a-clique *is* $O(k^2)$.

Is-sociable. This procedure (provided in Figure 3) decides if a vertex u is sociable or not. This is done by multiple independent invocations of Sample-a-clique for the set $S = \{u\}$. In Subsection 3.4 we prove the following lemma regarding the correctness and running time of the procedure Is-sociable.

[2] In the second additive term there is actually a dependence on $2^k/(k-2)!$, which we ignored for the sake of simplicity.

Lemma 15. *Let T be a degrees-typical multiset. If* ls-sociable$(u, T, m(T), D(T), \overline{m}, \overline{C}_k, n, k, \overline{\epsilon}, \overline{\delta})$ *is invoked with a sociable vertex u, then with probability at least $1 - \overline{\delta}/n$, the procedure returns* `sociable`*, and if u is shy, then with probability at least $1 - \overline{\delta}/n$, the procedure returns* `shy`*. The running time of* ls-sociable *is* $O\left(\frac{\overline{m}^{k/2}}{\overline{C}_k} \cdot \frac{k \cdot 2^k \cdot \log(n/\overline{\delta})}{(k-2)! \cdot \overline{\epsilon}^{2-1/k}}\right)$.

Proof sketch of the first item in Theorem 10. The full proof of Theorem 10 appears in Subsection 3.6. Here we provide a proof sketch for the case that $\overline{C}_k \in [C_k/4, C_k]$, relying on Lemmas 12–15.

By the first premise of this item, $\overline{m} \in [(1-\overline{\epsilon})m, m]$. Hence, by Lemma 12, with high probability the multiset T is degrees-typical. From this point on we condition on this event.

Consider (as a thought experiment) invoking ls-sociable on *all* vertices with the degrees-typical multiset T. Based on these invocations, we can define a partition $P_{is} = (V_0, V_1)$, where V_0 contains all vertices for which ls-sociable returns `shy` and V_1 contains all vertices for which ls-sociable returns `sociable`. By Lemma 15 and a union bound over all vertices, we get that with probability at least $1 - \overline{\delta}$, P_{is} is an appropriate partition (as defined in Definition 6). Conditioned on P_{is} being appropriate (and using our assumptions on $\overline{m}$ and $\overline{C}_k$), by Corollary 8 we have that $\alpha_{P_{is}}(V) \in [(1-\overline{\epsilon})C_k, C_k]$.

Now consider the selection of the multiset S. We show that the size s of this sample ensures that with high probability $\alpha_{P_{is}}(S)$ is close to its expected value, $\frac{s}{n} \cdot \alpha_{P_{is}}(V)$, so that $\alpha_{P_{is}}(S) \in (1 \pm \overline{\epsilon})^2 \cdot \frac{s}{n} \cdot C_k$. We condition on this event as well. Since T is degree-typical, by Lemma 14, whenever we invoke Sample-a-clique (in Step 5a) it returns each k-tuple in $\mathcal{C}(S)$ with probability approximately $\frac{1}{m(S) \cdot (2\sqrt{\overline{m}})^{k-2}}$. Observe that if Sample-a-clique returns a k-tuple $K_i = (u_i, v_i, w_{i,1}, \ldots, w_{i,k-2})$, then in Steps 5(b)i and 5(b)ii the algorithm determines whether the corresponding k-clique is assigned to u_i according to P_{is} and sets χ_i to 1. Therefore, $\Pr[\chi_i = 1]$ is approximately $\frac{(k-2)! \cdot \alpha_{P_{is}}(S)}{m(S) \cdot (2\sqrt{\overline{m}})^{k-2}}$ (recall that for each k-clique that a vertex $u \in S$ participates in, there are $(k-2)!$ corresponding k-tuples in $\mathcal{C}(S)$). By the setting of the number of invocations, q, of Sample-a-clique, with high probability, the sum of the χ_i's is close to its expected value, and the output of the algorithm is as claimed.

3.3 Sampling a clique

In this subsection we provide the procedure Sample-a-clique and prove Lemma 14. The procedure first samples a uniform edge (u, v) in $E(S)$. It then tries to construct a k-clique that this edge participates in by selecting $k-2$ additional vertices. More precisely, it tries to construct a k-tuple $(u, v, w_1, \ldots, w_{k-2})$ in $\mathcal{C}(S)$. Recall that such a k-tuple satisfies: $u \in S$, $\{u, v, w_1, \ldots, w_{k-2}\}$ induces a k-clique, and $v \prec w_j$ for each $j \in [k-2]$. To this end the procedure repeats the following $k-2$ times. If v is a low-degree vertex, then it selects a uniform neighbor w of v, and maintains it with probability $d(v)/2\sqrt{\overline{m}}$. If v is a high-degree vertex, then the procedure tries to sample a random high-degree vertex. It does so by first sampling a uniform edge $(x, y) \in E(T)$ and if y is a high-degree neighbor of v, performing rejection sampling according to the degree of y. We prove that conditioned on T being degrees-typical, we obtain each k-tuple in $\mathcal{C}(S)$ with almost equal probability.

Proof of Lemma 14: Let $(a, b, z_1, \ldots, z_{k-2})$ be a k-tuple in $\mathcal{C}(S)$. Recall that by the definition of $\mathcal{C}(S)$ we have that $a \in S$, the subgraph induced by $\{a, b, z_1, \ldots, z_{k-2}\}$ is a k-clique and $b \prec z_j$ for every $j \in [k-2]$. If Sample-a-clique does not return `fail`, then its output is a tuple $(u, v, w_1, \ldots, w_{k-2})$ in $\mathcal{C}(S)$. The probability that the procedure returns the tuple $(a, b, z_1, \ldots, z_k)$

Sample-a-clique $(S, T, m(S), m(T), D(S), D(T), k, \overline{m})$

1. Sample a uniform edge $e = (u, v)$ in $E(S)$ (using the data structure $D(S)$).
2. For $j = 1$ to $k - 2$ do:
   (a) If $d(v) \leq 2\sqrt{\overline{m}}$ (v is a low-degree vertex), then:
      i. Uniformly select a neighbor w_j of v.
      ii. Keep w_j with probability $\frac{d(v)}{2\sqrt{\overline{m}}}$ and with probability $1 - \frac{d(v)}{2\sqrt{\overline{m}}}$ **return** `fail`.
   (b) Else (v is a high-degree vertex):
      i. Sample a random edge (x, y) in $E(T)$ (using the data structure $D(T)$).
      ii. If $d(y) \leq 2\sqrt{\overline{m}}$, **return** `fail`.
      iii. With probability $\frac{m(T)}{d(y) \cdot \frac{t}{n} \cdot 2\sqrt{\overline{m}}}$ set $w_j = y$, otherwise, **return** `fail`.
3. For every pair of vertices in $\{u, v, w_1, \ldots, w_{k-2}\}$ query if there is an edge between the two vertices.
4. If the subgraph induced by $\{u, v, w_1, \ldots, w_{k-2}\}$ is a k-clique and $v \prec w_j$ for every $j \in [k-2]$ then **return** $K = (u, v, w_1, \ldots, w_{k-2})$. Otherwise **return** `fail`.

Figure 2: A procedure for sampling a k-clique incident to S with almost uniform probability.

is

$$\begin{aligned}
&\Pr\Big[(u,v) = (a,b) \text{ and } \forall j \in [k-2]\,, w_j = z_j\Big] \\
&\quad = \Pr[(u,v) = (a,b)] \cdot \Pr\Big[\forall j \in [k-2]\,, w_j = z_j \mid (u,v) = (a,b)\Big]\,.
\end{aligned}$$

Clearly, $\Pr[(u, v) = (a, b)] = \frac{1}{m(S)}$, so it remains to compute the probability that $w_j = z_j$ for each $j \in [k-2]$, conditioned on $(u, v) = (a, b)$.

If $b = v$ is a low-degree vertex, then the vertices $w_1, \ldots, w_{k-2}$ are uniformly selected random neighbors of v. For each $j \in [k-2]$, the probability that $w_j = z_j$ and that the procedure did not return `fail` due to rejection sampling is $\frac{1}{d(v)} \cdot \frac{d(v)}{2\sqrt{\overline{m}}} = \frac{1}{2\sqrt{\overline{m}}}$. Therefore, $\Pr\Big[\forall j \in [k-2]\,, w_j = z_j \mid (u,v) = (a,b)\Big] = 1/(2\sqrt{\overline{m}})^{k-2}$.

Otherwise (b is a high-degree vertex), since $b \prec z_j$ for each $j \in [k-2]$, the vertices $z_1, \ldots, z_{k-2}$ are also high-degree vertices. In this case (conditioned on $(u, v) = (a, b)$), the procedure tries to sample $k - 2$ high-degree vertices by sampling edges originating from the vertices of T. We next prove that, in Step 2a, any specific high-degree vertex is sampled with probability in $\left(1 \pm \frac{\overline{\epsilon}}{k}\right) \frac{1}{2\sqrt{\overline{m}}}$.

Since T is degrees-typical (as defined in Definition 11), $m(T) \leq \frac{t}{n} \cdot 4\overline{m}$. This implies that for every high-degree vertex z,

$$\frac{m(T)}{d(z) \cdot \frac{t}{n} \cdot 2\sqrt{\overline{m}}} \leq \frac{\frac{t}{n} \cdot 4\overline{m}}{2\sqrt{\overline{m}} \cdot \frac{t}{n} \cdot 2\sqrt{\overline{m}}} = 1$$

Thus, Step 2(b)i is valid. Since T is degrees-typical, for every high-degree vertex z, $d_T(z) \in \left(1 \pm \frac{\overline{\epsilon}}{k}\right) \cdot \frac{t}{n} \cdot d(z)$.

For any vertex y, the probability of obtaining an edge in Step 2(b)i with y as an endpoint is $d_T(y)/m(T)$. Therefore, for each $j \in [k-2]$, the probability that $w_j = z_j$ in Step 4 is

$$\frac{d_T(z_j)}{m(T)} \cdot \frac{m(T)}{d(z_j) \cdot \frac{t}{n} \cdot 2\sqrt{\overline{m}}} \in \left(1 \pm \frac{\overline{\epsilon}}{k}\right) \cdot \frac{1}{2\sqrt{\overline{m}}}\,.$$

It follows that the probability that the procedure returns any specific k-tuple in $\mathcal{C}(S)$ is $(1 \pm \overline{\epsilon}) \cdot \frac{1}{m(S)(2\sqrt{\overline{m}})^{k-2}}$.

It remains to bound the running time of the procedure. Given the data structure $D(S)$, it takes time $O(1)$ to sample an edge in $E(S)$, and similarly it takes time $O(1)$ to sample an edge in $D(T)$. The procedure samples a single edge in $E(S)$ and possibly $k-2$ edges in $E(T)$. Adding the time to perform queries on all pairs of vertices in $\{u, v, w_1, \ldots, w_{k-2}\}$ (in addition to a degree query on each of these vertices), the total running time is $O(k^2)$. □

3.4 Is-sociable

The procedure Is-sociable determines (with high success probability) whether a given vertex u is sociable or shy. For vertices that are neither sociable nor shy, it can answer arbitrarily. Recall that by Definition 4, the distinction between a sociable u and a shy u involves bounds on both $d(u)$ and $c_k(u)$. These will be critical in bounding the running time of Is-sociable. The procedure basically invokes Sample-a-clique repeatedly to check if $c_k(u)$ is larger than the specified threshold.

Is-sociable $(u, T, m(T), D(T), \overline{m}, \overline{C}_k, n, k, \overline{\epsilon}, \overline{\delta})$

1. Query the degree of u and if $d(u) > 4\overline{m}/(\overline{\epsilon}\overline{C}_k)^{1/k}$ then **return** `sociable`.
2. For $i = 1$ to $r = \frac{d(u)\cdot(2\sqrt{\overline{m}})^{k-2}}{(k-2)!\cdot(k\cdot(50\overline{C}_k)^{1-1/k}/\overline{\epsilon}^{1/k})} \cdot \frac{15\ln(n/\overline{\delta})}{\overline{\epsilon}^2}$ do:
   (a) Invoke Sample-a-clique$(\{u\}, T, d(u), m(T), D(\{u\}), D(T), k, \overline{m})$.
   (b) If a k-tuple (corresponding to a k-clique) was returned, then set $\chi_i = 1$. Otherwise (the procedure returned `fail`), set $\chi_i = 0$.
3. Let $\widehat{c}_k(u) = \frac{d(u)\cdot 2\sqrt{\overline{m}}}{r} \cdot \sum_{i=1}^{r} \chi_i$.
4. If $\widehat{c}_k(u) \geq \frac{1}{2}(k \cdot (50\overline{C}_k)^{1-1/k}/\overline{\epsilon}^{1/k})$, then **return** `sociable`, otherwise, **return** `shy`.

Figure 3: A procedure for determining with high probability whether a given vertex is sociable or not.

Proof of Lemma 15: Consider any fixed vertex u. Let $\chi = \frac{1}{r}\sum_{i=1}^{r} \chi_i$, where $\chi_1, \ldots, \chi_r$ are as defined in Step 2b of Is-sociable. Note that $S = \{u\}$, $m(S) = m(\{u\}) = d(u)$ and $|\mathcal{C}(S)| = \mathcal{C}(\{u\}) = (k-2)! \cdot c_k(u)$. By Lemma 14 and the assumption that T is degrees-typical, $\mathrm{Exp}[\chi] \in (1 \pm \overline{\epsilon}) \cdot \frac{(k-2)!\cdot c_k(u)}{d(u)\cdot(2\sqrt{\overline{m}})^{k-2}}$.

First consider the case that u is sociable. By Definition 4, either $d(u) > 4\overline{m}/(\overline{\epsilon}\overline{C}_k)^{1/k}$ or $c_k(u) \geq k \cdot (50\overline{C}_k)^{1-1/k}/\overline{\epsilon}^{1/k}$ (or both). Clearly, if $d(u) > 4\overline{m}/(\overline{\epsilon}\overline{C}_k)^{1/k}$, then the procedure returns `sociable`. Hence, assume that $c_k(u) \geq k \cdot (50\overline{C}_k)^{1-1/k}/\overline{\epsilon}^{1/k}$. By Chernoff's inequality and the setting of $r = \frac{d(u)\cdot(2\sqrt{\overline{m}})^{k-2}}{(k-2)!\cdot(k\cdot(50\overline{C}_k)^{1-1/k}/\overline{\epsilon}^{1/k})} \cdot \frac{15\ln(n/\overline{\delta})}{\overline{\epsilon}^2}$ in Step 2 of the procedure,

$$\Pr\left[\frac{1}{r}\sum_{i=1}^{q} \chi_i < (1-\overline{\epsilon}) \cdot \mathrm{Exp}[\chi]\right] \;<\; \exp\left(-\frac{\overline{\epsilon}^2 \cdot \mathrm{Exp}[\chi] \cdot r}{3}\right) \;\leq\; \frac{\overline{\delta}}{n} .$$

It follows that if T is typical and u is sociable, then with probability at least $1 - \overline{\delta}/n$,

$$\widehat{c}_k(u) \geq (1-\overline{\epsilon})^2 \cdot c_k(u) \geq \frac{1}{2} \cdot k \cdot (50\overline{C}_k)^{1-1/k}/\overline{\epsilon}^{1/k},$$

causing the procedure to return `sociable`.

Now consider the case that u is shy. By Definition 4, $c_k(u) \leq \frac{1}{4} \cdot k \cdot (50\overline{C}_k)^{1-1/k}/\overline{\epsilon}^{1/k}$, implying that $\text{Exp}[\chi] < (1+\overline{\epsilon}) \cdot \frac{\frac{(k-2)!}{4} \cdot (k \cdot (50\overline{C}_k)^{1-1/k}/\overline{\epsilon}^{1/k})}{d(u) \cdot (2\sqrt{\overline{m}})^{k-2}}$. Hence, by Chernoff's inequality, and by the setting of r,

$$\Pr\left[\frac{1}{r}\sum_{i=1}^{r}\chi_i > \frac{3}{2} \cdot \left(\frac{(1+\overline{\epsilon})}{4} \cdot k \cdot (50\overline{C}_k)^{1-1/k}/\overline{\epsilon}^{1/k}\right)\right] < \frac{\overline{\delta}}{n}.$$

Therefore, if T is degrees-typical and u is shy, then with probability at least $1-\overline{\delta}/n$, the algorithm returns `shy`.

It remains to bound the running time of the procedure. By Lemma 14, the procedure Sample-a-clique runs in time $O(k^2)$. Crucially, Sample-a-clique is invoked only if $d(u) \leq 4\overline{m}/(\overline{\epsilon}\overline{C}_k)^{1/k}$. Since $r = \frac{d(u) \cdot (2\sqrt{\overline{m}})^{k-2}}{(k-2)! \cdot (k \cdot (50\overline{C}_k)^{1-1/k}/\overline{\epsilon}^{1/k})} \cdot \frac{15\ln(n/\overline{\delta})}{\overline{\epsilon}^2}$, the running time of the procedure is $O\left(r \cdot k^2\right) = O\left(\frac{\overline{m}^{k/2}}{\overline{C}_k} \cdot \frac{k \cdot 2^k \cdot \log(n/\overline{\delta})}{(k-2)! \cdot \overline{\epsilon}^{2-1/k}}\right)$. □

3.5 Sampling a degrees-typical set

In this subsection we provide the (simple) procedure Sample-degrees-typical and prove Lemma 12 regarding its correctness and running time. As in the case of the choice of the sample S by Approximate-cliques, here too if the sample size t is larger than n, then the algorithm can simply set $T = V$.

Sample-degrees-typical $(n, k, \overline{m}, \overline{C}_k, \overline{\epsilon}, \overline{\delta})$

1. $\gamma = \min\{1/(4\overline{m}^{k/2}), \overline{\delta}\}$.
2. For $i = 1$ to $\log(2/\gamma)$ do:
   (a) Let T_i be a multiset of $t = \frac{10n \cdot \ln(2n/\gamma^2)}{(\overline{\epsilon}/k)^2 \cdot 2\sqrt{\overline{m}}}$ vertices chosen uniformly, at random.
   (b) Query the degrees of all the vertices in T_i and compute $m(T_i)$.
3. Let T be the first set T_i such that $m(T_i) \leq \frac{t}{n} \cdot 4\overline{m}$. If no such set exists, then **return** `fail`. Else, set up a data structure $D(T)$ that supports sampling a uniform edge in $E(T)$ in constant time.
4. **Return** $(T, m(T), D(T))$.

Figure 4: The procedure for sampling the multiset T.

Proof of Lemma 12: For each iteration i, consider the selection of the multiset T_i. For any fixed high-degree vertex $u \in V$ and for $j = 1, \ldots, t$, let $\chi_j(u) = 1$ if the j^{th} vertex in T_i is a neighbor of u and let $\chi_j(u) = 0$ otherwise. By the definition of $\chi_1(u), \ldots, \chi_t(u)$ and the premise of the lemma regarding $\overline{m}$, $\text{Exp}\left[\frac{1}{t}\sum_{j=1}^{t}\chi_j(u)\right] = \frac{d(u)}{n} \geq \frac{\sqrt{m}}{n}$. By Chernoff's inequality and the setting of $t = \frac{10n \cdot \ln(2n/\gamma^2)}{(\overline{\epsilon}/k)^2 \cdot 2\sqrt{\overline{m}}}$ in Step 3,

$$\Pr\left[\left|\frac{1}{t}\sum_{j=1}^{t}\chi_j(u) - \frac{d(u)}{n}\right| > \frac{\overline{\epsilon}}{k} \cdot \frac{d(u)}{n}\right] < 2\exp\left(-\frac{(\overline{\epsilon}/k)^2 \cdot t \cdot \frac{d(u)}{n}}{3}\right) < \frac{\gamma^2}{2n}.$$

By taking a union bound over all high-degree vertices, it holds that with probability at least $1 - \gamma^2/2$, for every high-degree vertex $u \in V$, $d_{T_i}(u) = \sum_{j=1}^{t}\chi_j(u) \in (1 \pm (\overline{\epsilon}/k)) \cdot \frac{t}{n} \cdot d(u)$. By

taking a union bound over the $\log(2/\gamma)$ sampled multisets T_i, it holds that with probability at least $1 - \frac{1}{2}\gamma^2 \log(2/\gamma) > 1 - \gamma/2$, for each of the multisets T_i and for every high-degree vertex $u \in V$, $d_{T_i}(u) \in \left(1 \pm \frac{\overline{\epsilon}}{k}\right) \cdot \frac{t}{n} \cdot d(u)$.

We now turn to bounding the probability that none of the multiset T_i satisfies $m(T_i) \leq \frac{t}{n} \cdot 4\overline{m}$. By the definition of $m(\cdot)$, for every T_i we have that $\mathrm{Exp}[m(T_i)] = \frac{t}{n} \cdot m$. By the assumption that $\overline{m} \in [(1-\overline{\epsilon})m, m]$ and by Markov's inequality, $\Pr[m(T_i) > \frac{t}{n} \cdot 4\overline{m}] \leq \Pr[m(T_i) > \frac{t}{n} \cdot 2m] < \frac{1}{2}$. Hence, the probability that $m(T_i) > \frac{t}{n} \cdot 4\overline{m}$ for every $i = 1, \ldots, \log(2/\gamma)$ is at most $\gamma/2$.

By combining the two failure probabilities, we get that with probability at least $1 - \gamma$, the algorithm returns a multiset T that is degrees-typical (as defined in Definition 11).

Finally, by a performing a preprocessing step that takes $\Theta(t)$ time it is possible to build a data structure $D(T)$ that allow for sampling each vertex $u \in S$ with probability proportional to $d(u)/m(T)$ (see e.g., [51, 52, 33]). This in turn implies that, using $D(T)$, it is possible to sample a uniform edge in $E(T)$ in constant time.

The running time of Sample-degrees-typical is $O\left(t \cdot \log(1/\gamma)\right) = O\left(\frac{n}{\sqrt{\overline{m}}} \cdot \frac{k^2 \cdot \log(n/\gamma) \cdot \log(1/\gamma)}{\overline{\epsilon}^2}\right)$. □

3.6 Proof of Theorem 10

In this subsection we prove Theorem 10. We first define the notion of a *cliques-typical* multiset.

Definition 16. *We say that a multiset S of size s is cliques-typical with respect to a partition P if $\alpha_P(S) \in (1 \pm \overline{\epsilon})^2 \cdot \frac{s}{n} \cdot C_k$.*

We establish a simple claim regarding the multiset S selected by our algorithm (appropriate partitions are as defined in Definition 6).

Claim 17. *Consider the multiset S sampled in Step 2 of Algorithm Approximate-cliques. For any fixed partition P of V, $\mathrm{Exp}[\alpha_P(S)] \leq \frac{s}{n} \cdot C_k$. Furthermore, if P is appropriate, $\overline{m} \geq (1-\epsilon)m$ and $\overline{C}_k \in [C_k/4, C_k]$, then with probability at least $1 - \overline{\delta}$ (over the choice of S), the multiset S is cliques-typical.*

Proof. Recall that by Definition 7, $\alpha_P(S)$ is the number of k-cliques assigned to the vertices of S, and that by Corollary 8, for every partition P, $\alpha_P(V) \leq C_k$. Hence, $\mathrm{Exp}[\alpha_P(S)] = \frac{s}{n} \cdot \alpha_P(V) \leq \frac{s}{n} \cdot C_k$.

We turn to consider the case that the partition P is appropriate, $\overline{m} \geq (1-\epsilon)m$ and $\overline{C}_k \in [C_k/4, C_k]$. By Corollary 8, $\alpha_P(V) \in [(1-\overline{\epsilon})C_k, C_k]$. By Definition 7, k-cliques are only assigned to vertices that are not sociable, implying that for every vertex $u \in V$, $\alpha_P(u) \leq c_k(u) \leq k \cdot (50\overline{C}_k)^{1-1/k}/\overline{\epsilon}^{1/k}$. Hence, by the multiplicative Chernoff bound and the setting of $s = \frac{700 \cdot k \cdot n \cdot \ln(1/\overline{\delta})}{\overline{\epsilon}^{2+1/k} \cdot \overline{C}_k^{1/k}}$,

$$\Pr\left[\left|\frac{1}{s}\sum_{u \in S} \alpha_P(u) - \frac{\alpha_P(V)}{n}\right| > \overline{\epsilon} \cdot \frac{\alpha_P(V)}{n}\right] < 2\exp\left(-\frac{\overline{\epsilon}^2 \cdot \frac{\alpha_P(V)}{n} \cdot s}{3 \cdot k \cdot (50\overline{C}_k)^{1-1/k}/\overline{\epsilon}^{1/k}}\right) < \overline{\delta} \,.$$

Therefore, if P is appropriate, $\overline{m} \geq (1-\epsilon)m$ and $\overline{C}_k \in [C_k/4, C_k]$, with probability at least $1 - \overline{\delta}$,

$$\alpha_P(S) \in (1 \pm \overline{\epsilon}) \cdot \frac{s}{n} \cdot \alpha_P(V) \in (1 \pm \overline{\epsilon})^2 \cdot \frac{s}{n} \cdot C_k,$$

which implies that S is cliques-typical by Definition 16. □

3.6.1 The case $\overline{C}_k \in [C_k/4, C_k]$

We start with the first item in the theorem. Recall that by the first premise of this item, $\overline{m} \in [(1-\overline{\epsilon})m, m]$. By Lemma 12, with probability at least $1 - \min\{1/(4\overline{m}^{k/2}), \overline{\delta}\} \geq 1 - \overline{\delta}$ the procedure Sample-degrees-typical returns a multiset T that is degrees-typical. We henceforth condition on this event.

Since T is degrees-typical, by Lemma 15, an invocation of the procedure Is-sociable with a shy vertex u returns `shy` with probability at least $1 - \overline{\delta}/n$, and similarly an invocation with a sociable vertex u returns `sociable` with probability at least $1 - \overline{\delta}/n$. Consider (as a thought experiment) running the procedure Is-sociable on all the vertices in the graph, and letting V_0 be the set of vertices for which the procedure returned `shy`, $V_1 = V \setminus V_0$ and $P_{is} = (V_0, V_1)$. Conditioned on the event that T is degrees-typical, by Lemma 15 and by taking a union bound over all the vertices in V, with probability at least $1 - \overline{\delta}$ the partition P_{is} is appropriate (as defined in Definition 6). Suppose we fix the random coins used by Is-sociable on all vertices to a setting that indeed induces an appropriate partition P_{is}, and assume that all calls made by the algorithm to Is-sociable are answered consistently with P_{is}. (The probability that P_{is} is not appropriate is taken into account in the total failure probability of the algorithm.) Conditioned on P_{is} being appropriate (and using our assumptions on $\overline{m}$ and $\overline{C}_k$), by Corollary 8 we have that $\alpha_{P_{is}}(V) \in [(1-\overline{\epsilon})C_k, C_k]$.

Now consider the multiset S sampled in Step 2 of the algorithm. By Claim 17, with probability at least $1-\overline{\delta}$, S is cliques-typical with respect to P_{is}. That is, $\alpha_{P_{is}}(S) \in (1\pm\overline{\epsilon})^2 \cdot \frac{s}{n} \cdot C_k$. We condition on this event as well. Since T is degree-typical, by Lemma 14, whenever we invoke Sample-a-clique (in Step 5a) it returns each k-tuple in $\mathcal{C}(S)$ with probability in $(1 \pm \overline{\epsilon}) \cdot \frac{1}{m(S)\cdot(2\sqrt{\overline{m}})^{k-2}}$.

By the description of Steps 5(b)i and 5(b)ii, χ_i is set to 1 only if the procedure Sample-a-clique returns a k-tuple $K = (u, v, w_1, \ldots, w_{k-2})$ in $\mathcal{C}(S)$ that is assigned to u according to P_{is}. For each k-clique assigned to a vertex $u \in S$, the number of corresponding k-tuples in $\mathcal{C}(S)$ is $(k-2)!$. Therefore, for a cliques-typical multiset S,

$$\mathrm{Exp}[\chi_i] \in (1 \pm \overline{\epsilon}) \cdot \frac{\alpha_{P_{is}}(S) \cdot (k-2)!}{m(S) \cdot (2\sqrt{\overline{m}})^{k-2}} \in (1 \pm \overline{\epsilon})^3 \cdot \frac{\frac{s}{n} \cdot C_k \cdot (k-2)!}{m(S) \cdot (2\sqrt{\overline{m}})^{k-2}} \,. \tag{1}$$

By the multiplicative Chernoff bound and by the setting of $q = \frac{m(S)\cdot(2\sqrt{\overline{m}})^{k-2}}{(1-\overline{\epsilon})^3\cdot(k-2)!\cdot\overline{C}_k\cdot(s/n)} \cdot \frac{10\ln(1/\overline{\delta})}{\overline{\epsilon}^2}$ in Step 5 of the algorithm,

$$\Pr\left[|\chi - \mathrm{Exp}[\chi]| > \overline{\epsilon} \cdot \mathrm{Exp}[\chi]\right] < 2\exp\left(-\frac{\overline{\epsilon}^2 \cdot \mathrm{Exp}[\chi] \cdot q}{3}\right) < \overline{\delta} \,.$$

Therefore, if $\overline{m} \in [(1-\overline{\epsilon})m, m]$, $\overline{C}_k \in [C_k/4, C_k]$, T is degrees-typical, P_{is} is appropriate and S is cliques-typical, then with probability at least $1 - \overline{\delta}$,

$$\chi \in (1 \pm \overline{\epsilon})^4 \cdot \frac{\frac{s}{n} \cdot C_k \cdot (k-2)!}{m(S) \cdot (2\sqrt{\overline{m}})^{k-2}} \implies \widehat{C}_k \in (1 \pm \overline{\epsilon})^4 \cdot C_k \in (1 \pm \epsilon) C_k \,,$$

where we have used the fact that $\overline{\epsilon} = \epsilon/5$. By taking a union bound over all bad events, $C_k \in (1 \pm \epsilon)C_k$ with probability at least $1 - 4\overline{\delta} > 1 - \delta$ (since $\overline{\delta} = \delta/4$).

3.6.2 The case $\overline{C}_k > C_k$

We now prove the second item of the theorem. As in the first item, since $\overline{m} \in [(1-\overline{\epsilon})m, m]$, by Lemma 12 with probability at least $1 - \min\{1/(4\overline{m}^{k/2}), \overline{\delta}\}$ the multiset T is degrees-typical.

Conditioned on T being degrees-typical, an invocation of the procedure Sample-a-clique returns each k-tuple in $\mathcal{C}(S)$ with probability in $(1 \pm \epsilon)\frac{1}{m(S)\cdot(2\sqrt{\overline{m}})^{k-2}}$.

In this case, since $\overline{C}_k > C_k$, the setting of r in the procedure Is-sociable is not sufficiently large to ensure that the procedure is accurate, and that the partition P_{is} is appropriate (with high probability). Similarly, $\alpha_{P_{is}}(S)$ might not be close to its expected value. Therefore, we only use an upper bound on the expected value of $\alpha_{P_{is}}(S)$, as explained next.

Consider the following random variables. For a multiset S and for each $i \in [q]$ let $\chi_i(S)$ denote the random variable χ_i conditioned on S (where q and χ_i are as defined in Step 5 of the algorithm). Let $Y_i(S) = \frac{m(S)\cdot(2\sqrt{\overline{m}})^{k-2}}{(k-2)!(s/n)} \cdot \chi_i(S)$, and let $\widehat{C}_k(S)$ denote the value of the random variable $\widehat{C}_k$ (which is the output of the algorithm), conditioned on S. We use the notation $\mathtt{S\text{-}c}$ to denote the random coins of the procedure Sample-a-clique. By the above discussion and by the setting of $\widehat{C}_k$ in Step 6, if T is degrees-typical, then

$$\begin{aligned}
\mathrm{Exp}\left[\widehat{C}_k\right] &= \mathrm{Exp}_S\left[\mathrm{Exp}_{\mathtt{S\text{-}c}}\left[\widehat{C}_k(S)\right]\right] \\
&= \mathrm{Exp}_S\left[\mathrm{Exp}_{\mathtt{S\text{-}c}}\left[\frac{m(S)\cdot(2\sqrt{\overline{m}})^{k-2}}{(k-2)!\cdot\frac{s}{n}} \cdot \frac{1}{q}\sum_{i=1}^{q}\chi_i(S)\right]\right] \\
&= \mathrm{Exp}_S\left[\mathrm{Exp}_{\mathtt{S\text{-}c}}\left[\frac{1}{q}\sum_{i=1}^{q}Y_i(S)\right]\right] = \mathrm{Exp}_S\left[\mathrm{Exp}_{\mathtt{S\text{-}c}}\left[Y_1(S)\right]\right] \ . \qquad (2)
\end{aligned}$$

The last equality holds simply because all the $Y_i(S)$ variables are equally distributed (for each fixed S). We note that q also depends on S (since it is a function of $m(S)$), but this does not effect our analysis, and hence this dependence is not explicit. As in the analysis of the case that $\overline{C}_k \in [C_k/4, C_k]$, since for each k-clique assigned to a vertex $u \in S$, the number of corresponding k-tuples in $\mathcal{C}(S)$ is $(k-2)!$, we have that if T is degrees-typical, then

$$\Pr[\chi_i(S) = 1] \leq (1+\overline{\epsilon}) \cdot \frac{\alpha_{P_{is}}(S)\cdot(k-2)!}{m(S)\cdot(2\sqrt{\overline{m}})^{k-2}} \ .$$

By the definition of the $Y_i(S)$ variables and Equation (2), it follows that

$$\mathrm{Exp}\left[\widehat{C}_k\right] \leq \mathrm{Exp}_S\left[(1+\overline{\epsilon})\cdot\frac{\alpha_{P_{is}}(S)}{s/n}\right] = (1+\overline{\epsilon})\cdot\frac{n}{s}\cdot\mathrm{Exp}_S\left[\alpha_{P_{is}}(S)\right] \ .$$

By Claim 17, for any partition P, $\mathrm{Exp}[\alpha_P(S)] \leq \frac{s}{n}\cdot C_k$. Therefore, if T is degrees-typical, then $\mathrm{Exp}\left[\widehat{C}_k\right] \leq (1+\overline{\epsilon})C_k$. Finally, by Markov's inequality (and recalling that $\overline{\epsilon} = \epsilon/5$),

$$\Pr\left[\widehat{C}_k > (1+\epsilon/2)(1+\overline{\epsilon})C_k\right] < 1-\epsilon/2 \ .$$

As noted in Subsection 3.2, we can assume that $\overline{\epsilon} \geq 1/\overline{m}^{k/2}$. Since T is not degrees-typical with probability at most $\min\{1/\overline{m}^{k/2}, \overline{\delta}\}$, by taking a union bound and by the setting of $\overline{\epsilon} = \epsilon/5$, it holds that with probability at least $\epsilon/2 - \min\{1/(4\overline{m}^{k/2}), \overline{\delta}\} > \epsilon/4$, the algorithm returns a value $\widehat{C}_k$ such that $\widehat{C}_k \leq (1+\epsilon/2)(1+\overline{\epsilon})C_k \leq (1+\epsilon)C_k$.

3.6.3 The expected query complexity and running time

By Lemma 12 and the assumption that $\overline{C}_k \leq \overline{m}^{k/2}$, the invocation of the procedure Sample-degrees-typical takes $O\left(\frac{n}{\overline{C}_k^{1/k}} \cdot \frac{k^2\cdot\log(n/\gamma)\cdot\log(1/\gamma)}{\overline{\epsilon}^2}\right)$ time, for $\gamma = \Theta\left(\min\{1/\overline{m}^{k/2}, 1/\delta\}\right)$. The

sampling of S and the computation of $m(S)$ and $D(S)$ take time $O(s) = O\left(\frac{k \cdot n}{\overline{C}_k^{1/k}} \cdot \frac{\ln(1/\overline{\delta})}{\overline{\epsilon}^{2+1/k}}\right)$. By Lemma 14, each invocation of Sample-a-clique in Step 5a takes time $O(k^2)$, and Step 5(b)ii takes constant time. Therefore, excluding the invocations of the Is-sociable procedure, the running time of the for loop in Step 5 is $O\left(k^2 \cdot q\right)$. Since $\mathrm{Exp}[m(S)] = \frac{s}{n} \cdot m$ and by the setting of q, it follows that the expected running time of the for loop is $O\left(\frac{m \cdot \overline{m}^{(k-2)/2}}{\overline{C}_k} \cdot \frac{k^2 \cdot \ln(1/\overline{\delta})}{(k-2)! \cdot \overline{\epsilon}^2}\right)$. It remains to bound the running time resulting from the invocations of the procedure Is-sociable.

We first bound the expected number of invocations of Is-sociable when the multiset T is degrees-typical. Let I denote the number of invocations of Is-sociable. Similarly to the analysis of the case $\overline{C}_k > C_k$, let $z_i(S)$ be a 0/1 random variable that is defined as follows: $z_i(S) = 1$ if (and only if) the procedure Sample-a-clique returned a clique in the i^{th} step of the for loop in Step 5a of the algorithm, conditioned on the set S. As before, let S-c denote the random coins of Sample-a-clique. Here it is actually relevant that q depends on S, and therefore we use the notation $q(S)$. By the definition of I,

$$\mathrm{Exp}[I] = \mathrm{Exp}_S\left[\mathrm{Exp}_{\text{S-c}}\left[\sum_{i=1}^{q(S)} z_i(S)\right]\right] = \mathrm{Exp}_S\left[q(S) \cdot \mathrm{Exp}_{\text{S-c}}\left[z_1(S)\right]\right] = \mathrm{Exp}_S\left[\mathrm{Exp}_{\text{S-c}}\left[q(S) \cdot z_1(S)\right]\right] .$$

Recall that if T is degrees-typical, then by Lemma 14, an invocation of Sample-a-clique returns each k-tuple in $\mathcal{C}(S)$ with probability in $(1 \pm \overline{\epsilon}) \cdot \frac{1}{m(S) \cdot (2\sqrt{\overline{m}})^{k-2}}$. Hence,

$$\Pr_{\text{S-c}}[z_i(S) = 1] \leq (1 + \overline{\epsilon}) \frac{|\mathcal{C}(S)|}{m(S) \cdot (2\sqrt{\overline{m}})^{k-2}} .$$

By the setting of $q(S)$,

$$\mathrm{Exp}_{\text{S-c}}[q(S) \cdot z_1(S)] \leq \frac{(1 + \overline{\epsilon})|\mathcal{C}(S)|}{(1 - \overline{\epsilon})^3 \cdot (k-2)! \cdot \overline{C}_k \cdot \frac{s}{n}} \cdot \frac{10 \ln(1/\overline{\delta})}{\overline{\epsilon}^2} .$$

Since $\mathrm{Exp}[|\mathcal{C}(S)|] = \frac{s}{n} \cdot (k-2)! \cdot k \cdot C_k$, it follows that $\mathrm{Exp}[I] \leq \frac{C_k}{\overline{C}_k} \cdot \frac{50k \ln(1/\overline{\delta})}{\overline{\epsilon}^2}$. Therefore, by Lemma 15, if T is degrees-typical, then the expected running time resulting from all of the invocations of the Is-sociable procedure is $O\left(\frac{\overline{m}^{k/2}}{\overline{C}_k} \cdot \frac{C_k}{\overline{C}_k} \cdot \frac{k^2 \cdot 2^k \cdot \log(n/\overline{\delta}) \log(1/\overline{\delta})}{(k-2)! \cdot \overline{\epsilon}^{2-1/k}}\right)$.

If T is not degrees-typical then we can no longer upper bound the success probability of Sample-a-clique, implying that the number of invocations of Is-sociable can be $\Theta(q)$. However, since T is not degrees typical with probability at most $\frac{1}{\overline{m}^{k/2}}$, this does not affect the expected running time resulting from the invocations of Is-sociable. The remaining steps take constant time, and therefore the total running time of the algorithm is $O\left(\frac{n}{\overline{C}_k^{1/k}} \cdot \frac{k^2 \cdot \log(n/\gamma) \cdot \log(1/\gamma)}{\epsilon^{2+1/k}} + \frac{\min\{m, \overline{m}\} \cdot \overline{m}^{(k-2)/2}}{\overline{C}_k} \cdot \frac{C_k}{\overline{C}_k} \cdot \frac{k^2 \cdot 2^k \cdot \log(n/\overline{\delta}) \log(1/\overline{\delta})}{(k-2)! \cdot \overline{\epsilon}^{2-1/k}}\right)$ which is $O\left(\frac{n}{\overline{C}_k^{1/k}} + \frac{\max\{m, \overline{m}\} \cdot \overline{m}^{(k-2)/2}}{\overline{C}_k} \cdot \frac{C_k}{\overline{C}_k}\right) \cdot \mathrm{poly}(\log n, 1/\epsilon, \log(1/\delta), k)$, since $\gamma = \Theta\left(\min\{1/\overline{m}^k, \delta\}\right)$.

Finally, as discussed in the beginning of Subsection 3.2, if q is greater than $\overline{m}$ then the algorithm may query beforehand for all the edges incident to S and their neighbors, and if it views more than $\overline{m}$ edges then it can abort. Hence, the number of queries is always bounded by $O\left(\frac{n}{\overline{C}_k^{1/k}}\right) \cdot \mathrm{poly}(\log n, 1/\epsilon, k) + \min\{m, \overline{m}\}$.

3.7 The search algorithm

In this subsection we describe an algorithm that returns an estimate of the number of k-cliques in a graph G without prior knowledge on the number of edges or k-cliques in the graph. We prove this by establishing a more general claim:

Theorem 18. *Let $\mathcal{A}\left(\overline{v}, \epsilon, \delta, \overrightarrow{V}\right)$ be an algorithm that is given parameters $\overline{v}, \epsilon, \delta$ and possibly an additional set of parameters denoted $\overrightarrow{V}$, for which the following holds.*

1. *If $\overline{v} \in [v/4, v]$, then with probability at least $1-\delta$, $\mathcal{A}$ returns a value $\widehat{v}$ such that $\widehat{v} \in (1 \pm \epsilon)v$.*
2. *If $\overline{v} > v$, then $\mathcal{A}$ returns a value $\widehat{v}$, such that with probability at least $\epsilon/4$, $\widehat{v} \leq (1+\epsilon)v$.*
3. *The expected running time of $\mathcal{A}$, denoted $E_{rt}\left(\mathcal{A}\left(\overline{v}, \epsilon, \delta, \overrightarrow{V}\right)\right)$, is monotonically non-increasing with $\overline{v}$ and furthermore, if $\overline{v} < v$, then $E_{rt}\left(\mathcal{A}\left(\overline{v}, \epsilon, \delta, \overrightarrow{V}\right)\right) \leq E_{rt}\left(\mathcal{A}\left(v, \epsilon, \delta, \overrightarrow{V}\right)\right) \cdot (v/\overline{v})^{\ell}$ for some constant $\ell > 0$.*
4. *The maximal running time of $\mathcal{A}$ is D.*

Then there exists an algorithm $\mathcal{A}'$ that, when given an upper bound B on v, a parameter ϵ and a set of parameters $\overrightarrow{V}$, returns a value X such that the following holds.

1. *$\mathcal{A}'(B, \epsilon, \overrightarrow{V})$ returns a value X such that $X \in (1 \pm \epsilon)v$ with probability at least $4/5$.*
2. *The expected running time of $\mathcal{A}'\left(B, \epsilon, \overrightarrow{V}\right)$ is $E_{rt}\left(\mathcal{A}\left(v, \epsilon, \delta, \overrightarrow{V}\right)\right) \cdot \mathrm{poly}(\log B, 1/\epsilon, \ell)$ for $\delta = \Theta\left(\frac{\epsilon}{2^{\ell}(\ell + \log\log(B))}\right)$.*
3. *The maximal running time of $\mathcal{A}'$ is $D \cdot \mathrm{poly}(\log B, 1/\epsilon, \ell)$.*

The algorithm $\mathcal{A}'$ referred to by the theorem is provided in Figure 5. We note that the algorithm and the proof of Theorem 18 are a direct generalization of the search algorithm Estimate and the proof of Theorem 12 in [15]. However, this generalization may be useful as a "black box" in future work. We assume that $\epsilon \leq 1/4$, since otherwise we can simply set ϵ to $1/4$.

$\mathcal{A}'(\mathcal{A}, B, \epsilon, \overrightarrow{V})$

1. Let $\widetilde{v} = B$.
2. Let $\delta' = 1/(5 \cdot 2^{\ell})$.
3. While $\widetilde{v} \geq 1$ do:
   (a) For $\overline{v} = B, B/2, \ldots, \widetilde{v}$ do:
      i. Let $r = (4/\epsilon) \cdot \ln(2\log^2(B)/\delta')$ and let $\delta = \delta'/(2r)$.
      ii. Let $X_{\overline{v}}$ be the minimum value returned over r invocations of $\mathcal{A}\left(\overline{v}, \epsilon, \delta, \overrightarrow{V}\right)$.
      iii. If $X_{\overline{v}} \geq (1+\epsilon)\overline{v}$ then **return** $\overline{v}$.
   (b) Let $\widetilde{v} = \widetilde{v}/2$.
4. **Return** `fail`.

Figure 5: The search algorithm.

The following definition will be useful in the proof of the theorem.

Definition 19. *We say that a value X is a good estimate of v if $X \in (1 \pm \epsilon)v$.*

Proof of Theorem 18: Our search algorithm has two nested loops running with decreasing values of "guesses" for the value of v. The outer loop runs with $\widetilde{v}$, which is our current guess for the value of v, and the purpose of the inner for loop is to enhance the success probability of the algorithm when $\widetilde{v}$ "passes" the good guess of v and runs with values smaller than $v/4$. We provide the full details subsequently, and start by considering only the outer while loop. Namely, imagine for now that instead of the for loop in Step 3a, we have the command "Let $\overline{v} = \widetilde{v}$" and the rest of the algorithm is as described in Figure 5.

First consider iterations of the while loop for which $\widetilde{v} > v$. By Item 2 in the properties of Algorithm $\mathcal{A}$, for values $\widetilde{v}$ such that $\widetilde{v} > v$, the probability that $\mathcal{A}\left(\overline{v}, \epsilon, \delta, \overrightarrow{V}\right)$ returns a value such that $\widehat{v} > (1+\epsilon)v$ is at most $1-\epsilon/4$. Hence, the probability that the minimum value returned over r invocations, that is, $X_{\widetilde{v}}$, satisfies $X_{\widetilde{v}} > (1+\epsilon)v$, is at most $(1-\epsilon/4)^r = (1-\epsilon/4)^{(4/\epsilon)\cdot\ln(2\log^2(B)/\delta')} < \frac{\delta'}{2\log^2(B)}$. It follows that for each value $\widetilde{v} > v$, with probability at least $1 - \frac{\delta'}{2\log^2(B)}$,

$$X_{\widetilde{v}} < (1+\epsilon)v < (1+\epsilon)\widetilde{v}.$$

This implies that for each value $\widetilde{v} > v$, with probability at least $1 - \frac{\delta'}{2\log^2(B)}$, the algorithm $\mathcal{A}'$ will continue to run with $\widetilde{v} = \widetilde{v}/2$.

Now consider values of $\widetilde{v}$ such that $\widetilde{v} \in [v/4, v]$. By Item 1 in the properties of Algorithm $\mathcal{A}$, if $\widetilde{v} \in [v/4, v]$, then with probability at least $1 - \delta'$, $\mathcal{A}\left(\overline{v}, \epsilon, \delta, \overrightarrow{V}\right)$ returns a value $\widehat{v}$, such that $\widehat{v} \in (1 \pm \epsilon)v$. By the setting $\delta = \delta'/(2r)$, and by taking a union bound, with probability at least $1 - \delta'/2$, all the r invocations return a good estimate of v, implying that $X_{\widetilde{v}} \in (1 \pm \epsilon)v$ (i.e., $X_{\widetilde{v}}$ is a good estimate of v. Note that in such a case,

$$X_{\widetilde{v}} \geq (1-\epsilon)v \geq (1-\epsilon) \cdot 2\widetilde{v} \geq (1+\epsilon)\widetilde{v}.$$

(Here we have used the assumption that $\epsilon \leq 1/4$.) Hence, once $\mathcal{A}'$ reaches a value $\widetilde{v} \in [v/4, v/2]$, with probability at least $1 - \delta'/2$, it returns a value $X_{\widetilde{v}}$ that is a good estimate of v.

Finally, if $\mathcal{A}'$ reaches values $\overline{v} < v/4$, we no longer have a guarantee on the probability that $\mathcal{A}$ returns a value $\widehat{v}$ or on the quality of the estimate $\widehat{v}$. Hence, we also have an inner for loop so that whenever we halve the guess $\widetilde{v}$ we first run with all the values $B, \ldots, \widetilde{v}$. This ensures that even if the algorithm did not return a value when running with a good guess $\widetilde{v} \in [v/4, v/2]$, we can still bound the probability that it will continue to run with decreasing values of $\widetilde{v}$.

We now consider the original algorithm with both the outer while loop and the inner for loop as described in Figure 5. There are at most $\log(B)$ invocations of the while loop with values $\widetilde{v} > v$, implying that there are at most $\log^2(B)$ invocations of the for loop in Step 3(a)i with a value $\overline{v} > v$. Therefore, by the above analysis, the probability that $\mathcal{A}'$ returns a value that is not a good estimate of v in these invocations is at most $\delta'/2$. Hence, with probability at least $1 - \delta'/2$ we will reach an invocation in which $\widetilde{v} \in [v/4, v/2]$ and $\overline{v} \in [v/4, v/2]$, for which with probability at least $1 - \delta'/2$ the algorithm $\mathcal{A}$ returns a value $X_{\overline{v}}$ that is a good estimate of v. By taking a union bound and by the setting of δ', the algorithm $\mathcal{A}'$ returns a value that is a good estimate of v with probability at least $1 - \delta' > 4/5$.

We turn to analyze the running time of $\mathcal{A}'$. By Item 3, for values of $\overline{v}$ such that $\overline{v} \geq v$, the expected running time of $\mathcal{A}\left(\overline{v}, \epsilon, \delta, \overrightarrow{V}\right)$ is $E_{rt}\left(\mathcal{A}\left(v, \epsilon, \delta, \overrightarrow{V}\right)\right)$. By the above analysis, once $\mathcal{A}'$ reaches a value $\widetilde{v} < v/4$, since we run with all values $\overline{v} = B, \ldots, \widetilde{v}$, the probability that the algorithm will halve the guess to $\widetilde{v} = \widetilde{v}/2$ is at most δ'. Hence, the probability of invoking $\mathcal{A}$ with

a value $\overline{v} = v/2^z$ is at most $(\delta')^z$. By Item 3 in the properties of Algorithm $\mathcal{A}$, when running with values $\overline{v} < v$, it holds that $E_{rt}\left(\mathcal{A}\left(\overline{v}, \epsilon, \delta, \overrightarrow{V}\right)\right) \leq E_{rt}\left(\mathcal{A}\left(v, \epsilon, \delta, \overrightarrow{V}\right)\right) \cdot (v/\overline{v})^{\ell}$. Therefore, the expected running time of $\mathcal{A}'$ is upper bounded by

$$\log^2(B) \cdot r \cdot E_{rt}\left(\mathcal{A}\left(v, \epsilon, \delta, \overrightarrow{V}\right)\right)) + \sum_{z=1}^{\log B} (\delta')^z \cdot 2^{\ell \cdot z} \cdot E_{rt}\left(\mathcal{A}\left(v, \epsilon, \delta, \overrightarrow{V}\right)\right)$$
$$= \quad E_{rt}\left(\mathcal{A}\left(v, \epsilon, \delta, \overrightarrow{V}\right)\right) \cdot \mathrm{poly}(\log B, 1/\epsilon, \ell) \ ,$$

where the first term is due to the invocations in which $\overline{v} \geq v$ and the second is due to invocations in which $\overline{v} < v$, and the equality is due to the setting of $\delta' < 1/2^{\ell}$. Therefore, Item 2 of the theorem holds. The proof of Item 3 is immediate. □

The following is a corollary of Theorem 10 and Theorem 18 and is a restatement of Theorem 1.

Corollary 20. *There exists an algorithm that, given n, $k \geq 3$, and query access to a graph G, returns a value X such that $X \in (1 \pm \epsilon)C_k$ with probability at least $2/3$. The expected running time of the algorithm is $O\left(\frac{n}{C_k^{1/k}} + \frac{m^{k/2}}{C_k \cdot (k-2)!}\right) \cdot \mathrm{poly}(\log n, 1/\epsilon, k)$, and its expected query complexity is $O\left(\frac{n}{C_k^{1/k}} + \min\left\{m, \frac{m^{k/2}}{C_k \cdot (k-2)!}\right\}\right) \cdot \mathrm{poly}(\log n, 1/\epsilon, k)$.*

Proof. We start by obtaining an estimate $\overline{m}$ of m such that with probability at least $1 - \min\{1/n, 1/m^{k/2}\}$ it holds that $\overline{m} \in [(1 - \epsilon/4)m, m]$. This can be done by invoking the algorithm of Goldreich and Ron [23] for estimating the number of edges $\Theta(\log(n^{2k}))$ times and taking the median value returned. Next we invoke $\mathcal{A}'(\mathsf{Approximate\text{-}cliques}, B = \min\{n^k, \overline{m}^{k/2}\}, \epsilon, \overrightarrow{V})$ with $\overrightarrow{V} = (n, k, \overline{m})$ and return the value X returned by $\mathcal{A}'$.

By Theorem 10, if $\overline{m} \in [(1-\epsilon/5)m, m]$, then Approximate-cliques satisfies the conditions required from Algorithm $\mathcal{A}$ in Theorem 18, with $\ell = 2$ and $D = n + m^{k/2}$ (since this is the maximal running time when $\overline{C}_k = 1$). Hence, by Theorem 18 and by the union bound, $\mathcal{A}'$ returns a value X such that with probability at least $4/5 - 1/m^k > 2/3$, it holds that $X \in (1 \pm \epsilon)C_k$.

By [23], the first step of approximating the number of edges $\overline{m}$ with success probability at least $1 - \min\{1/n, 1/m^k\}$ takes time $O\left(\frac{n}{\sqrt{m}}\right) \cdot \mathrm{poly}(\log n, 1/\epsilon, k)$ which by Claim 3 is at most $O\left(\frac{n}{C_k^{1/k}}\right) \cdot \mathrm{poly}(\log n, 1/\epsilon, k)$. When $\overline{m} \in [1 - (\epsilon/5)m, m]$, by Item 3 in Theorem 10, and by Item 2 in the properties of Algorithm $\mathcal{A}'$, the expected running time of the algorithm is

$$O\left(\frac{n}{C_k^{1/k}} + \frac{m^{k/2}}{C_k}\right) \cdot \mathrm{poly}\left(\log n, 1/\epsilon, k\right) \ .$$

Since $\overline{m} \notin [(1 - \epsilon/5)m, m]$ with probability at most $\min\{1/n, 1/m^{k/2}\}$ and the maximal running time of Approximate-cliques is at most $n + m^{k/2}$ up to $\mathrm{poly}(\log n, 1/\epsilon, k)$ factors, this event does not effect the expected query complexity.

By Item 3 in Theorem 10, Algorithm Approximate-cliques never performs more than $O\left(\frac{n}{C_k^{1/k}}\right) \cdot \mathrm{poly}(\log n, 1/\epsilon, \log(1/\delta), k) + \min\{m, \overline{m}\}$ queries. It follows that the expected query complexity is

$$O\left(\frac{n}{C_k^{1/k}} + \min\left\{m, \frac{m^{k/2}}{C_k}\right\}\right) \cdot \mathrm{poly}\left(\log n, 1/\epsilon, k\right) \ ,$$

as claimed ■

4 A lower bound

We now turn to prove an $\Omega\left(\frac{n}{C_k^{1/k}} + \min\left\{m, \frac{m^{k/2}}{C_k}\right\}\right)$ lower bound on the number of necessary queries for estimating the number of k-cliques in a graph. Our lower bound matches the upper bound up to a factor of $(c \cdot k)^k$ for some small constant c and up to polynomial factors of $\log n$ and $1/\overline{\epsilon}$. We prove that the lower bound holds for any multiplicative algorithm for estimating the number of k-cliques. That is, any algorithm that for a graph G over n vertices and m edges returns a value $\widetilde{C}_k$ such that $C_k/F \leq \widetilde{C}_k \leq F \cdot C_k$ for some predetermined approximation factor F. The lower bound follows from the next two theorems (where in both the allowed queries are degree queries, neighbor queries and pair-queries).

Theorem 21. *Any multiplicative approximation algorithm for the number of k-cliques in a graph must perform $\Omega\left(\frac{n}{C_k^{1/k}}\right)$ queries.*

Theorem 22. *Any multiplicative approximation algorithm for the number of k-cliques in a graph must perform $\Omega\left(\min\left\{m, \frac{m^{k/2}}{C_k \cdot (c \cdot k)^k}\right\}\right)$ queries for some constant c.*

The proof of the first theorem is simple and we provide it here for the sake of completeness (it is very similar to [15, Thm. 14]).

Proof of Theorem 21: For any two values n and C_k so that $C_k \leq \binom{n}{k}$ we describe a graph G_1 and a family of graphs $\mathcal{G}_2$ for which the following holds. The graph G_1 has n vertices and no k-cliques, and every graph in the family $\mathcal{G}_2$ has n vertices and $\Theta(C_k)$ k-cliques. The graph G_1 consists of n isolated vertices. Each graph in the family $\mathcal{G}_2$ consists of a clique over r vertices, where r is the integer that minimizes $\left|\binom{r}{k} - C_k\right|$, and $n - r$ isolated vertices.[3] The only difference between the different graphs in $\mathcal{G}_2$ is in the labeling of the vertices (where there is a graph for each labeling). Clearly, in order to distinguish with constant probability between G_1 and a random graph from the family $\mathcal{G}_2$, any algorithm must sample a vertex from the clique. Since the probability of hitting a clique vertex in a graph from $\mathcal{G}_2$ is $\frac{r}{n}$ and $r = \Theta(C_k^{1/k})$, at least $\Omega\left(n/C_k^{1/k}\right)$ queries are needed in order to distinguish with high constant probability between a graph from G_1 and a random graph from $\mathcal{G}_2$. □

The proof of Theorem 22 is a bit more involved and builds on the lower bound of Eden et al. [15, Thm. 15] for approximating the number of triangles, which we recall next. Eden et al. prove that $\Omega\left(\min\{\frac{m^{3/2}}{t}, m\}\right)$ queries are necessary in order to obtain a multiplicative estimate of the number of triangles for graphs with m edges and t triangles. The next claim directly follows from the proof of Thm. 15 in [15].

Claim 23. *For any two values, ℓ and $t \in [1, \ell^{3/2}]$, there exists a graph H_1 and a family of graphs $\mathcal{H}_2$ such that the following holds. The graph H_1 and all graphs in $\mathcal{H}_2$ have $\Theta(\sqrt{\ell})$ vertices and $\Theta(\ell)$ edges. The graph H_1 contains no triangles, and the graphs in $\mathcal{H}_2$ all have $\Theta(t)$ triangles. In order to distinguish between the graph H_1 and a random graph from $\mathcal{H}_2$ with probability at least $2/3$, $\Omega\left(\min\left\{\frac{m^{3/2}}{t}, m\right\}\right)$ queries are necessary.*

Our proof builds upon Claim 23.

Proof of Theorem 22: For every set of values n, m and C_k, we define a graph G_1 and a family of graphs $\mathcal{G}_2$ such that the following holds. The graph G_1 and all the graphs in $\mathcal{G}_2$ have n vertices

[3] If one wants to avoid isolated vertices, then the isolated vertices in G_1 and in $\mathcal{G}_2$ can be replaced by, say, a path.

and $\Theta(m)$ edges. The graph G_1 has no k-cliques, and every graph in $\mathcal{G}_2$ has $\Theta(C_k)$ k-cliques. We consider the following ranges of C_k separately:

- $C_k > \left(\frac{\sqrt{m}}{k}\right)^{k-2}$
- $C_k \in \left[1, \left(\frac{\sqrt{m}}{k}\right)^{k-2}\right]$.

We start by considering the case $C_k > \left(\frac{\sqrt{m}}{k}\right)^{k-2}$. Let $r = \lfloor n_k/(\sqrt{m}/k)^{k-3} \rfloor$, so that by the assumption on C_k it holds that $r \geq \sqrt{m}/k$. By Claim 3, C_k is upper bounded by $m \cdot \binom{\sqrt{m}}{k-2}$, implying that $r \leq e^{k-2} \cdot m^{3/2}$. Let $t = \max\{\min\{r, \binom{\sqrt{m}}{3}\}, \sqrt{m}\}$ so that if $r \leq \sqrt{m}$ then $t = \sqrt{m}$, if $r > \binom{\sqrt{m}}{3}$ then $t = \binom{\sqrt{m}}{3}$, and otherwise $t = r$. Finally, let H_1 and $\mathcal{H}_2$ be as described in Claim 23 with $\ell = m$ and t as just defined. The graph G_1 has the following components. The subgraph H_1 (with $\sqrt{\ell} = \sqrt{m}$ vertices and no triangles), and additional $k-3$ sets with $\frac{\sqrt{m}}{k-2}$ vertices in each, denoted $A_1, \ldots, A_{k-3}$. There is a complete bipartite graph between the vertices of H_1 and every set A_i for $i \in [k-3]$, and a complete bipartite graph between any two sets A_i and $A_{i'}$ for $i, i' \in [k-3]$. The remaining vertices are isolated vertices. See Figure 6 for an illustration of the two families.

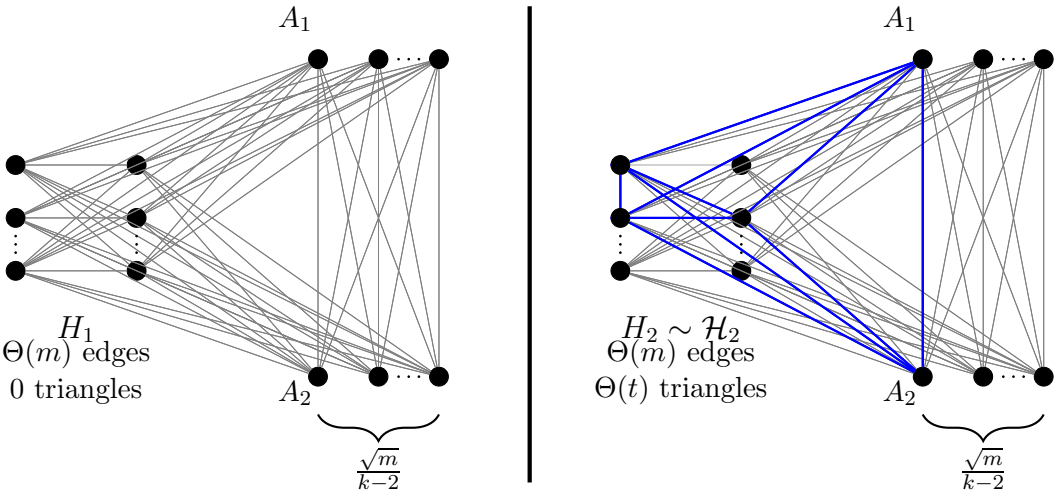

Figure 6: An illustration of the two families for $k = 5$.

The graph G_1 hence has m edges inside H_1 (by Claim 23 and our setting of $\ell = m$), $(k-3) \cdot \frac{m}{(k-2)}$ edges between the vertices of H_1 and the sets A_i and $(k-3) \cdot \frac{m}{(k-2)}$ between the different A_i sets. Hence, the graph G_1 has $\Theta(m)$ edges. Furthermore, since the sets $A_i, i \in [k-3]$ are independent sets, any maximum clique over the subgraph $A_1 \cup \ldots \cup A_{k-3}$ is of size $k-3$, and since the graph H_1 contains no triangles it follows that the graph G_1 contains no k-cliques.

The graphs in the family $\mathcal{G}_2$ have the same basic structure, and the only difference is that in every graph in $\mathcal{G}_2$ we use a subgraph $H_2 \sim \mathcal{H}_2$ rather than the graph H_1. Since the subgraph over $A_1 \cup \ldots \cup A_{k-3}$ contains exactly $(\sqrt{m}/(k-2))^{k-3}$ cliques of size $k-3$ and no larger cliques, the number of k-cliques in the graphs of the family $\mathcal{G}_2$ is $t \cdot (\sqrt{m}/(k-2))^{k-3}$, which is of order C_k up to a factor of e^{k-2}. By the above settings, C_k is of order $t \cdot (\sqrt{m}/k)^{k-3}$ (up to e^{k-2}), and therefore, by Claim 23, $\Omega\left(\min\left\{m, \frac{m^{3/2}}{t}\right\}\right) = \Omega\left(\frac{\sqrt{m}^{k/2}}{C_k \cdot (e \cdot k)^k}\right)$ queries are needed in order to

distinguish the graph G_1 from a random graph in the family $\mathcal{G}_2$. This completes he proof for the case that $C_k > (\sqrt{m}/k)^{k-2}$.

We now prove that when $C_k \in [1, (\sqrt{m}/k)^{k-2}]$, then $\Omega(m)$ queries are necessary for any multiplicative algorithm for approximating the number of k-cliques. Let r and t be such that $C_k = t \cdot (\sqrt{m}/k)^r$ for some integer r, implying that $t \in [1, \sqrt{m}/k]$. As before, let H_1 and $\mathcal{H}_2$ be as described in Claim 23 for $\ell = m$ and the current setting of t.

The graph G_1 has the following components. Similarly to the previous case, the basic component is the subgraph H_1, but now there are only r additional sets with $\frac{\sqrt{m}}{k-2}$ vertices in each, denoted $A_1, \ldots, A_r$ and the remaining vertices are isolated vertices. There is a complete bipartite graph between every set A_i and the vertices of H_1 for every $i \in [r]$ and between every two sets $A_i, A_{i'}$ for $i, i' \in [r]$. Therefore, the graph G_1 has no k-cliques and $\Theta(m)$ edges ($\Theta(m)$ edges inside H_1 and additional $\Theta(m \cdot (\ell/k)^2)$ incident to the vertices of the A_i sets). The graphs in $\mathcal{G}_2$ are identical except that we now use a subgraph $H_2 \sim \mathcal{H}_2$, so that we get that the number of triangles in H_2 is $\Theta(t)$ for t as set above. By the setting of t it holds that $m < \sqrt{m}^{3/2}/t$, and it follows from Claim 23 that $\Theta(m)$ queries are required for any algorithm in order to distinguish between the graph G_1 and a random graph from $\mathcal{G}_2$. This completes the proof for the case $C_k \in [1, (\sqrt{m}/k)^{k-2}]$. □

References

[1] A. Abboud, A. Backurs, and V. V. Williams. If the current clique algorithms are optimal, so is valiant's parser. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 98–117, 2015.

[2] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the Symposium on Principles of Database Systems (PODS)*, pages 5–14, 2012.

[3] H. Avron. Counting triangles in large graphs using randomized matrix trace estimation. In *Workshop on Large-scale Data Mining: Theory and Applications (LDMTA)*, volume 1, 2010.

[4] L. Becchetti, P. Boldi, C. Castillo, and A. Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 16–24, 2008.

[5] J. W. Berry, B. Hendrickson, R. A. LaViolette, and C. A. Phillips. Tolerating the Community Detection Resolution Limit with Edge Weighting. *Physical Review E*, 83(5), May 2011.

[6] R. S. Burt. Structural holes and good ideas. *American Journal of Sociology*, 110(2):349–399, 2004.

[7] B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM Journal on Computing*, 34(6):1370–1379, 2005.

[8] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, pages 493–507, 1952.

[9] N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on Computing*, 14(1):210–223, 1985.

[10] S. Chu and J. Cheng. Triangle listing in massive networks and its applications. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 672–680, 2011.

[11] J. S. Coleman. Social capital in the creation of human capital. *American Journal of Sociology*, 94:S95–S120, 1988.

[12] A. Czumaj, F. Ergün, L. Fortnow, A. Magen, I. Newman, R. Rubinfeld, and C. Sohler. Approximating the weight of the Euclidean minimum spanning tree in sublinear time. *SIAM Journal on Computing*, 35(1):91–109, 2005.

[13] A. Czumaj and C. Sohler. Estimating the weight of metric minimum spanning trees in sublinear time. *SIAM Journal on Computing*, 39(3):904–922, 2009.

[14] J. P. Eckmann and E. Moses. Curvature of co-links uncovers hidden thematic layers in the World Wide Web. *Proceedings of the National Academy of Sciences*, 99(9):5825–5829, 2002.

[15] T. Eden, A. Levi, D. Ron, and C Seshadhri. Approximately counting triangles in sublinear time. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 614–633, 2015.

[16] T. Eden, D. Ron, and C Seshadhri. Sublinear time estimation of degree distribution moments: The arboricity connection. In *Proceedings of International Colloquium on Automata, Languages, and Programming (ICALP)*, 2017. To appear.

[17] T. Eden and W. Rosenbaum. On sampling edges almost uniformly. *arXiv preprint arXiv:1706.09748*, 2017.

[18] F. Eisenbrand and F. Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1-3):57–67, 2004.

[19] David Eppstein, Maarten Löffler, and Darren Strash. Listing all maximal cliques in large sparse real-world graphs. *ACM Journal of Experimental Algorithmics*, 18, 2013.

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

[21] I. Finocchi, M. Finocchi, and E. G. Fusco. Clique counting in mapreduce: Algorithms and experiments. *ACM Journal of Experimental Algorithmics*, 20, 2015.

[22] B. Foucault Welles, A. Van Devender, and N. Contractor. Is a friend a friend?: Investigating the structure of friendship networks in virtual worlds. In *CHI Extended Abstracts on Human Factors in Computing Systems*, pages 4027–4032, 2010.

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

[24] M. Gonen, D. Ron, and Y. Shavitt. Counting stars and other small subgraphs in sublinear-time. *SIAM Journal on Discrete Mathematics*, 25(3):1365–1411, 2011.

[25] A. Hassidim, J. A. Kelner, H. N. Nguyen, and K. Onak. Local graph partitions for approximation and testing. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 22–31, 2009.

[26] P. W. Holland and S. Leinhardt. A method for detecting structure in sociometric data. *American Journal of Sociology*, 76:492–513, 1970.

[27] M. O. Jackson, T. Rodriguez-Barraquer, and X. Tan. Social capital and social quilts: Network patterns of favor exchange. *American Economic Review*, 102(5):1857?1897, 2012.

[28] H. Jowhari and M. Ghodsi. New streaming algorithms for counting triangles in graphs. In *Proceedings of the International Conference Computing and Combinatorics (COCOON)*, pages 710–716. Springer, 2005.

[29] D. M. Kane, K. Mehlhorn, T. Sauerwald, and H. Sun. Counting arbitrary subgraphs in data streams. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 598–609, 2012.

[30] M. N. Kolountzakis, G. L. Miller, R. Peng, and C. E. Tsourakakis. Efficient triangle counting in large graphs via degree-based vertex partitioning. *Internet Mathematics*, 8(1-2):161–185, 2012.

[31] G. Manoussakis. Listing all maximal cliques in sparse graphs in optimal time. *CoRR*, abs/1501.01819, 2015.

[32] S. Marko and D. Ron. Approximating the distance to properties in bounded-degree and general sparse graphs. *ACM Transactions on Algorithms*, 5(2):22, 2009.

[33] G. Marsaglia, W. W. Tsang, J. Wang, et al. Fast generation of discrete random variables. *Journal of Statistical Software*, 11(3):1–11, 2004.

[34] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[35] J. Neetil and S. Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

[36] H. N. Nguyen and K. Onak. Constant-time approximation algorithms via local improvements. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 327–336, 2008.

[37] K. Onak, D. Ron, M. Rosen, and R. Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proceedings of the Symposium on Discrete Algorithms (SODA)*, pages 1123–1131, 2012.

[38] M. Parnas and D. Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science*, 381(1-3):183–196, 2007.

[39] A. Portes. Social capital: Its origins and applications in modern sociology. *Knowledge and Social Capital*, pages 43–67, 2000.

[40] T. Schank and D. Wagner. Approximating clustering coefficient and transitivity. *Journal of Graph Algorithms and Applications*, 9:265–275, 2005.

[41] T. Schank and D. Wagner. Finding, counting and listing all triangles in large graphs, an experimental study. In *Experimental and Efficient Algorithms*, pages 606–609. 2005.

[42] C. Seshadhri, T. G. Kolda, and A. Pinar. Community structure and scale-free collections of Erdös-Rényi graphs. *Physical Review E*, 85(5):056109, May 2012.

[43] C. Seshadhri, A. Pinar, and T. G. Kolda. Fast triangle counting through wedge sampling. In *International Conference on Data Mining (ICDM)*, 2013.

[44] S. Suri and S. Vassilvitskii. Counting triangles and the curse of the last reducer. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 607–614, 2011.

[45] K. Tangwongsan, A. Pavan, and S. Tirthapura. Parallel triangle counting in massive streaming graphs. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, 2013.

[46] C. E. Tsourakakis. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *International Conference on Data Mining (ICDM)*, pages 608–617, 2008.

[47] C. E. Tsourakakis. The k-clique densest subgraph problem. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 1122–1132, 2015.

[48] C. E. Tsourakakis, U. Kang, G.L. Miller, and C. Faloutsos. Doulion: counting triangles in massive graphs with a coin. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 837–846, 2009.

[49] C. E. Tsourakakis, M. N. Kolountzakis, and G. L. Miller. Triangle sparsifiers. *Journal of Graph Algorithms and Applications*, 15(6):703–726, 2011.

[50] V. Vassilevska. Efficient algorithms for clique problems. *Information Processing Letters*, 109(4):254–257, 2009.

[51] A. J. Walker. New fast method for generating discrete random numbers with arbitrary frequency distributions. *Electronics Letters*, 10(8):127–128, 1974.

[52] A. J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software*, 3(3):253–256, 1977.

[53] Y. Yoshida, M. Yamamoto, and H. Ito. An improved constant-time approximation algorithm for maximum. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 225–234, 2009.