A Simple Sublinear-Time Algorithm for Counting Arbitrary Subgraphs via Edge Sampling

Sepehr Assadi[*] Michael Kapralov[†] Sanjeev Khanna[‡]

Abstract

In the subgraph counting problem, we are given a (large) input graph $G(V, E)$ and a (small) target graph H (e.g., a triangle); the goal is to estimate the number of occurrences of H in G. Our focus here is on designing *sublinear-time* algorithms for approximately computing number of occurrences of H in G in the setting where the algorithm is given *query access* to G. This problem has been studied in several recent papers which primarily focused on specific families of graphs H such as triangles, cliques, and stars. However, not much is known about approximate counting of arbitrary graphs H in the literature. This is in sharp contrast to the closely related subgraph enumeration problem that has received significant attention in the database community as the database join problem. The AGM bound shows that the maximum number of occurrences of any arbitrary subgraph H in a graph G with m edges is $O(m^{\rho(H)})$, where $\rho(H)$ is the *fractional edge-cover* of H, and enumeration algorithms with matching runtime are known for any H.

In this work, we bridge this gap between the subgraph counting and subgraph enumeration problems by designing a simple sublinear-time algorithm that can estimate the number of occurrences of any arbitrary graph H in G, denoted by $\#H$, to within a $(1 \pm \varepsilon)$-approximation with high probability in $O(\frac{m^{\rho(H)}}{\#H}) \cdot \text{poly}(\log n, 1/\varepsilon)$ time. Our algorithm is allowed the standard set of queries for general graphs, namely degree queries, pair queries and neighbor queries, plus an additional edge-sample query that returns an edge chosen uniformly at random. The performance of our algorithm matches those of Eden *et al.* [FOCS 2015, STOC 2018] for counting triangles and cliques and extend them to all choices of subgraph H under the additional assumption of edge-sample queries. We further show that our algorithm works for a more general version of the estimation problem where edges of G and H have colors, which corresponds to the database join size estimation problem. For this slightly more general setting, we also establish a matching lower bound for any choice of subgraph H.

[*] Department of Computer and Information Science, University of Pennsylvania. Supported in part by the National Science Foundation grant CCF-1617851. Email: `sassadi@cis.upenn.edu`.

[†] School of Computer and Communication Sciences, EPFL. Email: `michael.kapralov@epfl.ch`. Supported in part by ERC Starting Grant 759471.

[‡] Department of Computer and Information Science, University of Pennsylvania. Supported in part by the National Science Foundation grants CCF-1617851 and CCF-1763514. Email: `sanjeev@cis.upenn.edu`.

1 Introduction

Counting (small) subgraphs in massive graphs is a fundamental algorithmic problem, with a wide range of applications in bioinformatics, social network analysis, spam detection and graph databases (see, e.g. [10, 13, 40]). In social network analysis, the ratio of the number of triangles in a network to the number of length 2 paths (known as the clustering coefficient) as well as subgraph counts for larger subgraphs H have been proposed as important metrics for analyzing massive networks [47]. Similarly, motif counting are popular method for analyzing protein-protein interaction networks in bioinformatics (e.g., [40]). In this paper we consider designing efficient algorithms for this task.

Formally, we consider the following problem: Given a (large) graph $G(V, E)$ with m edges and a (small) subgraph $H(V_H, E_H)$ (e.g., a triangle) and a precision parameter $\varepsilon \in (0, 1)$, output a $(1 \pm \varepsilon)$-approximation[1] to the number of occurrences of H in G. Our goal is to design an algorithm that runs in time *sublinear* in the number m of edges of G, and in particular makes a sublinear number of the following types of queries to the graph G:

- **Degree query** v**:** the degree d_v of any vertex $v \in V$;
- **Neighbor query** (v, i)**:** what vertex is the i-th neighbor of the vertex $v \in V$ for $i \leq d_v$;
- **Pair query** (u, v)**:** test for pair of vertices $u, v \in V$, whether or not (u, v) is an edge in E.
- **Edge-sample query:** sample an edge e uniformly at random from E.

The first three queries are the standard baseline queries (see Chapter 10 of Goldreich's book [26]) assumed by nearly all sublinear time algorithms for counting small subgraphs such as triangles or cliques [19, 21] (see [28] for the necessity of using pair queries for counting subgraphs beside stars). The last query is somewhat less standard but is also considered in the literature prior to our work, for example in [2] for counting stars in sublinear time, and in [22] in the context of lower bounds for subgraph counting problems.

1.1 Our Contributions

For the sake of clarity, we suppress any dependencies on the approximation parameter ε, $\log n$-terms, and the size of graph H, using the $O^*(\cdot)$ notation. Our results are parameterized by the *fractional edge-cover number* of the subgraph H (see Section 3 for the formal definition). Our goal in this paper is to approximately compute the number of occurrences $\#H$ of H in G, formally defined as number of subgraphs H' of G such that H and H' are isomorphic to each other.

Theorem 1. *There exists a randomized algorithm that given a precision parameter $\varepsilon \in (0, 1)$, a subgraph H, and a query access to the input graph G, with high probability outputs a $(1 \pm \varepsilon)$ approximation to the number of occurrences of H in G, denoted by $\#H$, using:*

$$O^*\Big(\min\Big\{m, \frac{m^{\rho(H)}}{\#H}\Big\}\Big) \text{ queries and } O^*\Big(\frac{m^{\rho(H)}}{\#H}\Big) \text{ time.}$$

The algorithm uses degree queries, neighbor queries, pair queries, and edge-sample queries.

[1] Throughout, we say that a is a $(1 \pm \varepsilon)$ approximation to b iff $(1 - \varepsilon) \cdot b \leq a \leq (1 + \varepsilon) \cdot b$.

Since the fractional edge-cover number of any k-clique K_k is $k/2$, as a corollary of Theorem 1, we obtain sublinear algorithms for counting triangles, and in general k-cliques using

$$O^*\left(\min\left\{m, \frac{m\sqrt{m}}{\#K_3}\right\}\right) \text{ and } O^*\left(\min\left\{m, \frac{m^{k/2}}{\#K_k}\right\}\right),$$

queries respectively. These bounds match the previous results of Eden *et al.* [19, 21] modulo an additive term of $O^*(\frac{n}{(\#K_3)^{1/3}})$ for triangles in [19] and $O^*(\frac{n}{(\#K_k)^{1/k}})$ for arbitrary cliques in [21] which is needed in the absence of edge-sample queries that are not used by [19, 21]. Our bounds settle a conjecture of Eden and Rosenbaum [22] in the affirmative by showing that one can avoid the aforementioned additive terms depending on n in query complexity by allowing edge-sample queries. We now elaborate more on different aspects of our result in Theorem 1.

AGM Bound and Database Joins. The problem of *enumerating* all occurrences of a graph H in a graph G and, more generally, the database join problem, has been considered extensively in the literature. A fundamental question here is that given a database with m entries (e.g. a graph G with m edges) and a conjunctive query H (e.g. a small graph H), what is the maximum possible size of the output of the query (e.g., number of occurrences of H in G)? The AGM bound of Atserias, Grohe and Marx [4] provides a tight upper bound of $m^{\rho(H)}$ (up to constant factors), where $\rho(H)$ is the fractional edge cover of H. The AGM bound applies to databases with several relations, and the fractional edge cover in question is weighted according to the sizes of the different relations. A similar bound on the number of occurrences of a hypergraph H inside a hypergraph G with m hyperedges was proved earlier by Friedgut and Kahn [25], and the bound for graphs is due to Alon [3]. Recent work of Ngo et al. [41] gave algorithms for evaluating database joins in time bounded by worst case output size for a database with the same number of entries. When instantied for the subgraph enumeration problem, their result gives an algorithm for enumerating all occurrences of H in a graph G with m edges in time $O(m^{\rho(H)})$.

Our Theorem 1 is directly motivated by the connection between subgraph counting and subgraph enumeration problems and the AGM bound. In particular, Theorem 1 provides a natural analogue of AGM bound for estimation algorithms by stating that if the number of occurrences H is $\#H \leq m^{\rho(H)}$, then a $(1 \pm \varepsilon)$-approximation to $\#H$ can be obtained in $O^*(\frac{m^{\rho(H)}}{\#H})$ time. Additionally, as we show in Section 4.3, Theorem 1 can be easily extended to the more general problem of database join size estimation (for binary relations). This problem corresponds to a subgraph counting problem in which the graphs G and H are both *edge-colored* and our goal is to count the number of copies of H in G with the same colors on edges. Our algorithm can solve this problem also in $O^*(\frac{m^{\rho(H)}}{\#H_c})$ time where $\#H_c$ denotes the number of copies of H with the same colors in G.

Optimality of Our Bounds. Our algorithm in Theorem 1 is optimal from different points of view. Firstly, by a lower bound of [22] (building on [19, 21]), the bounds achieved by our algorithm when H is any k-clique is optimal among all algorithms with the same query access (including the edge-sample query). In Theorem 3, we further prove a lower bound showing that for *odd cycles* as well, the bounds achieved by Theorem 1 are optimal. These results hence suggest that Theorem 1 is *existentially optimal*: there exists several natural choices for H such that Theorem 1 achieves the optimal bounds. However, there also exist choices of H for which the bounds in Theorem 1 are suboptimal. In particular, Aliakbarpour *et al.* [2] presented an algorithm for estimating occurrences of any star S_ℓ for $\ell \geq 1$ using $O^*(\frac{m}{(\#S_\ell)^{1/\ell}})$ queries in our query model (including edge-sample queries) which is always at least as good as our bound in Theorem 1, but potentially can be better. On the other hand, we show that our current algorithm, with almost no further modification, in fact achieves this stronger bound *using a different analysis* (see Remark 4.4 for details).

Additionally, as we pointed out before, our algorithm can solve the more general database join size estimation for binary relations, or equivalently the subgraph counting problem with colors on edges. In Theorem 4, we prove that for this more general problem, our algorithm in Theorem 1 indeed achieves optimal bounds for *all choices* of the subgraph H.

Edge-Sample Queries. The edge-sample query that we assume is not part of the standard access model for sublinear algorithms, namely the "general graph" query model (see, e.g. [36]). Nonetheless, we find allowing for this query "natural" owing to the following factors:

Theoretical implementation. Edge sampling queries can be implemented with an $\widetilde{O}(n/\sqrt{m})$ multiplicative overhead in query and time using the recent result of [23], or with an $O(n)$ additive preprocessing time (which is still sublinear in m) by querying degrees of all vertices. Hence, we can focus on designing algorithms by allowing these queries and later replacing them by either of the above implementations in a black-box way at a certain additional cost.

Practical implementation. Edge sampling is a common practice in analyzing social networks [37, 38] or biological networks [1]. Another scenario when random edge sampling is possible is when we can access a random location of the memory that is used to store the graph. To quote [2]: "because edges normally take most of the space for storing graphs, an access to a random memory location where the adjacency list is stored, would readily give a random edge." Hence, assuming edge sampling queries can be considered valid in many scenarios.

Understanding the power of random edge queries. Edge sampling is a critical component of various sublinear time algorithms for graph estimation [2, 19–21, 23]. However, except for [2] that also assumed edge-sample queries, all these other algorithms employ different workarounds to these queries. As we show in this paper, decoupling these workarounds from the rest of the algorithm by allowing edge-sample queries results in considerably simpler and more general algorithms for subgraph counting and is hence worth studying on its own. We also mention that studying the power of edge-sample queries has been cast as an open question in [22] as well.

Implications to Streaming Algorithms. Subgraph counting is also one of the most studied problems in the graph streaming model (see, e.g. [6, 7, 11, 12, 16, 30, 31, 35, 39, 46] and references therein). In this model, the edges of the input graph are presented one by one in a stream; the algorithm makes a single or a small number of passes over the stream and outputs the answer after the last pass. The goal in this model is to minimize the memory used by the algorithm (similar-in-spirit to minimizing the query complexity in our query model).

Our algorithm in Theorem 1 can be directly adapted to the streaming model, resulting in an algorithm for subgraph counting that makes $O(1)$ passes over the stream and uses a memory of size $O^*\Big(\min\Big\{m, \frac{m^{\rho(H)}}{\#H}\Big\}\Big)$. For the case of counting triangles and cliques, the space complexity of our algorithm matches the best known algorithms of McGregor *et al.* [39] and Bera and Chakrabarti [7] which are known to be optimal [7]. To the best of our knowledge, the only previous streaming algorithms for counting arbitrary subgraphs H are those of Kane *et al.* [35] and Bera and Chakrabarti [7] that use, respectively, one pass and $O^*(\frac{m^{2\cdot|E(H)|}}{(\#H)^2})$ space, and two passes and $O^*(\frac{m^{\beta(H)}}{\#H})$ space, where $\beta(H)$ is the *integral* edge-cover number of H. As $\rho(H) \leq \beta(H) \leq |E(H)|$ by definition and $\#H \leq m^{\rho(H)}$ by AGM bound, the space complexity of our algorithm is always at least as good as the ones in [7, 35] but potentially can be much smaller.

1.2 Main Ideas in Our Algorithm

Our starting point is the AGM bound which implies that the number of "potential copies" of H in G is at most $m^{\rho(H)}$. Our goal of estimating $\#H$ then translates to counting how many of these potential copies form an actual copy of H in G. A standard approach at this point is the *Monte Carlo method:* sample a potential copy of H in G *uniformly at random* and check whether it forms an actual copy of H or not; a simple exercise in concentration inequalities then implies that we only need $O(\frac{m^{\rho(H)}}{\#H})$ many *independent* samples to get a good estimate of $\#H$.

This approach however immediately runs into a technical difficulty. Given only a query access to G, it is not at all clear how to sample a potential copy of H from the list of all potential copies. Our first task is then to design a procedure for sampling potential copies of H from G. In order to do so, we again consider the AGM bound and the optimal fractional edge-cover that is used to derive this bound. We first prove a simple structural result that states that an optimal fractional edge-cover of H can be supported only on edges that form a disjoint union of *odd cycles* and *stars* (in H). This allows us to decompose H into a collection of odd cycles and stars and treat any arbitrary subgraph H as a collection of these simpler subgraphs that are suitably connected together.

The above decomposition reduces the task of sampling a potential copy of H to sampling a collection of odd cycles and stars. Sampling an odd cycle C_{2k+1} on $2k+1$ edges is as follows: sample k edges $e_1, \ldots, e_k$ uniformly at random from G; pick one of the endpoints of e_1 and sample a vertex v from the neighborhood of this endpoint uniformly at random. With some additional care, one can show that the tuple $(e_1, \ldots, e_k, v)$ sampled here is enough to identify an odd cycle of length $2k+1$ uniquely. To sample a star C_ℓ with ℓ petals, we sample a vertex v from G with probability proportional to its degree (by sampling a random edge and picking one of the two endpoints uniformly), and then sample ℓ vertices $w_1, \ldots, w_\ell$ from the neighborhood of v. Again, with some care, this allows us to sample a potential copy of a star S_ℓ. We remark that these sampling procedures are related to sampling triangles in [19] and stars in [2]. Finally, to sample a potential copy of H, we simply sample all its odd cycles and stars in the decomposition using the method above. We should note right away that this however does *not* result in a uniformly at random sample of potential copies of H as various parameters of the graph G, in particular degrees of vertices, alter the probability of sampling each potential copy.

The next and paramount step is then how to use the samples above to estimate the value of $\#H$. Obtaining an unbiased estimator of $\#H$ based on these samples is not hard as we can identify the probability each potential copy is sampled with in this process (which is a function of degrees of vertices of the potential copy in G) and reweigh each sample accordingly. Nevertheless, the variance of a vanilla variant of this sampling and reweighing approach is quite large for our purpose. To fix this, we use an idea similar to that of [19] for counting triangles: sample a "partial" potential copy of H first and fix it; sample *multiple* "extensions" of this partial potential copy to a complete potential copy and use the average of estimates based on each extension to reduce the variance. More concretely, this translates to sampling multiple copies of the first cycle for the decomposition and for each sampled cycle, recursively sampling multiple copies of the remainder of H as specified by the decomposition. A careful analysis of this recursive process—which is the main technical part of the paper—allows us to bound the variance of the estimator by $O(m^{\rho(H)}) \cdot (\#H)$. Repeating such an estimator $O(\frac{m^{\rho(H)}}{\#H})$ times independently and taking the average value then gives us a $(1 \pm \varepsilon)$-approximation to $\#H$ by a simple application of Chebyshev's inequality.

1.3 Further Related Work

In addition to the previous work in [2, 19, 21] that are already discussed above, sublinear-time algorithms for estimating subgraph counts and related parameters such as average degree and degree distribution moments have also been studied in [20, 24, 27, 28]. Similarly, sublinear-time algorithms are also studied for estimating other graph parameters such as weight of the minimum spanning tree [14, 17, 18] or size of a maximum matching or a minimum vertex cover [29, 42–44, 50] (this is by no means a comprehensive summary of previous results).

Subgraph counting has also been studied extensively in the graph streaming model (see, e.g. [6, 7, 11, 12, 16, 30–33, 35, 39, 46] and references therein). In this model, the edges of the input graph are presented one by one in a stream; the algorithm makes a single or a small number of passes over the stream and outputs the answer after the last pass. The goal in this model is to minimize the memory used by the algorithm similar-in-spirit to minimizing the query complexity in our query model. However, the streaming algorithms typically require reading the entire graph in the stream which is different from our goal in sublinear-time algorithms.

2 Preliminaries

Notation. For any integer $t \geq 1$, we let $[t] := \{1, \ldots, t\}$. For any event $\mathcal{E}$, $\mathbb{I}(\mathcal{E}) \in \{0, 1\}$ is an indicator denoting whether $\mathcal{E}$ happened or not. For a graph $G(V, E)$, $V(G) := V$ denotes the vertices and $E(G) := E$ denotes the edges. For a vertex $v \in V$, $N(v)$ denotes the neighbors of v, and $d_v := |N(v)|$ denotes the degree of v.

To any edge $e = \{u, v\}$ in G, we assign two directed edges $\vec{e}_1 = (u, v)$ and $\vec{e}_2 = (v, u)$ called the directed copies of e and let $\vec{E}$ be the set of all these directed edges. We also fix a total ordering $\prec$ on vertices whereby for any two vertices $u, v \in V$, $u \prec v$ iff $d_u < d_v$, or $d_u = d_v$ and u appears before v in the lexicographic order. To avoid confusion, we use letters a, b and c to denote the vertices in the subgraph H, and letters u, v and w to denote the vertices of G.

We use the following standard variant of Chebyshev's inequality.

Proposition 2.1. *For any random variable X and integer $t \geq 1$, $\Pr\left(|X - \mathbb{E}[X]| \geq t\right) \leq \frac{\mathbb{V}\mathrm{ar}[X]}{t^2}$.*

We also recall the law of total variance that states the for two random variables X and Y,

$$\mathbb{V}\mathrm{ar}[Y] = \mathbb{E}_x\left(\mathbb{V}\mathrm{ar}[Y \mid X = x]\right) + \mathbb{V}\mathrm{ar}_x\left[\mathbb{E}[Y \mid X = x]\right]. \tag{1}$$

We use the following standard graph theory fact in our proofs (see Appendix A.1 for a proof).

Proposition 2.2 (cf. [15])**.** *For any graph $G(V, E)$, $\sum_{(u,v) \in E} \min(d_u, d_v) \leq 5m\sqrt{m}$.*

Assumption on Size of Subgraph H. Throughout the paper, we assume that the size of the subgraph H is a fixed constant independent of the size of the graph G and hence we suppress the dependency on size of H in various bounds in our analysis using O-notation.

3 A Graph Decomposition Using Fractional Edge-Covers

In this section, we give a simple decomposition of the subgraph H using fractional edge-covers. We start by defining fractional edge-covers formally (see also Figure 1).

Definition 1 (Fractional Edge-Cover Number)**.** *A fractional edge-cover of $H(V_H, E_H)$ is a mapping $\psi : E_H \rightarrow [0,1]$ such that for each vertex $a \in V_H$, $\sum_{e \in E_H, a \in e} \psi(e) \geq 1$. The fractional edge-cover number $\rho(H)$ of H is the minimum value of $\sum_{e \in E_H} \psi(e)$ among all fractional edge-covers ψ.*

The fractional edge-cover number of a graph can be computed by the following LP:

$$\begin{aligned} \rho(H) \quad = \quad & \text{minimize} & & \sum_{e \in E(H)} x_e \\ & \text{subject to} & & \sum_{e \in E_H : a \in e} x_e \geq 1 \text{ for all vertices } a \in V(H). \end{aligned} \tag{2}$$

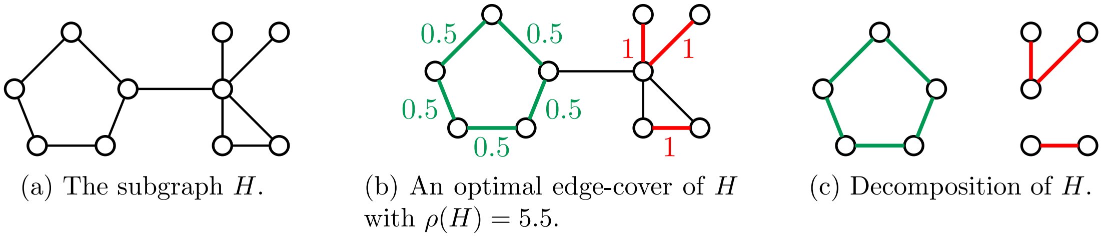
The following lemma is the key to our decomposition. The proof is based on standard ideas in linear programming and is provided in Appendix A.2 for completeness.

Lemma 3.1. *Any subgraph H admits an optimal fractional edge-cover x^* such that the support of x^*, denoted by $\textsc{supp}(x^*)$, is a collection of vertex-disjoint odd cycles and star graphs, and,*

1. *for every odd cycle $C \in \textsc{supp}(x^*)$, $x^*_e = 1/2$ for all $e \in C$;*
2. *for every edge $e \in \textsc{supp}(x^*)$ that does* not *belong to any odd cycle, $x_e = 1$.*

3.1 The Decomposition

We now present the decomposition of H using Lemma 3.1. Let x^* be an optimal fractional edge-cover in Lemma 3.1 and let $\mathcal{C}_1, \ldots, \mathcal{C}_o$ be the odd-cycles in the support of x^* and $\mathcal{S}_1, \ldots, \mathcal{S}_s$ be the stars. We define $\mathcal{D}(H) := \{\mathcal{C}_1, \ldots, \mathcal{C}_o, \mathcal{S}_1, \ldots, \mathcal{S}_s\}$ as the decomposition of H (see Figure 1 below for an illustration).



(a) The subgraph H.

(b) An optimal edge-cover of H with $\rho(H) = 5.5$.

(c) Decomposition of H.

Figure 1: Illustration of the our decomposition for H based on fractional edge-covers.

For every $i \in [o]$, let the length of the odd cycle $\mathcal{C}_i$ be $2k_i + 1$ (i.e., $\mathcal{C}_i = C_{2k_i+1}$); we define $\rho^C_i := k_i + 1/2$. Similarly, for every $j \in [s]$, let the number of petals in $\mathcal{S}_j$ be ℓ_j (i.e., $\mathcal{S}_j = S_{\ell_j}$); we define $\rho^S_j := \ell_j$. By Lemma 3.1,

$$\rho(H) = \sum_{i=1}^{o} \rho^C_i + \sum_{j=1}^{s} \rho^S_j. \tag{3}$$

Recall that by AGM bound, the total number of copies of H possible in G is $m^{\rho(H)}$. We also use the following simple lemma which is a direct corollary of the AGM bound.

Lemma 3.2. *Let $I := \{i_1, \ldots, i_o\}$ and $J := \{j_1, \ldots, j_s\}$ be subsets of $[o]$ and $[s]$, respectively. Suppose $\widetilde{H}$ is the subgraph of H on vertices of the odd cycles $\mathcal{C}_{i_1}, \ldots, \mathcal{C}_{i_o}$ and stars $\mathcal{S}_{j_1}, \ldots, \mathcal{S}_{j_s}$. Then the total number of copies of $\widetilde{H}$ in G is at most $m^{\rho(\widetilde{H})}$ for $\rho(\widetilde{H}) \leq \sum_{i \in I} \rho^C_i + \sum_{j \in J} \rho^S_j$.*

Proof. Let x^* denote the optimal value of LP (2) in the decomposition $\mathcal{D}(H)$. Define y^* as the projection of x^* to edges present in $\widetilde{H}$. It is easy to see that y^* is a feasible solution for LP (2) of $\widetilde{H}$ with value $\sum_{i \in I} \rho_i^C + \sum_{j \in J} \rho_j^S$. The lemma now follows from the AGM bound for $\widetilde{H}$. ∎

3.2 Profiles of Cycles, Stars, and Subgraphs

We conclude this section by specifying the representation of the potential occurrences of the subgraph H in G based on the decomposition $\mathcal{D}(H)$.

Odd cycles: We represent a potential occurrence of an odd cycle C_{2k+1} in G as follows. Let $\boldsymbol{e} = (\vec{e}_1, \ldots, \vec{e}_k) \in \vec{E}^k$ be an ordered tuple of k directed copies of edges in G and suppose $\vec{e}_i := (u_i, v_i)$ for all $i \in [k]$. Define $u^*_{\boldsymbol{e}} = u_1$ and let w be any vertex in $N(u^*_{\boldsymbol{e}})$. We refer to any such collection $(\boldsymbol{e}, w)$ as a *profile* of C_{2k+1} in G. We say that "the profile $(\boldsymbol{e}, w)$ forms a cycle C_{2k+1} in G" iff (i) u_1 is the smallest vertex on the cycle according to $\prec$, (ii) $v_1 \prec w$, *and* (iii) the edges $(u_1, v_1), (v_1, u_2), \ldots, (u_k, v_k), (v_k, w), (w, u_1)$ all exist in G and hence there is a copy of C_{2k+1} on vertices $\{u_1, v_1, u_2, v_2, \ldots, u_k, v_k, w\}$ in G. Note that under this definition and our definition of $\#C_{2k+1}$, each copy of C_{2k+1} correspond to exactly one profile $(\boldsymbol{e}, w)$ and vice versa. As such,

$$\#C_{2k+1} = \sum_{\boldsymbol{e} \in \vec{E}^k} \sum_{w \in N(u^*_{\boldsymbol{e}})} \mathbb{I}\Big((\boldsymbol{e}, w) \text{ forms a cycle } C_{2k+1} \text{ in } G\Big). \tag{4}$$

Stars: We represent a potential occurrence of a star S_ℓ in G by $(v, \boldsymbol{w})$ where v is the center of the star and $\boldsymbol{w} = (w_1, \ldots, w_\ell)$ are the ℓ petals. We refer to $(v, \boldsymbol{w})$ as a *profile* of S_ℓ in G. We say that "the profile $(v, \boldsymbol{w})$ forms a star S_ℓ in G" iff (i) $|\boldsymbol{w}| > 1$, *or* (ii) $(\ell =) |\boldsymbol{w}| = 1$ and $v \prec w_1$; in both cases there is a copy of S_ℓ on vertices $v, w_1, \ldots, w_\ell$. Under this definition, each copy of S_ℓ corresponds to exactly one profile $(v, \boldsymbol{w})$. As such,

$$\#S_\ell = \sum_{v \in V} \sum_{\boldsymbol{w} \in N(v)^\ell} \mathbb{I}\Big((v, \boldsymbol{w}) \text{ forms a star } S_\ell \text{ in } G\Big). \tag{5}$$

Arbitrary subgraphs: We represent a potential occurrence of H in G by an $(o+s)$-tuple $\boldsymbol{R} := ((\boldsymbol{e}_1, w_1), \ldots, (\boldsymbol{e}_o, w_o), (v_1, \boldsymbol{w}_1), \ldots, (v_s, \boldsymbol{w}_s))$ where $(\boldsymbol{e}_i, w_i)$ is a profile of the cycle $\mathcal{C}_i$ in $\mathcal{D}(H)$ and $(v_j, \boldsymbol{w}_j)$ is a profile of the star $\mathcal{S}_j$. We refer to $\boldsymbol{R}$ as a *profile* of H and say that "the profile $\boldsymbol{R}$ forms a copy of H in G" iff (i) each profile forms a corresponding copy of $\mathcal{C}_i$ or $\mathcal{S}_j$ in $\mathcal{D}(H)$, and (ii) the remaining edges of H between vertices specified by $\boldsymbol{R}$ all are present in G (note that by definition of the decomposition $\mathcal{D}(H)$, all vertices of H are specified by $\boldsymbol{R}$). As such,

$$\#H = \sum_{\boldsymbol{R}} \mathbb{I}\Big(\boldsymbol{R} \text{ forms a copy of } H \text{ in } G\Big) \cdot f(H), \tag{6}$$

for a fixed constant $f(H)$ depending only on H as defined below. Let $\pi : V_H \to V_H$ be an automorphism of H. Let $C_1, \ldots, C_o$, $S_1, \ldots, S_s$ denote the cycles and stars in the decomposition of H. We say that π is decomposition preserving if for every $i = 1, \ldots, o$ cycle C_i is mapped to a cycle of the same length and for every $i = 1, \ldots, s$ S_i is mapped to a star with the same number of petals. Let the number of decomposition preserving automorphisms of H be denoted by Z, and define $f(H) = 1/Z$. Define the quantity $\widetilde{\#H} := \sum_{\boldsymbol{R}} \mathbb{I}\Big(\boldsymbol{R} \text{ forms a copy of } H \text{ in } G\Big)$ which is equal to $\#H$ modulo the scaling factor of $f(H)$. It is immediate that estimating $\#H$ and $\widetilde{\#H}$ are equivalent to each other and hence in the rest of the paper, with a slight abuse of notation, we use $\#H$ and $\widetilde{\#H}$ interchangeably.

4 A Sublinear-Time Algorithm for Subgraph Counting

We now present our sublinear time algorithm for approximately counting number of any given arbitrary subgraph H in an underlying graph G and prove Theorem 1. The main component of our algorithm is an unbiased estimator random variable for $\#H$ with low variance. The algorithm in Theorem 1 is then obtained by simply repeating this unbiased estimator in parallel enough number of times (based on the variance) and outputting the average value of these estimators.

4.1 A Low-variance Unbiased Estimator for $\#H$

We present a low-variance unbiased estimator for $\#H$ in this section. Our algorithm is a sampling based algorithm. In the following, we first introduce two separate subroutines for sampling odd cycles (`odd-cycle-sampler`) and stars (`star-sampler`), and then use these components in conjunction with the decomposition we introduced in Section 3, to present our full algorithm. We should right away clarify that `odd-cycle-sampler` and `star-sampler` are not exactly sampling a cycle or a star, but rather sampling a set of vertices and edges (in a non-uniform way) that can potentially form a cycle or star in G, i.e., they sample a profile of these subgraphs defined in Section 3.2.

The `odd-cycle-sampler` Algorithm

We start with the following algorithm for sampling an odd cycle C_{2k+1} for some $k \geq 1$. This algorithm outputs a simple data structure, named the *cycle-sampler tree*, that provides a convenient representation of the samples taken by our algorithm (see Definition 2 immediately after the description of the algorithm). This data structure can be easily avoided when designing a cycle counting algorithm, but will be quite useful for reasoning about the recursive structure of our sampling algorithm for general graphs H.

```
odd-cycle-sampler($G, C_{2k+1}$).

  1. Sample $k$ directed edges $\boldsymbol{e} := (\vec{e}_1, \ldots, \vec{e}_k)$ uniformly at random (with replacement) from $G$
     with the constraint that for $\vec{e}_1 = (u_1, v_1)$, $u_1 \prec v_1$.
  2. Let $u^*_{\boldsymbol{e}} := u_1$ and let $d^*_{\boldsymbol{e}} := d_{u^*_{\boldsymbol{e}}}$.
  3. For $i = 1$ to $t_{\boldsymbol{e}} := \lceil d^*_{\boldsymbol{e}}/\sqrt{m} \rceil$: Sample a vertex $w_i$ uniformly at random from $N(u^*_{\boldsymbol{e}})$.
  4. Let $\boldsymbol{w} := (w_1, \ldots, w_{t_{\boldsymbol{e}}})$. Return the cycle-sampler tree $\mathcal{T}(\boldsymbol{e}, \boldsymbol{w})$ (see Definition 2).
```

Definition 2 (Cycle-Sampler Tree)**.** *The cycle-sampler tree $\mathcal{T}(\boldsymbol{e}, \boldsymbol{w})$ for the tuple $(\boldsymbol{e}, \boldsymbol{w})$ sampled by* `odd-cycle-sampler`(G, C_{2k+1}) *is the following 2-level tree $\mathcal{T}$:*

- *Each node α of the tree contains two attributes:* $\mathsf{label}[\alpha]$ *which consists of some of the edges and vertices in $(\boldsymbol{e}, \boldsymbol{w})$, and an integer* $\mathsf{value}[\alpha]$.
- *For the root α_r of $\mathcal{T}$,* $\mathsf{label}[\alpha_r] := \boldsymbol{e}$ *and* $\mathsf{value}[\alpha_r] := (2m)^k/2$.

  ($\mathsf{value}[\alpha_r]$ *is equal to the inverse of the probability that $\boldsymbol{e}$ is sampled by* `odd-cycle-sampler`).
- *The root α_r has $t_{\boldsymbol{e}}$ child-nodes in $\mathcal{T}$ for a parameter $t_{\boldsymbol{e}} = \lceil d^*_{\boldsymbol{e}}/\sqrt{m} \rceil$ (consistent with line 3 of* `odd-cycle-sampler`(G, C_{2k+1}) *above).*

- *For the i-th child-node α_i of root, $i \in [t_{\boldsymbol{e}}]$,* $\mathsf{label}[\alpha_i] := w_i$ *and* $\mathsf{value}[\alpha_i] := d^*_{\boldsymbol{e}}$

  ($\mathsf{value}[\alpha_i]$ *is equal to the inverse of the probability that w_i is sampled by* `odd-cycle-sampler`*, conditioned on $\boldsymbol{e}$ being sampled).*

Moreover, for each root-to-leaf path $\mathcal{P}_i := (\alpha_r, \alpha_i)$ (for $i \in [t_{\boldsymbol{e}}]$), define $\mathsf{label}[\mathcal{P}_i] := \mathsf{label}[\alpha_r] \cup \mathsf{label}[\alpha_i]$ *and* $\mathsf{value}[\mathcal{P}_i] := \mathsf{value}[\alpha_r] \cdot \mathsf{value}[\alpha_i]$ ($\mathsf{label}[\mathcal{P}_i]$ *is a profile of the cycle C_{2k+1} as defined in Section 3.2).*

See Figure 2a for an illustration of a cycle-sampler tree.

`odd-cycle-sampler` can be implemented in our query model by using k edge-sample queries (and picking the correct direction for e_1 based on $\prec$ and one of the two directions uniformly at random for the other edges) in Line (1), two degree queries in Line (2), and one neighbor query in Line (3). This results in $O(k)$ queries in total for one iteration of the for-loop in Line (3). As such, the total query complexity of `odd-cycle-sampler` is $O(t_{\boldsymbol{e}})$ (recall that k is a constant). It is also straightforward to verify that we can compute the cycle-sampler tree $\mathcal{T}$ of an execution of `odd-cycle-sampler` with no further queries and in $O(t_{\boldsymbol{e}})$ time. We bound the query complexity of this algorithm by bounding the expected number of iterations in the for-loop.

Lemma 4.1. *For the parameter $t_{\boldsymbol{e}}$ in Line (3) of* `odd-cycle-sampler`*, $\mathbb{E}[t_{\boldsymbol{e}}] = O(1)$.*

Proof. By definition, $t_{\boldsymbol{e}} := \lceil d^*_{\boldsymbol{e}}/\sqrt{m} \rceil$ for $d^*_{\boldsymbol{e}} = \min(d_u, d_v)$ for an edge $e_1 = (u, v)$ chosen uniformly at random from G. As such, by Proposition 2.2,

$$\mathbb{E}[t_{\boldsymbol{e}}] = 1 + O(m^{-1/2}) \cdot \frac{1}{m} \cdot \sum_{(u,v)\in E} \min(d_u, d_v) \underset{\text{Proposition 2.2}}{=} O(m^{-3/2}) \cdot 5m\sqrt{m} = O(1). \qquad \blacksquare$$

We now define a process for estimating the number of odd cycles in a graph using the information stored in the cycle-sampler tree and the `odd-cycle-sampler` algorithm. While we do not use this process in a black-box way in our main algorithm, abstracting it out makes the analysis of our main algorithm simpler to follow and more transparent, and serves as a warm-up for our main algorithm.

Warm-up: An Estimator for Odd Cycles. Let $\mathcal{T} :=$ `odd-cycle-sampler`(G, C_{2k+1}) be the output of an invocation of `odd-cycle-sampler`. Note that the cycle-sampler tree $\mathcal{T}$ is a random variable depending on the randomness of `odd-cycle-sampler`. We define the random variable X_i such that $X_i := \mathsf{label}[\mathcal{P}_i]$ for the i-th root-to-leaf path iff $\mathsf{label}[\mathcal{P}_i]$ forms a copy of C_{2k+1} in G and otherwise $X_i := 0$ (according to the definition of Section 3). We further define $Y := \frac{1}{t_{\boldsymbol{e}}} \cdot \sum_{i=1}^{t_{\boldsymbol{e}}} X_i$ (note that $t_{\boldsymbol{e}}$ is also a random variable). Our estimator algorithm can compute the value of these random variables using the information stored in the tree $\mathcal{T}$ plus additional $O(k) = O(1)$ queries for each of the $t_{\boldsymbol{e}}$ root-to-leaf path $\mathcal{P}_i$ to detect whether $(\boldsymbol{e}, w_i)$ forms a copy of H or not. Thus, the query complexity and runtime of the estimator is still $O(t_{\boldsymbol{e}})$ (which in expectation is $O(1)$ by Lemma 4.1). We now analyze its expectation and variance.

Lemma 4.2. *For the random variable Y associated with* `odd-cycle-sampler`(G, C_{2k+1})*,*

$$\mathbb{E}[Y] = (\#C_{2k+1}), \qquad \mathbb{V}\text{ar}[Y] \leq (2m)^k\sqrt{m} \cdot \mathbb{E}[Y].$$

Proof. We first analyze the expected value of X_i's and then use this to bound $\mathbb{E}[Y]$. For any $i \in [t_{\boldsymbol{e}}]$, we have, $X_i = \mathsf{value}[\alpha_r] \cdot \mathsf{value}[\alpha_i] \cdot \mathbb{I}(\mathsf{label}[\alpha_r] \cup \mathsf{label}[\alpha_i]$ forms a copy of $C_{2k+1})$, where, as per Definition 2, α_r is the root of $\mathcal{T}$. As such,

$$\begin{aligned}
\mathbb{E}\left[X_i\right] &= \sum_{\boldsymbol{e}\in\vec{E}^k}\sum_{w\in N(u^*_{\boldsymbol{e}})} \Pr\left(\mathsf{label}[\alpha_r]=\boldsymbol{e}\right)\cdot\Pr\left(\mathsf{label}[\alpha_i]=w\right) \\
&\qquad\qquad\cdot\,\mathbb{I}\left((\boldsymbol{e},w)\text{ forms a copy of }C_{2k+1}\right)\cdot\mathsf{value}[\alpha_r]\cdot\mathsf{value}[\alpha_i] \\
&= \frac{2}{(2m)^k}\cdot\sum_{\boldsymbol{e}}\left(\frac{1}{d^*_{\boldsymbol{e}}}\cdot\sum_{w}\mathbb{I}\left((\boldsymbol{e},w)\text{ forms a copy of }C_{2k+1}\right)\cdot\frac{(2m)^k}{2}\cdot d^*_{\boldsymbol{e}}\right) \\
&= \sum_{\boldsymbol{e}}\sum_{w}\mathbb{I}\left((\boldsymbol{e},w)\text{ forms a copy of }C_{2k+1}\right) \underset{\text{Eq }(4)}{=} (\#C_{2k+1}).
\end{aligned}$$

As $\mathbb{E}\left[Y\right]=\mathbb{E}\left[X_i\right]$ for any $i\in[t]$ by linearity of expectation, we obtain the desired bound on $\mathbb{E}\left[Y\right]$.

We now bound $\mathbb{V}\text{ar}\left[Y\right]$ using the fact that it is obtained by taking average of $t_{\boldsymbol{e}}$ random variables that are independent *after* we condition on the choice of $\boldsymbol{e}$. We formalize this as follows. Note that for any two $i\neq j$, the random variables $X_i\mid\boldsymbol{e}$ and $X_j\mid\boldsymbol{e}$ are independent of each other (even though X_i and X_j in general are correlated). By the law of total variance in Eq (1),

$$\mathbb{V}\text{ar}\left[Y\right]=\mathbb{E}\left[\mathbb{V}\text{ar}\left[Y\mid\boldsymbol{e}\right]\right]+\mathbb{V}\text{ar}\left[\mathbb{E}\left[Y\mid\boldsymbol{e}\right]\right]. \tag{7}$$

We bound each term separately now. Recall that $Y:=\frac{1}{t_{\boldsymbol{e}}}\sum_{i=1}^{t_{\boldsymbol{e}}}X_i$ and X_i's are independent conditioned on $\boldsymbol{e}$. As such,

$$\begin{aligned}
\mathbb{E}\left[\mathbb{V}\text{ar}\left[Y\mid\boldsymbol{e}\right]\right] &= \frac{2}{(2m)^k}\sum_{\boldsymbol{e}\in\vec{E}^k}\mathbb{V}\text{ar}\left[Y\mid\boldsymbol{e}\right]=\frac{2}{(2m)^k}\sum_{\boldsymbol{e}}\frac{1}{t^2_{\boldsymbol{e}}}\cdot\sum_{i=1}^{t_{\boldsymbol{e}}}\mathbb{V}\text{ar}\left[X_i\mid\boldsymbol{e}\right] \\
&\qquad\qquad\text{(by conditional independence of } X_i\text{'s)} \\
&\leq \frac{2}{(2m)^k}\sum_{\boldsymbol{e}}\frac{1}{t_{\boldsymbol{e}}}\,\mathbb{E}\left[X_1^2\mid\boldsymbol{e}\right]. \qquad\text{(as distribution of all } X_i\text{'s are the same)}
\end{aligned}$$

Hence, it suffices to calculate $\mathbb{E}\left[X_1^2\mid\boldsymbol{e}\right]$. We have,

$$\begin{aligned}
\mathbb{E}\left[X_1^2\mid\boldsymbol{e}\right] &= \sum_{w\in N(u^*)}\Pr\left(\mathsf{label}[\alpha_1]=w\right)\cdot\mathbb{I}\left((\boldsymbol{e},w)\text{ forms a copy of }C_{2k+1}\right)\cdot\left(\mathsf{value}[\alpha_r]\cdot\mathsf{value}[\alpha_1]\right)^2 \\
&= \frac{1}{d^*_{\boldsymbol{e}}}\cdot\sum_{w}\mathbb{I}\left((\boldsymbol{e},w)\text{ forms a copy of }C_{2k+1}\right)\cdot\left(\frac{(2m)^k}{2}\cdot d^*_{\boldsymbol{e}}\right)^2 \\
&\leq \frac{(2m)^{2k}}{4}\cdot d^*_{\boldsymbol{e}}\cdot\sum_{w}\mathbb{I}\left((\boldsymbol{e},w)\text{ forms a copy of }C_{2k+1}\right) \\
&\underset{\text{Eq }(4)}{=} \frac{(2m)^{2k}}{4}\cdot d^*_{\boldsymbol{e}}\cdot(\#C_{2k+1}\mid\boldsymbol{e}),
\end{aligned}$$

where $(\#C_{2k+1}\mid\boldsymbol{e})$ denotes the number of copies of C_{2k+1} containing the sub-profile $\boldsymbol{e}=(\vec{e}_1,\ldots,\vec{e}_k)$. By plugging in this bound in the above equation, we have,

$$\begin{aligned}
\mathbb{E}\left[\mathbb{V}\text{ar}\left[Y\mid\boldsymbol{e}\right]\right] &\leq \frac{2}{(2m)^k}\sum_{\boldsymbol{e}\in\vec{E}^k}\frac{1}{t_{\boldsymbol{e}}}\cdot\frac{(2m)^{2k}}{4}\cdot d^*_{\boldsymbol{e}}\cdot(\#C_{2k+1}\mid\boldsymbol{e}) \\
&\leq \frac{(2m)^k\sqrt{m}}{2}\sum_{\boldsymbol{e}}(\#C_{2k+1}\mid\boldsymbol{e})=\frac{(2m)^k\sqrt{m}}{2}\cdot(\#C_{2k+1}),
\end{aligned} \tag{8}$$

by the choice of $t_{\boldsymbol{e}} \geq d^*_{\boldsymbol{e}}/\sqrt{m}$. We now bound the second term in Eq (7). Note that by the by proof of $\mathbb{E}\left[Y\right] = (\#C_{2k+1})$ above, we also have $\mathbb{E}\left[Y \mid \boldsymbol{e}\right] = (\#C_{2k+1} \mid \boldsymbol{e})$. As such,

$$\begin{aligned}\mathbb{V}\text{ar}\left[\mathbb{E}\left[Y \mid \boldsymbol{e}\right]\right] = \mathbb{V}\text{ar}\left[(\#C_{2k+1} \mid \boldsymbol{e})\right] \leq \mathbb{E}\left[(\#C_{2k+1} \mid \boldsymbol{e})^2\right] = \frac{2}{(2m)^k} \sum_{\boldsymbol{e}} (\#C_{2k+1} \mid \boldsymbol{e})^2 \\ \leq \frac{2}{(2m)^k} \cdot \left(\sum_{\boldsymbol{e}} (\#C_{2k+1} \mid \boldsymbol{e})\right)^2 = \frac{2}{(2m)^k} \left(\#C_{2k+1}\right)^2 \leq \sqrt{m} \cdot \left(\#C_{2k+1}\right),\end{aligned}$$

where the last inequality is by AGM bound in Lemma 3.2 which states that $(\#C_{2k+1}) \leq m^k\sqrt{m}$ (as $\rho(C_{2k+1}) = k + 1/2$). Plugging in this bound in the second term of Eq (7) and using Eq (8) for the first term yields:

$$\begin{aligned}\mathbb{V}\text{ar}\left[Y\right] &= \mathbb{E}\left[\mathbb{V}\text{ar}\left[Y \mid \boldsymbol{e}\right]\right] + \mathbb{V}\text{ar}\left[\mathbb{E}\left[Y \mid \boldsymbol{e}\right]\right] \\ &\leq \frac{(2m)^k\sqrt{m}}{2} \cdot \left(\#C_{2k+1}\right) + \sqrt{m} \cdot \left(\#C_{2k+1}\right) \leq (2m)^k\sqrt{m} \cdot \left(\#C_{2k+1}\right) = (2m)^k\sqrt{m} \cdot \mathbb{E}\left[Y\right],\end{aligned}$$

where the equality is by the bound on $\mathbb{E}\left[Y\right]$ proven in the first part. ∎

The `star-sampler` Algorithm

We now give an algorithm for sampling a star S_ℓ with ℓ petals. Similar to `odd-cycle-sampler`, this algorithm also outputs a simple data structure, named the *star-sampler tree*, that provides a convenient representation of the samples taken by our algorithm (see Definition 3, immediately after the description of the algorithm). This data structure can be easily avoided when designing a star counting algorithm, but will be quite useful for reasoning about the recursive structure of our sampling algorithm for general graphs H.

`star-sampler`(G, S_ℓ).

1. Sample a vertex $v \in V$ chosen with probability proportional to its degree in G (i.e., for any vertex $u \in V$, $\Pr\left(u \text{ is chosen as the vertex } v\right) = d_u/2m$).
2. Sample ℓ vertices $\boldsymbol{w} := (w_1, \ldots, w_\ell)$ from $N(v)$ uniformly at random (without replacement).
3. Return the star-sampler tree $\mathcal{T}(v, \boldsymbol{w})$ (see Definition 3).

Definition 3 (Star-Sampler Tree)**.** *The star-sampler tree $\mathcal{T}(v, \boldsymbol{w})$ for the tuple $(v, \boldsymbol{w})$ sampled by* `star-sampler`(G, S_ℓ) *is the following 2-level tree $\mathcal{T}$ (with the same attributes as in Definition 2) with only two nodes:*

- *For the root α_r of $\mathcal{T}$,* $\mathsf{label}[\alpha_r] := v$ *and* $\mathsf{value}[\alpha_r] := 2m/d_v$*.*

  ($\mathsf{value}[\alpha_r]$ *is equal to the inverse of the probability that v is sampled by* `star-sampler`*).*

- *The root α_r has exactly one child-node α_l in $\mathcal{T}$ with* $\mathsf{label}[\alpha_l] = \boldsymbol{w} = (w_1, \ldots, w_\ell)$ *and* $\mathsf{value}[\alpha_l] = \binom{d_v}{\ell}$*.*

  ($\mathsf{value}[\alpha_l]$ *is equal to the inverse of the probability that $\boldsymbol{w}$ is sampled by* `star-sampler`*, conditioned on v being sampled).*

Moreover, for the root-to-leaf path $\mathcal{P} := (\alpha_r, \alpha_l)$, we define $\mathsf{label}[\mathcal{P}] := \mathsf{label}[\alpha_r] \cup \mathsf{label}[\alpha_l]$ *and* $\mathsf{value}[\mathcal{P}] := \mathsf{value}[\alpha_r] \cdot \mathsf{value}[\alpha_l]$. ($\mathsf{label}[\mathcal{P}]$ *is a representation of the star S_ℓ as defined in Section 3.2*).

See Figures 2b and 2c for an illustration of star-sampler trees.

`star-sampler` can be implemented in our query model by using one edge-sample query in Line (1) and then picking one of the endpoints uniformly at random, a degree query to determine the degree of v, and ℓ neighbor queries in Line (2), resulting in $O(\ell)$ queries in total. It is also straightforward to verify that we can compute the star-sampler tree $\mathcal{T}$ of an execution of `star-sampler` with no further queries and in $O(1)$ time.

We again define a process for estimating the number of stars in a graph using the information stored in the star-sampler tree and the `star-sampler` algorithm, as a warm-up to our main result in the next section.

Warm-up: An Estimator for Stars. The star-sampler tree $\mathcal{T}$ is a random variable depending on the randomness of `star-sampler`. We define the random variable X such that $X := \mathsf{value}[\mathcal{P}]$ for the root-to-leaf path of $\mathcal{T}$ iff $\mathsf{label}[\mathcal{P}]$ forms a copy of S_ℓ in G and otherwise $X := 0$. Our estimator algorithm can compute the value of this random variable using only the information stored in the tree $\mathcal{T}$ with no further queries to the graph (by simply checking if all w_i's in $\boldsymbol{w}$ are distinct). As such, the query complexity and runtime of the estimator algorithm is still $O(1)$. We now prove,

Lemma 4.3. *For the random variable X associated with* `star-sampler`(G, S_ℓ),

$$\mathbb{E}[X] = (\#S_\ell), \qquad \mathbb{V}\mathrm{ar}[X] \leq 2m^\ell \cdot \mathbb{E}[X].$$

Proof. Firstly, we have $X = \mathsf{value}[\alpha_r] \cdot \mathsf{value}[\alpha_l] \cdot \mathbb{I}(\mathsf{label}[\alpha_r] \cup \mathsf{label}[\alpha_l] \text{ forms a copy of } S_\ell)$. As such,

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{v \in V} \sum_{\boldsymbol{w} \in N(v)^\ell} \Pr(\mathsf{label}[\alpha_r] = v) \cdot \Pr(\mathsf{label}[\alpha_l] = \boldsymbol{w}) \\
&\qquad\qquad \cdot \mathbb{I}((v, \boldsymbol{w}) \text{ forms a copy of } S_\ell) \cdot \mathsf{value}[\alpha_r] \cdot \mathsf{value}[\alpha_l] \\
&= \sum_v \frac{d_v}{2m} \cdot \sum_{\boldsymbol{w}} \frac{1}{\binom{d_v}{\ell}} \cdot \mathbb{I}((v, \boldsymbol{w}) \text{ forms a copy of } S_\ell) \cdot (2m/d_v) \cdot \binom{d_v}{\ell} \\
&= \sum_v \sum_{\boldsymbol{w}} \mathbb{I}((v, \boldsymbol{w}) \text{ forms a copy of } S_\ell) = (\#S_\ell).
\end{aligned}$$

This proves the desired bound on the exception. We now bound $\mathbb{V}\mathrm{ar}[X]$.

$$\begin{aligned}
\mathbb{V}\mathrm{ar}[X] \leq \mathbb{E}\left[X^2\right] &= \sum_{v \in V} \sum_{\boldsymbol{w} \in N(v)^\ell} \Pr(\mathsf{label}[\alpha_r] = v) \cdot \Pr(\mathsf{label}[\alpha_l] = \boldsymbol{w}) \\
&\qquad\qquad \cdot \mathbb{I}((v, \boldsymbol{w}) \text{ forms a copy of } S_\ell) \cdot (\mathsf{value}[\alpha_r] \cdot \mathsf{value}[\alpha_l])^2 \\
&= \sum_v \frac{d_v}{2m} \cdot \sum_{\boldsymbol{w}} \frac{1}{\binom{d_v}{\ell}} \cdot \mathbb{I}((v, \boldsymbol{w}) \text{ forms a copy of } S_\ell) \cdot \left((2m/d_v) \cdot \binom{d_v}{\ell}\right)^2 \\
&= \sum_v \sum_{\boldsymbol{w}} \mathbb{I}((v, \boldsymbol{w}) \text{ forms a copy of } S_\ell) \cdot (2m/d_v) \cdot \binom{d_v}{\ell} \\
&\leq \sum_v \sum_{\boldsymbol{w}} \mathbb{I}((v, \boldsymbol{w}) \text{ forms a copy of } S_\ell) \cdot 2m \cdot {d_v}^{\ell-1} \qquad \left(\text{since } \tbinom{d_v}{\ell} \leq d_v^\ell\right)
\end{aligned}$$

$$\begin{aligned}&\leq 2m^{\ell} \cdot \sum_{v} \sum_{\boldsymbol{w}} \mathbb{I}((v, \boldsymbol{w}) \text{ forms a copy of } S_{\ell}) && \text{(since } d_v \leq m\text{)}\\ &= 2m^{\ell} \cdot (\# S_{\ell}) = 2m^{\ell} \cdot \mathbb{E}\left[X\right],\end{aligned}$$

by the bound on $\mathbb{E}\left[X\right]$ in the first part. ■

Remark 4.4. *As we pointed out earlier, the bounds achieved by our Theorem 1 for counting stars are suboptimal in the light of the results in [2]. In Appendix A.3, we show that in fact our estimator in this section—using* a different analysis *which is similar to that of [2]—also matches the optimal bounds achieved by [2]. This suggests that even for the case of stars our algorithm in Theorem 1 is still optimal even though the general bounds in the theorem statement are not. We note that our main estimator algorithm relies on the particular analysis of the estimator for stars presented in this section as the alternate analysis does not seem to compose with the rest of the argument.*

The Estimator Algorithm for Arbitrary Subgraphs

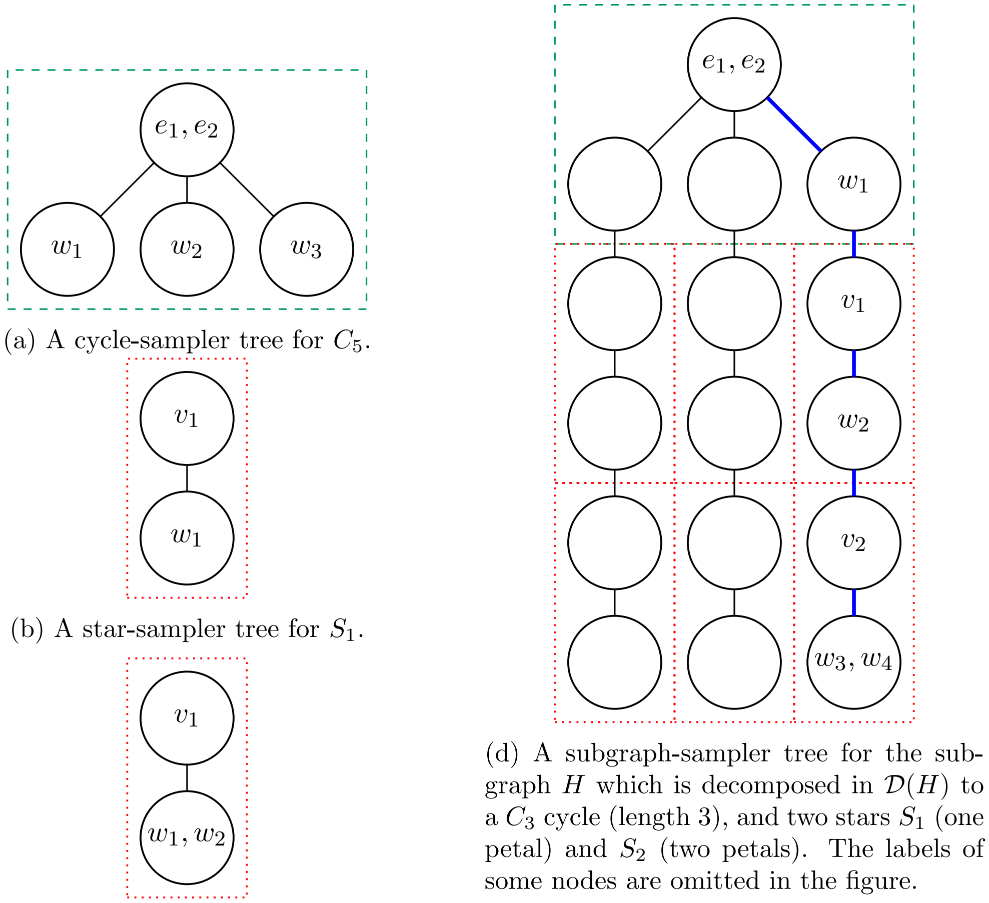
We now present our main estimator for the number of occurrences of an arbitrary subgraph H in G, denoted by $(\#H)$. Recall the decomposition $\mathcal{D}(H) := \{\mathcal{C}_1, \ldots, \mathcal{C}_o, \mathcal{S}_1, \ldots, \mathcal{S}_s\}$ of H introduced in Section 3. Our algorithm creates a *subgraph-sampler tree* $\mathcal{T}$ (a generalization of cycle-sampler and star-sampler trees in Definitions 2 and 3) and use it to estimate $(\#H)$. We define the subgraph-sampler tree $\mathcal{T}$ and the algorithm `subgraph-sampler`(G, H) that creates it simultaneously:

Subgraph-Sampler Tree. The subgraph-sampler tree $\mathcal{T}$ is a z-level tree for $z := (2o + 2s)$ returned by `subgraph-sampler`(G, H). The algorithm `subgraph-sampler` constructs $\mathcal{T}$ as follows (see Figure 2 for an illustration).

Sampling Odd Cycles. In `subgraph-sampler`(G, H), we first run `odd-cycle-sampler`$(G, \mathcal{C}_1)$ and initiate $\mathcal{T}$ to be its output cycle-sampler tree. For every (current) leaf-node α of $\mathcal{T}$, we run `odd-cycle-sampler`$(G, \mathcal{C}_2)$ *independently* to obtain a cycle-sampler tree $\mathcal{T}_{\alpha}$ (we say that α *started* the sampling of $\mathcal{T}_{\alpha}$). We then extend the tree $\mathcal{T}$ with two new layers by connecting each leaf-node α to the root of $\mathcal{T}_{\alpha}$ that started its sampling. This creates a 4-level tree $\mathcal{T}$. We continue like this for o steps, each time appending the tree obtained by `odd-cycle-sampler`$(G, \mathcal{C}_j)$ for $j \in [o]$, to the (previous) leaf-node that started this sampling. This results in a $(2o)$-level tree. Note that the nodes in the tree $\mathcal{T}$ can have different degrees as the number of leaf-nodes in the cycle-sampler tree is not necessarily the same always (not even for two different trees associated with one single $\mathcal{C}_j$ through different calls to `odd-cycle-sampler`$(G, \mathcal{C}_j)$).

Sampling Stars. Once we iterated over all odd cycles of $\mathcal{D}(H)$, we switch to processing stars $\mathcal{S}_1, \ldots, \mathcal{S}_s$. The approach is identical to the previous part. Let α be a (current) leaf-node of $\mathcal{T}$. We run `star-sampler`$(G, \mathcal{S}_1)$ to obtain a star-sampler tree $\mathcal{T}_{\alpha}$ and connect α to $\mathcal{T}_{\alpha}$ to extend the levels of tree by 2 more. We continue like this for s steps, each time appending the tree obtained by `star-sampler`$(G, \mathcal{S}_j)$ for $j \in [s]$, to the (former) leaf-node that started this sampling. This results in a z-level tree $\mathcal{T}$. Note that all nodes added when sampling stars have exactly one child-node (except for the leaf-nodes) as by Definition 3, star-sampler trees always contain only two nodes.

Labels and Values. Each node α of $\mathcal{T}$ is again given two attributes, $\mathsf{label}[\alpha]$ and $\mathsf{value}[\alpha]$, which are defined to be exactly the same attributes in the corresponding cycle-sampler or star-sampler tree that was used to define these nodes (recall that each node of $\mathcal{T}$ is "copied" from a node in either a cycle-sampler or a star-sampler tree). Finally, for each root-to-leaf path $\mathcal{P}$ in $\mathcal{T}$, we define $\mathsf{label}[\mathcal{P}] := \bigcup_{\alpha \in \mathcal{P}} \mathsf{label}[\alpha]$ and $\mathsf{value}[\mathcal{P}] := \prod_{\alpha \in \mathcal{P}} \mathsf{value}[\alpha]$. In particular, $\mathsf{label}[\mathcal{P}] := ((\boldsymbol{e}_1, w_1), \ldots, (\boldsymbol{e}_o, w_o), (v_1, \boldsymbol{w}_1), \ldots, (v_s, \boldsymbol{w}_s))$ by definition of labels of cycle-sampler and star-sampler



(a) A cycle-sampler tree for C_5.

(b) A star-sampler tree for S_1.

(c) A star-sampler tree for S_2.

(d) A subgraph-sampler tree for the subgraph H which is decomposed in $\mathcal{D}(H)$ to a C_3 cycle (length 3), and two stars S_1 (one petal) and S_2 (two petals). The labels of some nodes are omitted in the figure.

Figure 2: Illustration of the sampler-subgraph $\mathcal{T}$ for the subgraph H of Figure 1. The (blue) thick line in part (d) shows a root-to-leaf path $\mathcal{P}$ with $\mathsf{value}[\mathcal{P}] = (m^2) \cdot (d^*_{(e_1,e_2)}) \cdot (\frac{2m}{d_{v_1}}) \cdot \binom{d_{v_1}}{1} \cdot (\frac{2m}{d_{v_2}}) \cdot \binom{d_{v_2}}{2}$. The variable X of $\mathcal{P}$ is equal to $\mathsf{value}[\mathcal{P}]$ iff the profile $((e_1, e_2, w_1), (v_1, w_2), (v_2, w_2, w_3))$ forms a copy of H in G.

trees. As such $\mathsf{label}[\mathcal{P}]$ is a representation of the subgraph H as defined in Section 3.2. By making $O(1)$ additional pair-queries to query all the remaining edges of this representation of H we can determine whether $\mathsf{label}[\mathcal{P}]$ forms a copy of H or not.

This concludes the description of `subgraph-sampler`(G, H) and its output subgraph-sampler tree $\mathcal{T}$. We start analyzing this algorithm by bounding its query complexity.

Lemma 4.5. *The expected query complexity and running time of* `subgraph-sampler` *is* $O(1)$*.*

Proof. As was the case for `odd-cycle-sampler` and `star-sampler`, in the `subgraph-sampler` also the query complexity of the algorithm is within a constant factor of number of nodes in the subgraph-sampler tree $\mathcal{T}$ that it returns. Hence, we only need to bound the number of nodes in $\mathcal{T}$.

Let $\mathcal{L}_1, \ldots, \mathcal{L}_z$ denote the set of nodes in layers 1 to z of $\mathcal{T}$. $\mathcal{L}_1$ contains only the root α_r of $\mathcal{T}$. Let $\boldsymbol{e}_r := \mathsf{label}[\alpha_r]$. By definition of the cycle-sampler tree that forms the first two layers of $\mathcal{T}$, the number of child-nodes of α_r is $t_{\boldsymbol{e}_r}$ (defined in Line (3) of `odd-cycle-sampler`). As such $|\mathcal{L}_2| = t_{\boldsymbol{e}_1}$. The nodes in any even layer of $\mathcal{T}$ have only a single child-node by construction, hence $|\mathcal{L}_3| = |\mathcal{L}_2|$.

Now, for each node α in $\mathcal{L}_3$ with label $\boldsymbol{e}_\alpha := \mathsf{label}[\alpha]$, the number of child-nodes is $t_{\boldsymbol{e}_\alpha}$ (the number of child-nodes of a node in an even layer is always one by construction). Hence, $|\mathcal{L}_4| = \sum_{\alpha \in \mathcal{L}_3} t_{\boldsymbol{e}_\alpha}$. By continuing like this, we obtain that for each layer $\mathcal{L}_{2i}$ for $i \leq o$, $|\mathcal{L}_{2i}| = \sum_{\alpha \in \mathcal{L}_{2i-1}} t_{\boldsymbol{e}_\alpha}$. After this, we reach the layers corresponding to star-sampler subgraphs, in which every non-leaf node has exactly one child-node, and hence $|\mathcal{L}_j| = |\mathcal{L}_{2o}|$ for all $j \geq 2o$. Moreover note that each odd-layer node of $\mathcal{T}$ *independently* starts the sampling of its subtree (even independently of its parent-nodes).

Define $t_1, \ldots, t_o$ as the random variables denoting the number of leaf-nodes in the cycle-sampler trees for $\mathcal{C}_1, \ldots \mathcal{C}_o$ in the decomposition $\mathcal{D}(H)$. By Lemma 4.1, $\mathbb{E}[t_i] = O(1)$ for all $i \in [o]$. By the above discussion, we have,

$$\mathbb{E}[|\mathcal{L}_1 \cup \ldots \cup \mathcal{L}_z|] = O(z) \cdot \mathbb{E}[t_1] \cdot \mathbb{E}[t_2] \cdot \ldots \cdot \mathbb{E}[t_o] + O(z) \underset{\text{Lemma 4.1}}{=} O(1)^{O(z)} = O(1),$$

as $z = O(1)$ since size of H is constant. Finally, note that running time of the algorithm is at most a constant factor larger than its query complexity. ∎

We are now ready to present our estimator algorithm using `subgraph-sampler` and the subgraph-sampler tree $\mathcal{T}$ it outputs.

An Estimator for Arbitrary Subgraphs. Note that as before the subgraph-sampler tree $\mathcal{T}$ itself is a random variable depending on the randomness of `subgraph-sampler`. For any root-to-leaf path $\mathcal{P}_i := \alpha_1, \ldots, \alpha_z$ of $\mathcal{T}$, we define the random variable X_i such that $X_i := \mathsf{value}[\mathcal{P}_i]$ iff $\mathsf{label}[\mathcal{P}_i]$ forms a copy of H in G and otherwise $X_i := 0$. We further define $Y := (\frac{1}{t}\sum_{i=1}^{t} X_i)$, where t is the number of leaf-nodes of $\mathcal{T}$ (which itself is a random variable). These random variables can all be computed from $\mathcal{T}$ and `subgraph-sampler` with at most $O(1)$ further pair-queries per each root-to-leaf path $\mathcal{P}$ of the tree to determine if indeed $\mathsf{label}[\mathcal{P}]$ forms a copy of H in G or not. As such, query complexity and runtime of this algorithm is proportional to `subgraph-sampler` (which in expectation is $O(1)$ by Lemma 4.5). In the following two lemmas, we show that Y is a low-variance unbiased estimator of $(\#H)$. To continue, we first need some notation.

Notation. For any node α in $\mathcal{T}$, we use $\mathcal{T}_\alpha$ to denote the sub-tree of $\mathcal{T}$ rooted at α. For a leaf-node α, we define a random variable Y_α which is $\mathsf{value}[\alpha]$ iff for the root-to-leaf path $\mathcal{P}$ ending in α, $\mathsf{label}[\mathcal{P}]$ forms a copy of H in G and otherwise Y_α is 0. For an internal node α in $\mathcal{T}$ with t child-nodes $\alpha_1, \ldots, \alpha_t$, we define $Y_\alpha = \mathsf{value}[\alpha] \cdot \left(\frac{1}{t} \cdot \sum_{i=1}^{t} Y_i\right)$. It is easy to verify that Y_{α_r} for the root α_r of $\mathcal{T}$ is the same as the estimator random variable Y defined earlier. Furthermore, for a node α in level ℓ of $\mathcal{T}$, we define $\boldsymbol{L}_\alpha := (\mathsf{label}[\alpha_1], \mathsf{label}[\alpha_2], \ldots, \mathsf{label}[\alpha_{\ell-1}])$, where $\alpha_1, \ldots, \alpha_{\ell-1}$ forms the path from the root of $\mathcal{T}$ to the parent of α.

We analyze the expected value and the variance of the estimator in the following two lemmas.

Lemma 4.6. *For the random variable Y for* `subgraph-sampler`(G, H)*, $\mathbb{E}[Y] = (\#H)$.*

Proof. We prove this lemma inductively by showing that for any node α in an *odd layer* of $\mathcal{T}$,

$$\mathbb{E}[Y_\alpha \mid \boldsymbol{L}_\alpha] = (\#H \mid \boldsymbol{L}_\alpha),$$

where $(\#H \mid \boldsymbol{L}_\alpha)$ denotes the number of copies of H in G that contain the vertices and edges specified by $\boldsymbol{L}_\alpha$ (according to the decomposition $\mathcal{D}(H)$). $\mathbb{E}[Y_\alpha \mid \boldsymbol{L}_\alpha]$ measures the value of Y_α after we fix the rest of the tree $\mathcal{T}$ and let the sub-tree $\mathcal{T}_\alpha$ be chosen randomly as in `subgraph-sampler`.

The base case of the induction, i.e., for vertices in the last odd layer of $\mathcal{T}$ follows exactly as in the proofs of Lemmas 4.2 and 4.3 (as will also become evident shortly) and hence we do not repeat it here. We now prove the induction hypothesis. Fix a vertex α in an odd layer ℓ. We consider two cases based on whether $\ell < 2o$ (hence α is root of a cycle-sampler tree) or $\ell > 2o$ (hence α is root of a star-sampler tree).

Case of $\ell < 2o$. In this case, the sub-tree $\mathcal{T}_\alpha$ in the next two levels is a cycle-sampler tree, hence,

$$\mathbb{E}\left[Y_\alpha \mid \boldsymbol{L}_\alpha\right] = \sum_{\boldsymbol{e}} \Pr\left(\mathsf{label}[\alpha] = \boldsymbol{e}\right) \cdot \mathsf{value}[\alpha] \cdot \left(\frac{1}{t_{\boldsymbol{e}}} \sum_{i=1}^{t_e} \mathbb{E}\left[Y_{\alpha_i} \mid \boldsymbol{L}_\alpha, \boldsymbol{e}\right]\right) \qquad \text{(here, } \alpha_i\text{'s are child-nodes of } \alpha)$$

$$= \sum_{\boldsymbol{e}} \frac{1}{t_{\boldsymbol{e}}} \sum_{i=1}^{t_e} \mathbb{E}\left[Y_{\alpha_i} \mid \boldsymbol{L}_\alpha, \boldsymbol{e}\right] \qquad \text{(as by definition, } \mathsf{value}[\alpha] = \Pr\left(\mathsf{label}[\alpha] = \boldsymbol{e}\right)^{-1})$$

Note that each α_i has exactly one child-node, denoted by β_i. As such,

$$\begin{aligned}
\mathbb{E}\left[Y_\alpha \mid \boldsymbol{L}_\alpha\right] &= \sum_{\boldsymbol{e}} \frac{1}{t_{\boldsymbol{e}}} \sum_{i=1}^{t_e} \mathbb{E}\left[Y_{\alpha_i} \mid \boldsymbol{L}_\alpha, \boldsymbol{e}\right] \\
&= \sum_{\boldsymbol{e}} \frac{1}{t_{\boldsymbol{e}}} \sum_{i=1}^{t_e} \sum_{w} \Pr\left(\mathsf{label}[\alpha_i] = w\right) \cdot \mathsf{value}[\alpha_i] \cdot \mathbb{E}\left[Y_{\beta_i} \mid \boldsymbol{L}_\alpha, \boldsymbol{e}, w\right] \\
&= \sum_{\boldsymbol{e}} \frac{1}{t_{\boldsymbol{e}}} \sum_{i=1}^{t_e} \sum_{w} \mathbb{E}\left[Y_{\beta_i} \mid \boldsymbol{L}_{\beta_i}\right] \\
&\qquad \text{(by definition } \mathsf{value}[\alpha_i] = \Pr\left(\mathsf{label}[\alpha_i] = w\right)^{-1} \text{ and } \boldsymbol{L}_{\beta_i} = \boldsymbol{L}_\alpha, (\boldsymbol{e}, w)) \\
&= \sum_{\boldsymbol{e}} \frac{1}{t_{\boldsymbol{e}}} \sum_{i=1}^{t_e} \sum_{w} (\#H \mid \boldsymbol{L}_{\beta_i}) = \sum_{\boldsymbol{e}} \frac{1}{t_{\boldsymbol{e}}} \sum_{i=1}^{t_e} \sum_{w} (\#H \mid \boldsymbol{L}_\alpha, (\boldsymbol{e}, w)) \\
&\qquad \text{(by induction hypothesis for odd-layer nodes } \beta_i\text{'s)} \\
&= \sum_{\boldsymbol{e}} \sum_{w} (\#H \mid \boldsymbol{L}_\alpha, (\boldsymbol{e}, w)) = (\#H \mid \boldsymbol{L}_\alpha).
\end{aligned}$$

This concludes the proof of induction hypothesis in this case. Note that this proof was basically the same proof for the expectation bound of the estimator for cycle-sampler tree in Lemma 4.2.

Case of $\ell > 2o$. In this case, the sub-tree $\mathcal{T}_\alpha$ in the next two levels is a star-sampler tree. By the same analogy made in the proof of the previous part and Lemma 4.2, the proof of this part also follows directly from the proof of Lemma 4.3 for star-sampler trees. We hence omit the details.

We can now finalize the proof of Lemma 4.6, by noting that for the root α_r of $\mathcal{T}$, $\boldsymbol{L}_{\alpha_r}$ is the empty-set and hence, $\mathbb{E}\left[Y\right] = \mathbb{E}\left[Y_{\alpha_r} \mid \boldsymbol{L}_{\alpha_r}\right]$, which by induction is equal to $(\#H)$. ∎

Recall that $\rho(H)$ is the fractional edge-cover number of H and it is related to $\mathcal{D}(H)$ through Eq (3).

Lemma 4.7. *For the random variable Y for* `subgraph-sampler`(G, H), $\mathbb{V}\mathrm{ar}\left[Y\right] = O(m^{\rho(H)}) \cdot \mathbb{E}\left[Y\right]$.

Proof. We bound $\mathbb{V}\mathrm{ar}\left[Y\right]$ using a similar inductive proof as in Lemma 4.6. Recall the parameters $\rho^C_1, \ldots, \rho^C_o$ and $\rho^S_1, \ldots, \rho^S_s$ associated respectively with the cycles $\mathcal{C}_1, \ldots, \mathcal{C}_o$ and stars $\mathcal{S}_1, \ldots, \mathcal{S}_s$ of the decomposition $\mathcal{D}(H)$. For simplicity of notation, for any $i \in [o + s]$, we define ρ_{i+} as follows:

$$\text{for all } i \leq o,\ \rho_{i+} := \sum_{j=i}^{o} \rho^C_j + \sum_{j=1}^{s} \rho^S_j, \qquad \text{for all } o < i \leq o + s,\ \rho_{i+} := \sum_{j=i-o}^{s} \rho^S_j.$$

We inductively show that, for any node α in an *odd layer* $2\ell-1$ of $\mathcal{T}$,

$$\mathbb{V}\text{ar}\left[Y_\alpha \mid \boldsymbol{L}_\alpha\right] \leq 2^{2z-2\ell} \cdot m^{\rho_{\ell+}} \cdot (\#H \mid \boldsymbol{L}_\alpha),$$

where $(\#H \mid \boldsymbol{L}_\alpha)$ denotes the number of copies of H in G that contain the vertices and edges specified by $\boldsymbol{L}_\alpha$ (according to the decomposition $\mathcal{D}(H)$).

The induction is from the leaf-nodes of the tree to the root. The base case of the induction, i.e., for vertices in the last odd layer of $\mathcal{T}$ follows exactly as in the proofs of Lemmas 4.2 and 4.3 (as will also become evident shortly) and hence we do not repeat it here. We now prove the induction hypothesis. Fix a vertex α in an odd layer $2\ell-1$. We consider two cases based on whether $\ell \leq o$ (hence α is root of a cycle-sampler tree) or $\ell > o$ (hence α is root of a star-sampler tree).

Case of $\ell \leq o$. In this case, the sub-tree $\mathcal{T}_\alpha$ in the next two levels is a cycle-sampler tree corresponding to the odd cycle $\mathcal{C}_\ell$ of $\mathcal{D}(H)$. Let the number of edges in C_ℓ be $(2k+1)$ (i.e., $\mathcal{C}_\ell = C_{2k+1}$) Let $\boldsymbol{e}$ denote the label of the α. By the law of total variance in Eq (1)

$$\mathbb{V}\text{ar}\left[Y_\alpha \mid \boldsymbol{L}_\alpha\right] = \mathbb{E}\left[\mathbb{V}\text{ar}\left[Y_\alpha \mid \boldsymbol{e}\right] \mid \boldsymbol{L}_\alpha\right] + \mathbb{V}\text{ar}\left[\mathbb{E}\left[Y_\alpha \mid \boldsymbol{e}\right] \mid \boldsymbol{L}_\alpha\right]. \tag{9}$$

We start by bounding the second term in Eq (9) which is easier. By the inductive proof of Lemma 4.6, we also have, $\mathbb{E}\left[Y_\alpha \mid \boldsymbol{L}_\alpha, \boldsymbol{e}\right] = (\#H \mid \boldsymbol{L}_\alpha, \boldsymbol{e})$. As such,

$$\begin{aligned}
\mathbb{V}\text{ar}\left[\mathbb{E}\left[Y_\alpha \mid \boldsymbol{e}\right] \mid \boldsymbol{L}_\alpha\right] = \mathbb{V}\text{ar}\left[(\#H \mid \boldsymbol{L}_\alpha, \boldsymbol{e}) \mid \boldsymbol{L}_\alpha\right] &\leq \mathbb{E}\left[(\#H \mid \boldsymbol{L}_\alpha, \boldsymbol{e})^2 \mid \boldsymbol{L}_\alpha\right] \\
&= \sum_{\boldsymbol{e}} \Pr\left(\mathsf{label}[\alpha] = \boldsymbol{e}\right) \cdot (\#H \mid \boldsymbol{L}_\alpha, \boldsymbol{e})^2 = \frac{1}{m^k} \sum_{\boldsymbol{e}} (\#H \mid \boldsymbol{L}_\alpha, \boldsymbol{e})^2 \\
&\qquad (\Pr\left(\mathsf{label}[\alpha] = \boldsymbol{e}\right) = 1/m^k \text{ by definition of } \texttt{odd-cycle-sampler}) \\
&\leq \frac{1}{m^k} \Big(\sum_{\boldsymbol{e}} (\#H \mid \boldsymbol{L}_\alpha, \boldsymbol{e})\Big)^2 = \frac{1}{m^k} (\#H \mid \boldsymbol{L}_\alpha)^2 \\
&\leq m^{\rho_{\ell+}} \cdot (\#H \mid \boldsymbol{L}_\alpha).
\end{aligned} \tag{10}$$

The reason behind the last equality is that $(\#H \mid \boldsymbol{L}_\alpha)$ is at most equal to the number of copies of the subgraph of H consisting of $\mathcal{C}_\ell, \ldots, \mathcal{C}_o, \mathcal{S}_1, \ldots, \mathcal{S}_s$, which by Lemma 3.2 is at most $m^{\rho_{\ell+}}$ by definition of $\rho_{\ell+}$. We now bound the first and the main term in Eq (9),

$$\begin{aligned}
\mathbb{E}\left[\mathbb{V}\text{ar}\left[Y_\alpha \mid \boldsymbol{e}\right] \mid \boldsymbol{L}_\alpha\right] &= \sum_{\boldsymbol{e}} \Pr\left(\mathsf{label}[\alpha] = \boldsymbol{e}\right) \cdot \mathbb{V}\text{ar}\left[Y_\alpha \mid \boldsymbol{e}, \boldsymbol{L}_\alpha\right] \\
&= \sum_{\boldsymbol{e}} \frac{1}{m^k} \cdot m^{2k} \cdot \frac{1}{t_{\boldsymbol{e}}^2} \cdot \sum_{i=1}^{t_{\boldsymbol{e}}} \mathbb{V}\text{ar}\left[Y_{\alpha_i} \mid \boldsymbol{e}, \boldsymbol{L}_\alpha\right], \qquad (\text{here } \alpha_i\text{'s are child-nodes of } \alpha)
\end{aligned}$$

where the final equality holds because Y_{α_i}'s are independent conditioned on $\boldsymbol{e}, \boldsymbol{L}_\alpha$ and since Y_α is by definition m^k times the average of Y_{α_i}'s. Moreover, note that distribution of all Y_{α_i}'s are the same. Hence, by canceling the terms,

$$\mathbb{E}\left[\mathbb{V}\text{ar}\left[Y_\alpha \mid \boldsymbol{e}\right] \mid \boldsymbol{L}_\alpha\right] = m^k \cdot \sum_{\boldsymbol{e}} \frac{1}{t_{\boldsymbol{e}}} \cdot \mathbb{V}\text{ar}\left[Y_{\alpha_1} \mid \boldsymbol{e}, \boldsymbol{L}_\alpha\right], \tag{11}$$

We thus only need to bound $\mathbb{V}\text{ar}\left[Y_{\alpha_1} \mid \boldsymbol{e}, \boldsymbol{L}_\alpha\right]$. Recall that α_1 corresponds to a leaf-node in a cycle-sampler tree and hence its label is a vertex w from the neighborhood of $u^*_{\boldsymbol{e}}$ as defined in `odd-cycle-sampler`. We again use the law of total variance in Eq (1) to obtain,

$$\mathbb{V}\text{ar}\left[Y_{\alpha_1} \mid \boldsymbol{e}, \boldsymbol{L}_\alpha\right] = \mathbb{E}\left[\mathbb{V}\text{ar}\left[Y_{\alpha_1} \mid w\right] \mid \boldsymbol{e}, \boldsymbol{L}_\alpha\right] + \mathbb{V}\text{ar}\left[\mathbb{E}\left[Y_{\alpha_1} \mid w\right] \mid \boldsymbol{e}, \boldsymbol{L}_\alpha\right] \tag{12}$$

For the first term,

$$\begin{aligned}\mathbb{E}\left[\mathbb{V}\mathrm{ar}\left[Y_{\alpha_1} \mid w\right] \mid \boldsymbol{e}, \boldsymbol{L}_\alpha\right] &= \sum_{w \in N(u^*_{\boldsymbol{e}})} \Pr\left(\mathsf{label}[\alpha_1] = w\right) \cdot \mathbb{V}\mathrm{ar}\left[Y_{\alpha_1} \mid w, \boldsymbol{e}, \boldsymbol{L}_\alpha\right] \\ &= \sum_{w} \frac{1}{d^*_{\boldsymbol{e}}} \cdot (d^*_{\boldsymbol{e}})^2 \cdot \mathbb{V}\mathrm{ar}\left[Y_{\beta_1} \mid w, \boldsymbol{e}, \boldsymbol{L}_\alpha\right],\end{aligned}$$

where β_1 is the unique child-node of α_1 and so $Y_{\alpha_1} = \mathsf{value}[\alpha_1] \cdot Y_{\beta_1}$, while conditioned on $\boldsymbol{e}$, $\mathsf{value}[\alpha_1] = d^*_{\boldsymbol{e}}$. Moreover, as $\boldsymbol{L}_{\beta_1} = (\boldsymbol{L}_\alpha, \boldsymbol{e}, w)$, and by canceling the terms,

$$\begin{aligned}\mathbb{E}\left[\mathbb{V}\mathrm{ar}\left[Y_{\alpha_1} \mid w\right] \mid \boldsymbol{e}, \boldsymbol{L}_\alpha\right] &= \sum_{w} d^*_{\boldsymbol{e}} \cdot \mathbb{V}\mathrm{ar}\left[Y_{\beta_1} \mid \boldsymbol{L}_{\beta_1}\right] \\ &\leq \sum_{w} d^*_{\boldsymbol{e}} \cdot 2^{2z-2\ell-2} \cdot m^{\rho_{(\ell+1)+}} \cdot (\#H \mid \boldsymbol{L}_{\beta_1}), \qquad (13)\end{aligned}$$

where the inequality is by induction hypothesis for the odd-level node β_1. We now bound the second term in Eq (12) as follows,

$$\begin{aligned}\mathbb{V}\mathrm{ar}\left[\mathbb{E}\left[Y_{\alpha_1} \mid w\right] \mid \boldsymbol{e}, \boldsymbol{L}_\alpha\right] &\leq \mathbb{E}\left[\Big(\mathbb{E}\left[Y_{\alpha_1} \mid w\right]\Big)^2 \mid \boldsymbol{e}, \boldsymbol{L}_\alpha\right] \\ &= \sum_{w} \Pr\left(\mathsf{label}[\alpha_1] = w\right) \cdot \Big(\mathbb{E}\left[Y_{\alpha_1} \mid w, \boldsymbol{e}, \boldsymbol{L}_\alpha\right]\Big)^2 \\ &= \sum_{w} \frac{1}{d^*_{\boldsymbol{e}}} \cdot (d^*_{\boldsymbol{e}})^2 \cdot \Big(\mathbb{E}\left[Y_{\beta_1} \mid w, \boldsymbol{e}, \boldsymbol{L}_\alpha\right]\Big)^2 \\ &= \sum_{w} d^*_{\boldsymbol{e}} \cdot \Big(\mathbb{E}\left[Y_{\beta_1} \mid \boldsymbol{L}_{\beta_1}\right]\Big)^2 = \sum_{w} d^*_{\boldsymbol{e}} \cdot (\#H \mid \boldsymbol{L}_{\beta_1})^2 \\ &\leq \sum_{w} d^*_{\boldsymbol{e}} \cdot m^{\rho_{(\ell+1)+}} \cdot (\#H \mid \boldsymbol{L}_{\beta_1}). \qquad (14)\end{aligned}$$

Here, the second to last equality holds by the inductive proof of Lemma 4.6, and the last equality is because $(\#H \mid \boldsymbol{L}_{\beta_1}) \leq m^{\rho_{(\ell+1)+}}$ by Lemma 3.2, as $(\#H \mid \boldsymbol{L}_{\beta_1})$ is at most equal to the total number of copies of a subgraph of H on $\mathcal{C}_{\ell+1}, \ldots, \mathcal{C}_o, \mathcal{S}_1, \ldots, \mathcal{S}_s$ (and by definition of $\rho_{(\ell+1)+}$). We now plug in Eq (13) and Eq (14) in Eq (12),

$$\mathbb{V}\mathrm{ar}\left[Y_{\alpha_1} \mid \boldsymbol{e}, \boldsymbol{L}_\alpha\right] \leq \sum_{w} d^*_{\boldsymbol{e}} \cdot \Big(2^{2z-2\ell-2} \cdot m^{\rho_{(\ell+1)+}} \cdot (\#H \mid \boldsymbol{L}_{\beta_1}) + m^{\rho_{(\ell+1)+}} \cdot (\#H \mid \boldsymbol{L}_{\beta_1})\Big).$$

We now in turn plug this in Eq (11),

$$\begin{aligned}\mathbb{E}\left[\mathbb{V}\mathrm{ar}\left[Y_\alpha \mid \boldsymbol{e}\right] \mid \boldsymbol{L}_\alpha\right] &\leq m^k \sum_{\boldsymbol{e}} \frac{1}{t_{\boldsymbol{e}}} \sum_{w} d^*_{\boldsymbol{e}} \cdot \Big(2^{2z-2\ell-2} \cdot m^{\rho_{(\ell+1)+}} \cdot (\#H \mid \boldsymbol{L}_{\beta_1}) + m^{\rho_{(\ell+1)+}} \cdot (\#H \mid \boldsymbol{L}_{\beta_1})\Big) \\ &\leq m^k \sqrt{m} \cdot \sum_{\boldsymbol{e}} \sum_{w} 2^{2z-2\ell-1} \cdot m^{\rho_{(\ell+1)+}} \cdot (\#H \mid \boldsymbol{L}_{\beta_1}) \qquad \text{(as } t_{\boldsymbol{e}} \geq d^*_{\boldsymbol{e}}/\sqrt{m}\text{)} \\ &\leq 2^{2z-2\ell-1} \cdot m^{\rho_{\ell+}} \cdot \sum_{\boldsymbol{e}} \sum_{w} (\#H \mid \boldsymbol{L}_{\beta_1}) \\ &\qquad \text{(as } \rho^C_\ell = k + 1/2 \text{ and } \rho_{\ell+} = \rho^C_\ell + \rho_{(\ell+1)+} \text{ by definition)} \\ &= 2^{2z-2\ell-1} \cdot m^{\rho_{\ell+}} \cdot (\#H \mid \boldsymbol{L}_\alpha). \qquad \text{(as } \boldsymbol{L}_{\beta_1} = (\boldsymbol{L}_\alpha, \boldsymbol{e}, w)\text{)}\end{aligned}$$

Finally, by plugging in this and Eq (10) in Eq (9),

$$\mathbb{V}\text{ar}\left[Y_\alpha \mid \boldsymbol{L}_\alpha\right] = 2^{2z-2\ell-1} \cdot m^{\rho_{\ell+}} \cdot (\#H \mid \boldsymbol{L}_\alpha) + m^{\rho_{\ell+}} \cdot (\#H \mid \boldsymbol{L}_\alpha) \leq 2^{2z-2\ell} \cdot m^{\rho_{\ell+}} \cdot (\#H \mid \boldsymbol{L}_\alpha),$$

finalizing the proof of induction step in this case. We again remark that this proof closely followed the proof for the variance of the estimator for cycle-sampler tree in Lemma 4.2.

Case of $\ell > o$. In this case, the sub-tree $\mathcal{T}_\alpha$ in the next two levels is a star-sampler tree. By the same analogy made in the proof of the previous case and Lemma 4.2, the proof of this part also follows the proof of Lemma 4.3 for star-sampler trees. We hence omit the details.

To conclude, we have that $\mathbb{V}\text{ar}\left[Y\right] = \mathbb{V}\text{ar}\left[Y_{\alpha_r} \mid \boldsymbol{L}_{\alpha_r}\right] = O(m^{\rho(H)}) \cdot (\#H) = O(m^{\rho(H)}) \cdot \mathbb{E}\left[Y\right]$ as $Y = Y_{\alpha_r}$ for the root α_r of $\mathcal{T}$, $\boldsymbol{L}_{\alpha_r} = \emptyset$, $(\#H) = \mathbb{E}\left[Y\right]$ by Lemma 4.6, and $z = O(1)$. ∎

4.2 An Algorithm for Estimating Occurrences of Arbitrary Subgraphs

We now use our estimator algorithm from the previous section to design our algorithm for estimating the occurrences of an arbitrary subgraph H in G. In the following theorem, we assume that the algorithm has knowledge of m and also a lower bound on the value of $\#H$; these assumptions can be lifted easily as we describe afterwards.

Theorem 2. *There exists a sublinear time algorithm that uses degree, neighbor, pair, and edge sample queries and given a precision parameter $\varepsilon \in (0,1)$, an explicit access to a constant-size graph $H(V_H, E_H)$, a query access to the input graph $G(V,E)$, the number of edges m in G, and a lower bound $h \leq \#H$, with high probability outputs a $(1 \pm \varepsilon)$-approximation to $\#H$ using:*

$$O\Big(\min\left\{m, \frac{m^{\rho(H)}}{h} \cdot \frac{\log n}{\varepsilon^2}\right\}\Big) \text{ queries and } O\Big(\frac{m^{\rho(H)}}{h} \cdot \frac{\log n}{\varepsilon^2}\Big) \text{ time,}$$

in the worst-case.

Proof. Fix a sufficiently large constant $c > 0$. We run `subgraph-sampler`(G,H) for $k := \frac{c \cdot m^{\rho(H)}}{\varepsilon^2 \cdot h}$ time independently in parallel to obtain estimates $Y_1, \ldots, Y_k$ and let $Z := \frac{1}{k}\sum_{i=1}^{k} Y_i$. By Lemma 4.6, $\mathbb{E}\left[Z\right] = (\#H)$. Since Y_i's are independent, we also have

$$\mathbb{V}\text{ar}\left[Z\right] = \frac{1}{k^2}\sum_{i=1}^{k} \mathbb{V}\text{ar}\left[Y_i\right] \leq \frac{1}{k} \cdot O(m^{\rho(H)}) \cdot \mathbb{E}\left[Z\right] \leq \frac{\varepsilon^2}{10} \cdot \mathbb{E}\left[Z\right]^2,$$

by Lemma 4.7, and by choosing the constant c sufficiently larger than the constant in the O-notation of this lemma, together with the fact that $h \leq (\#H) = \mathbb{E}\left[Z\right]$. By Chebyshev's inequality (Proposition 2.1),

$$\Pr\left(\left|Z - \mathbb{E}\left[Z\right]\right| \geq \varepsilon \cdot \mathbb{E}\left[Z\right]\right) \leq \frac{\mathbb{V}\text{ar}\left[Z\right]}{\varepsilon^2 \cdot \mathbb{E}\left[Z\right]^2} \leq \frac{1}{10},$$

by the bound above on the variance. This means that with probability 0.9, this algorithm outputs a $(1 \pm \varepsilon)$-approximation of $\#H$. Moreover, the expected query complexity and running time of this algorithm is $O(k)$ by Lemma 4.5, which is $O(\frac{m^{\rho(H)}}{\varepsilon^2})$ (if $k \geq m$, we simply query all edges of the graph and solve the problem using an offline enumeration algorithm). To extend this result to a high probability bound and also making the guarantee of query complexity and run-time in the worst-case, we simply run this algorithm $O(\log n)$ times in parallel and stop each execution that uses more than 10 times queries than the expected query complexity of the above algorithm. ∎

The algorithm in Theorem 2 assumes the knowledge of h which is a lower bound on $(\#H)$. However, this assumption can be easily removed by making a geometric search on h starting from $m^{\rho(H)}/2$ which is (approximately) the largest value for $(\#H)$ all the way down to 1 in factors of 2, and stopping the search once the estimates returned for a guess of h became consistent with h itself. This only increases the query complexity and runtime of the algorithm by $\text{polylog}(n)$ factors. As this part is quite standard, we omit the details and instead refer the interested reader to [19,21]. This concludes the proof of our main result in Theorem 1 from the introduction.

4.3 Extension to the Database Join Size Estimation Problem

As pointed out earlier in the paper, the database join size estimation for binary relations can be modeled by the subgraph estimation problem where the subgraph H and the underlying graph G are additionally *edge-colored* and we are only interested in counting the copies of H in G with matching colors on the edges. In this abstraction, the edges of the graph G correspond to the entries of the database, and the color of edges determine the relation of the entry.

We formalize this variant of the subgraph counting problem in the following. In the *colorful* subgraph estimation problem, we are given a subgraph $H(V_H, E_H)$ with a coloring function $c_H : E_H \rightarrow \mathbb{N}$ and query access to a graph $G(V, E)$ along with a coloring function $c_G : E \rightarrow \mathbb{N}$. The set of allowed queries to G contains the degree queries, pair queries, neighbor queries, and edge-sample queries as before, with a simple change that whenever we query an edge (through the last three types of queries), the color of the edge according to c_G is also revealed to the algorithm. Our goal is to estimate the number of copies of H in G with matching colors, i.e., the *colorful* copies of H.

It is immediate to verify that our algorithm in this section can be directly applied to the colorful subgraph estimation problem with the only difference that when testing whether a subgraph forms a copy of H in G, we in fact check whether this subgraph forms a colorful copy of H in G instead. The analysis of this new algorithm is exactly as in the case of the original algorithm with the only difference that we switch the parameter $\#H$ to $\#H_c$ that only counts the number of copies of H with the same colors in G. To summarize, we obtain an algorithm with $O^*(\frac{m^{\rho(H)}}{\#H_c})$ query and time complexity for the colorful subgraph counting problem, which can in turn solves the database join size estimation problem for binary relations.

5 Lower Bounds

In this section, we prove two separate lower bounds that demonstrate the optimality of Theorem 1 in different scenarios. Our first lower bound in Section 5.1 establishes tight bounds for estimating the number of *odd cycles*. This result implies that in addition to cliques (that were previously proved [22]; see also in [19,21]), our algorithm in Theorem 1 also achieve optimal bounds for odd cycles. Next, in Section 5.2, we target the more general problem of database join size estimation for which we argued that our Theorem 1 continues to hold. We show that for this more general problem, our algorithm in Theorem 1 is in fact optimal *for all* choices of the subgraph H.

5.1 A Lower Bound for Counting Odd Cycles

We prove that the bound achieved by Theorem 1 for any odd cycle C_{2k+1} is optimal.

Theorem 3. *For any $k \geq 1$, any algorithm $\mathcal{A}$ that can output any multiplicative-approximation to the number of copies of the odd cycle C_{2k+1} in a given graph $G(V, E)$ with probability at least $2/3$ requires $\Omega(\frac{m^{k+\frac{1}{2}}}{\#C_{2k+1}})$ queries to G.*

Our proof of Theorem 3 uses communication complexity in the two player communication model of Yao [48]. Proving query complexity lower bounds using communication complexity tools in different scenarios has a rich history (see, e.g. [8,9,22] and references therein), and was nicely formulated by Eden and Rosenbaum in a recent work [22] for graph estimation problems.

We prove Theorem 3 using a reduction from the *set disjointness* problem in communication complexity. In the set disjointness problem, there are two players Alice and Bob that are given a bit-string $X \in \{0,1\}^N$ and $Y \in \{0,1\}^N$, respectively; their goal is to determine whether there exists an index $i \in [N]$ such that $X_i \wedge Y_i = 1$, by communicating a small number of bits between each other (the players have access to a shared source of random bits, called the *public randomness*). A celebrated result in communication complexity, first proved by [34] and further refined in [5, 45], states that communication complexity of this problem, the minimum number of bits of communication needed to solve this problem with probability at least 2/3, is $\Omega(N)$. This lower bound continues to hold even for the special case where we are promised that there exists at most one index i such that $X_i \wedge Y_i = 1$.

The Reduction from Set Disjointness

For simplicity of exposition, we consider the following variant of set disjointness in which the input to Alice and Bob are two-dimensional arrays $X_{i,j}$ and $Y_{i,j}$ for $(i,j) \in ([K] \times [K]) \setminus \bigcup_{i' \in [K]} \{(i',i')\}$; the goal now is to determine whether there exists (i,j) such that $X_{i,j} \wedge Y_{i,j} = 1$ under the promise that *at most* one such index may exist. We refer to this problem as $\mathsf{Disj}(X,Y)$. It is immediate to verify that communication complexity of Disj is $\Omega(K^2)$ using a straightforward reduction from the original set disjointness problem (under the aformentioned promise) with $N := K \cdot (K-1)$.

Fix any algorithm $\mathcal{A}$ for counting the number of copies of C_{2k+1} to within any multiplicative-approximation factor. We use $\mathcal{A}$ to design a communication protocol $\Pi_{\mathcal{A}}$ for solving $\mathsf{Disj}(X,Y)$ for an appropriately chosen value of K such that communication cost of the new protocol is within a constant factor of the query complexity of $\mathcal{A}$. To do this, the players construct a graph $G_{X,Y}(V,E)$ (corresponding to inputs X,Y of Alice and Bob) *implicitly* and run $\mathcal{A}$ on $G_{X,Y}$ by answering the queries of $\mathcal{A}$ on $G_{X,Y}$ through communicating with each together. The graph $G_{X,Y}$ is (implicitly) constructed as follows (see Figure 3 for an illustration):

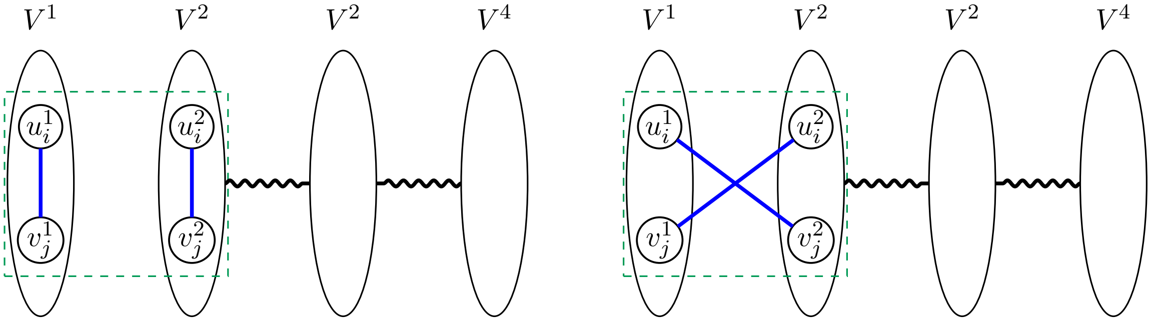
1. Partition the set of vertices V into $(k+1)$ layers $V^1, \ldots, V^{k+1}$ each of size K.
2. For every $1 < i < k+1$, connect every vertex in layer V^i to every vertex in layer V^{i+1}.
3. For every $(i,j) \in ([K] \times [K]) \setminus \bigcup_{i' \in [K]} \{(i',i')\}$, if $X_{i,j} \wedge Y_{i,j} = 1$, there exists an edge (u_i^1, v_j^1) for $u_i^1, v_j^1 \in V^1$ and another edge (u_i^2, v_j^2) for $u_i^2, v_j^2 \in V^2$.
4. For every $(i,j) \in (i,j) \in ([K] \times [K]) \setminus \bigcup_{i \in [K]} \{(i,i)\}$, if $X_{i,j} \wedge Y_{i,j} = 0$, there exists an edge (u_i^1, v_j^2) for $u_i^1 \in V^1$ and $v_j^2 \in V^2$ and another edge (u_j^1, v_i^2) for $u_j^1 \in V^1$ and $v_i^2 \in V^2$.

The following figure illustrates the graph $G_{X,Y}$ for the case of C_7.

Proposition 5.1. *For any X,Y with the promise that at most one index (i,j) have $X_{i,j} \wedge Y_{i,j} = 1$, in the graph $G_{X,Y}(V,E)$ constructed above:*

(i) The degrees of all vertices are fixed independent of the choice of X,Y.

(ii) If for all indices (i,j), $X_{i,j} \wedge Y_{i,j} = 0$, then $\#C_{2k+1} = 0$.



(a) When $X_{i,j} \wedge Y_{i,j} = 1$.

(b) When $X_{i,j} \wedge Y_{i,j} = 0$.

Figure 3: Illustration of the graph $G_{X,Y}$ for the odd cycle C_7 and the role of $X_{i,j}$ and $Y_{i,j}$ for some index (i,j) in the choice of edges in $G_{X,Y}$.

(iii) If there exists a unique index (i,j) such that $X_{i,j} \wedge Y_{i,j} = 1$, then $\#C_{2k+1} = \Omega\left(K^{2k-1}\right)$.

Proof. We prove each part separately:

(i) Follows immediately from the construction (see also Figure 3).

(ii) In this case, all edges of the graph are between V^i and V^{i+1} for some $1 \leq i < k+1$. As such, $G_{X,Y}$ is a bipartite graph with vertices in even layers in one side of the bipartition and the vertices in odd layers in the other side. This means that in this case $G_{X,Y}$ has no odd cycle.

(iii) In this case, there exists a single edge (u_i^1, v_j^1) inside V^1. By picking any pair of distinct vertices from $V^2 \setminus \left\{u_i^2, v_j^2\right\}$, any pair of distinct vertices from $V^3, \ldots, V^k$, a single vertex from V^{k+1}, and the vertices u_i^1, v_i^1 incident on this edge, we obtain a unique copy of C_{2k+1} in $G_{X,Y}$ (here, we used the assumption that the only edges missing between V^1 and V^2 are (u_i^1, v_j^2) and (u_j^1, v_i^2) by the assumption that at most one index (i,j) has $X_{i,j} \wedge Y_{i,j} = 1$). As such,

$$\#C_{2k+1} = 1 \cdot \binom{K-2}{2} \cdot \binom{K}{2}^{k-2} \cdot K \geq \left(\frac{K}{4}\right)^2 \cdot \left(\frac{K}{2}\right)^{2k-4} \cdot K = \Omega(K^{2k-1}),$$

as k is a constant.

This conclude the proof of Proposition 5.1. ∎

Now let $\mathcal{A}$ be a query algorithm for finding any multiplicative-approximation to C_{2k+1} on graphs $G_{X,Y}$ constructed above. By the first part of Proposition 5.1, we can safely assume that $\mathcal{A}$ knows degrees of all vertices in $G_{X,Y}$ as degrees of all vertices are always the same. Moreover, any edge-sample query performed by $\mathcal{A}$ can be instead performed by first sampling one of the vertices proportional to its degree (as all degrees are known to $\mathcal{A}$) and then making a random neighbor query on this vertex. As such, we assume without loss of generality that $\mathcal{A}$ only performs neighbor and pair queries. We now show how to design the protocol $\Pi_{\mathcal{A}}$ by simulating $\mathcal{A}$ on the graph $G_{X,Y}$.

The protocol $\Pi_{\mathcal{A}}$.

1. Alice and Bob use public randomness as the random bits needed by $\mathcal{A}$.
2. For every query performed by $\mathcal{A}$, the players determine the answer to the query on $G_{X,Y}$ as follows, update the state of $\mathcal{A}$ consistently, and continue to the next query.
   - *Pair query* (u, v)*:* If $u = u^1_i \in V^1$ and $v = v^2_j \in V^2$ (or vice versa), Alice communicates $X_{i,j}$ to Bob and Bob sends $Y_{i,j}$ to Alice. After this both players can determine the answer to this query by checking whether $X_{i,j} \wedge Y_{i,j} = 1$ or not. They do the same when both u, v are in V^1 or are in V^2. In any other case, the answer to the query is independent of the input to players and they can answer the query with no communication.
   - *Neighbor query* (u, j)*:* Suppose $u = u^1_i \in V^1$. If $i = j$, then the answer to the query is $v^2_j \in V^2$. Otherwise, Alice and Bob communicate $X_{i,j}$ and $Y_{i,j}$ and both players determine $X_{i,j} \wedge Y_{i,j}$. If $X_{i,j} \wedge Y_{i,j} = 0$, the answer to the query is $v^2_j \in V^2$ and otherwise it is v^1_j in V^1. This is done similarly for when $u = u^2_i \in V^2$. In any other case, the answer to the query is independent of the input to players and they can answer the query with no communication.
3. At the end, if the answer returned by $\mathcal{A}$ is non-zero, they return that there exists some index (i, j) such that $X_{i,j} \wedge Y_{i,j} = 1$ and otherwise they output no such index exists.

Proof of Theorem 3

We now prove the correctness of the protocol $\Pi_{\mathcal{A}}$ in the previous part and establish Theorem 3.

Proof of Theorem 3. Let $\mathcal{A}$ be any query algorithm for counting C_{2k+1} with probability of success at least 2/3, and let $\Pi_{\mathcal{A}}$ be the protocol created based on $\mathcal{A}$. By Proposition 5.1, for any input X, Y to $\mathsf{Disj}(X, Y)$ that satisfies the required promise, the graph $G(X, Y)$ contains a copy of C_{2k+1} iff there exists an index (i, j) such that $X_{i,j} \wedge Y_{i,j} = 1$. As such, the output of $\mathcal{A}$ on $G_{X,Y}$ (whenever correct) is non-zero iff there exists an index (i, j) such that $X_{i,j} \wedge Y_{i,j} = 1$. As the answer returned to each query of $\mathcal{A}$ in the protocol $\Pi_{\mathcal{A}}$ is consistent with the underlying graph $G_{X,Y}$, Alice and Bob can simulate $\mathcal{A}$ on $G_{X,Y}$ correctly and hence their output would be correct with probability at least 2/3. Additionally, simulating each query access of $\mathcal{A}$ requires $O(1)$ communication by players hence communication cost of $\Pi_{\mathcal{A}}$ is within constant factor of query complexity of $\mathcal{A}$.

Note that the number of edges in the graph $G_{X,Y}$ is $m = \Theta(K^2)$. By the lower bound of $\Omega(K^2)$ on the communication complexity of Disj, we obtain that query cost of $\mathcal{A}$ needs to be $\Omega(K^2) = \Omega(m)$. On the other hand, the last part of Proposition 5.1 implies that the number of copies of C_{2k+1} in G is $\Omega(m^{k-\frac{1}{2}})$. By re-parametrizing the lower bound of $\Omega(m)$ on the query complexity of $\mathcal{A}$, we obtain that $\mathcal{A}$ needs to make at least $\Omega(\frac{m^{k+\frac{1}{2}}}{\#C_{2k+1}})$, finalizing the proof. ∎

5.2 A Lower Bound for Database Join Size Estimation

Recall the colorful subgraph counting problem (the abstraction of database join size estimation problem) from Section 4.3. We prove the following theorem in this section.

Theorem 4. *For any subgraph $H(V_H, E_H)$ which contains at least one edge, suppose $\mathcal{A}$ is an algorithm for the colorful subgraph estimation problem that given H, a coloring $c_H : E_H \to \mathbb{N}$, and query access to $G(V, E)$ with m edges and coloring function $c_G : E \to \mathbb{N}$, can output a multiplicative-approximation to the number of colorful copies of H in G with probability at least $2/3$. Then, $\mathcal{A}$ requires $\Omega(\frac{m^{\rho(H)}}{\#H_c})$ queries, where $\#H_c$ is the number of colorful copies of H in G. The lower bound continues to hold even if the number of colors used by c_H and c_G is at most two.*

Recall the fractional edge-cover LP in of Section 3 (see LP (2)). The following linear program for fractional independent-set is the dual to the edge-cover LP (and hence by LP duality has the same optimal value):

$$\begin{aligned} \rho(H) \quad = \quad & \text{maximize} \quad \sum_{a \in V(H)} y_a \\ & \text{subject to} \quad y_a + y_b \leq 1 \text{ for all edges } (a, b) \in E(H). \end{aligned} \tag{15}$$

Throughout this section, we fix an optimal solution y^* of LP (15). We use y^* to design two distributions $\mathcal{G}_0$ and $\mathcal{G}_1$ on graphs G with $O(m)$ edges[2] such that any graph G sampled from $\mathcal{G}_0$, denoted by $G \sim \mathcal{G}_0$, contains no colorful copy of H (for a specific coloring of H to be described later), while any $G \sim \mathcal{G}_1$ contains many colorful copies of H. We then prove that any algorithm that makes only a small number of queries to the underlying graph cannot distinguish between graphs sampled from $\mathcal{G}_0$ and $\mathcal{G}_1$, concluding the proof.

Distributions $\mathcal{G}_0$ and $\mathcal{G}_1$

We first define the coloring c_H of H. Let $f^* := (a, b)$ be any arbitrary edge in H such that $y^*_a + y^*_b = 1$, i.e., is a tight constraint for y^* in LP (15). By optimality of y^* and as H is not a singleton vertex, such an edge f^* always exists. We now define $c_H(f^*) := 1$ and $c_H(f) := 0$ for any other edge $f \in E(H) \setminus \{f^*\}$.

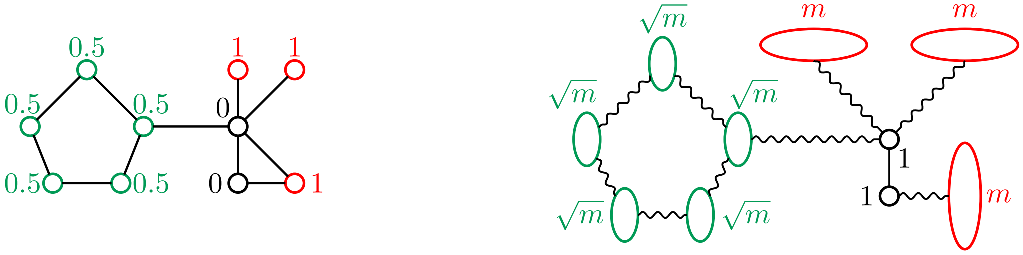
We now define the distribution $\mathcal{G}_0$. In fact, distribution $\mathcal{G}_0$ has all its mass on a single graph G_0 with coloring c_{G_0} which contains no colorful copy of H (under the coloring c_H defined above). Suppose H has $k \geq 2$ vertices denoted by $V(H) := \{a_1, \ldots, a_k\}$. The graph G_0 is constructed as follows. Firstly, the vertices of G_0 are partitioned into k sets $V(G_0) := V_1 \cup \ldots \cup V_k$ with $|V_i| = m^{y^*_{a_i}}$. Then for any edge $(a_i, a_j) \in E(H)$, we connect all vertices in V_i to all vertices in V_j in G_0. Finally, the coloring c_{G_0} of G_0 simply assigns the color 0 to *all* edges in G_0. See Figure 4 for an illustration.

The distribution $\mathcal{G}_1$ is constructed similarly (but on a larger support). Let G_0 be the single graph constructed by $\mathcal{G}_0$. Any graph $G \sim \mathcal{G}_1$ is constructed as follows: we first let $G = G_0$ and then choose a single edge e^* uniformly at random from the edges between V_i and V_j where (i, j) is chosen such that $f^* = (a_i, a_j)$ (recall the definition of edge f^* above). We then change the color $c_G(e^*) = 1$ (all other edges are still assigned the color 0). This concludes the description of distributions $\mathcal{G}_0$ and $\mathcal{G}_1$. We now present basic properties of these distributions.

Proposition 5.2. *For the two distributions $\mathcal{G}_0$ and $\mathcal{G}_1$:*

(i) Every graph G sampled from $\mathcal{G}_0$ or $\mathcal{G}_1$ contains $\Theta(m)$ edges.

[2] For simplicity of exposition, we let the graphs contain $O(m)$ edges instead of exactly m edges (but provide the algorithm with the exact number of edges in the graph); a simple rescaling of the bound immediately proves the lower bound for the case of graphs with exactly m edges as well.



(a) The subgraph H. The number next to each vertex a denotes y^*_a.

(b) The graph G_0 of $\mathcal{G}_0$. The number next to each block of vertices denotes the size of the block.

Figure 4: Illustration of the graph G_0 in distribution $\mathcal{G}_0$.

(ii) *The graph $G_0 \sim \mathcal{G}_0$ contains no colorful copies of H, while any graph $G \sim \mathcal{G}_1$ contains $m^{\rho(H)-1}$ colorful copies of H.*

(iii) *For a graph $G \sim \mathcal{G}_1$, the edge e^* is chosen uniformly at random among the m edges between V_i and V_j.*

Proof. We prove each part separately below.

(i) The number of edges sampled from the distributions $\mathcal{G}_0$ and $\mathcal{G}_1$ is the same, hence it only suffices to prove the bound for the (unique) graph G_0 in the support of $\mathcal{G}_0$. For any edge (a_i, a_j) in H, we have a bipartite clique between V_i and V_j in G_0, hence resulting in $|V_i| \cdot |V_j| = m^{y^*_{a_i}} \cdot m^{y^*_{a_j}} \leq m$ edges in G, where the final inequality is because y^* is a feasible solution of LP (15). As such, the number of edges in G_0 is $O(m)$ as size of H is constant.

(ii) There is no edge with color 1 in $G_0 \sim \mathcal{G}_0$, while H has an edge with color 1 and hence G_0 contains no colorful copy of H. On the other hand, in any graph $G \sim \mathcal{G}_1$, we can create a copy of H by mapping each vertex a_i of $V(H)$ which is not incident to f^* to any arbitrary vertex in V_i and then maping the edge f^* of H to e^* in G. Suppose $f^* = (a, b)$. The total number of colorful copies of H in G is then $\prod_{a_i \in V(H)\setminus\{a,b\}} |V_i| = m^{\sum_{a_i \in V(H)\setminus\{a,b\}} y^*_{a_i}} = m^{\rho(H)-1}$ as $\sum_{a_i \in V(H)} y^*_{a_i} = \rho(H)$ and $y^*_a + y^*_b = 1$.

(iii) The fact that e^* is chosen uniformly at random is by definition of distribution $\mathcal{G}_1$. The total number of edges between V_i and V_j where e^* is chosen from is $|V_i| \cdot |V_j| = m^{y^*_{a_i} + y^*_{a_j}} = m$ by the choice of $f^* = (a_i, a_j)$.

This concludes the proof of Proposition 5.2. ∎

Query Complexity of Distinguishing $\mathcal{G}_0$ and $\mathcal{G}_1$

We now prove that any query algorithm that can distinguish between instances sampled from $\mathcal{G}_0$ and $\mathcal{G}_1$ requires $\Omega(m)$ queries, proving the following lemma.

Lemma 5.3. *Define the distribution $\mathcal{G} := \frac{1}{2} \cdot \mathcal{G}_0 + \frac{1}{2} \cdot \mathcal{G}_1$. Suppose $\mathcal{A}$ is any algorithm that given a graph $G \sim \mathcal{G}$ with probability at least $2/3$ determines whether it belongs to (the support of) $\mathcal{G}_0$ or $\mathcal{G}_1$. Then $\mathcal{A}$ needs to make $\Omega(m)$ queries to the graph.*

Proof. We assume that $\mathcal{A}$ knows the partitioning of vertices of G into $V_1, \ldots, V_{|V(H)|}$ and is hence even aware of the set of edges in G (but not their colors); this can only strengthen our lower bound.

Assume $f^* = (a_i, a_j)$ and note that the only difference between the graphs in $\mathcal{G}_0$ and $\mathcal{G}_1$ is that the latter graphs have an edge e^* between V_i and V_j that is colored 1 instead of 0. This implies that the only "useful" queries performed by $\mathcal{A}$ are pair queries between vertices $u \in V_i$ and $v \in V_j$ (degree queries can be answered without querying the graph; neighbor queries can be simulated by a pair query as the set of neighbors are all known in advance; edge-sample queries can also be performed by pair queries by sampling one of the known edges uniformly at random and then querying the edge to determine its color).

Suppose towards a contradiction that $\mathcal{A}$ is an algorithm (possibly randomized) that given a graph $G \sim \mathcal{G}$ uses $o(m)$ queries and can determine whether G belongs to $\mathcal{G}_0$ or $\mathcal{G}_1$ with probability at least $2/3$. By fixing the randomness of this algorithm and an averaging argument (namely, the easy direction of Yao's minimax principle [49]), we obtain a deterministic algorithm $\mathcal{A}'$ that uses the same number of queries as $\mathcal{A}$ and output the correct answer with probability $2/3$, where the probability is now only taken over the randomness of the distribution $\mathcal{G}$.

Let $Q := (q_1, q_2, \ldots, q_\ell)$ for $\ell = o(m)$ determines the (potentially adaptively chosen) set of queries performed by $\mathcal{A}'$ before it outputs the answer. Since the set of edges in the graph are already known to $\mathcal{A}$, the only interesting part of the answer to each query q_i is whether the color of the edge queried by q_i is 0 or 1. With a slight abuse of notation, we write $q_i = 1$ if the color of the edge queried by q_i is 1 and $q_i = 0$ otherwise.

Notice that since $\mathcal{A}'$ is a deterministic algorithm, the next query q_i is determined solely based on the answer to queries $q_1, \ldots, q_{i-1}$. Let $\mathbf{0}_k$ denote the vector of all zeros of length k. As a result,

$$\Pr_{G \sim \mathcal{G}_1}\left[q_i = 1 \mid (q_1, \ldots, q_{i-1}) = \mathbf{0}_{i-1}\right] = \frac{1}{m - i + 1}.$$

This is because, conditioned on all $(q_1, \ldots, q_{i-1}) = \mathbf{0}_{i-1}$, the next query chosen by $\mathcal{A}'$ is fixed beforehand and is only based on the knowledge that the $i - 1$ edges queried so far cannot be e^*. As e^* is chosen uniformly at random from a set of m edges (by Part (iii) of Proposition 5.2), the bound above holds (note that we assumed without loss of generality that $\mathcal{A}'$ does not query an edge more than once). As a result of this, we have,

$$\Pr_{G \sim \mathcal{G}_1}\left[(q_1, \ldots, q_\ell) = \mathbf{0}_\ell\right] = \frac{m-1}{m} \cdot \frac{m-2}{m-1} \cdot \ldots \cdot \frac{m-\ell}{m-\ell-1} = 1 - \frac{\ell}{m}. \tag{16}$$

Let $O(q_1, \ldots, q_\ell) \in \{0, 1\}$ denote the output of $\mathcal{A}'$ based on the answers given to the queries $q_1, \ldots, q_\ell$. We have,

$$\Pr_{G \sim \mathcal{G}}\left[\mathcal{A}' \text{ is correct on } G\right] = \frac{1}{2} \cdot \Pr_{G \sim \mathcal{G}_0}\left[O(q_1, \ldots, q_\ell) = 0\right] + \frac{1}{2} \cdot \Pr_{G \sim \mathcal{G}_1}\left[O(q_1, \ldots, q_\ell) = 1\right] \tag{17}$$

The second term in RHS above can be upper bounded by,

$$\begin{aligned}\Pr_{G \sim \mathcal{G}_1}\left[O(q_1, \ldots, q_\ell) = 1\right] &\leq \Pr_{G \sim \mathcal{G}_1}\left[(q_1, \ldots, q_\ell) = \mathbf{0}_\ell\right] \cdot \Pr_{G \sim \mathcal{G}_1}\left[O(q_1, \ldots, q_\ell) = 1 \mid (q_1, \ldots, q_\ell) = \mathbf{0}_\ell\right] \\ &\qquad + \left(1 - \Pr_{G \sim \mathcal{G}_1}\left[(q_1, \ldots, q_\ell) = \mathbf{0}_\ell\right]\right) \\ &= \left(1 - \frac{\ell}{m}\right) \cdot \Pr_{G \sim \mathcal{G}_1}\left[O(q_1, \ldots, q_\ell) = 1 \mid (q_1, \ldots, q_\ell) = \mathbf{0}_\ell\right] + \frac{\ell}{m},\end{aligned}$$

by Eq (16). Plugging in this bound in Eq (17) implies that,

$$\Pr_{G\sim\mathcal{G}}\left[\mathcal{A}' \text{ is correct on } G\right] \leq \frac{1}{2}\cdot \Pr_{G\sim\mathcal{G}_0}\left[O(q_1,\ldots,q_\ell)=0\right] + \frac{1}{2}\cdot \Pr_{G\sim\mathcal{G}_1}\left[O(q_1,\ldots,q_\ell)=1 \mid (q_1,\ldots,q_\ell)=\mathbf{0}_\ell\right] + \frac{\ell}{2m}.$$

We argue that either $\Pr_{G\sim\mathcal{G}_1}\left[O(q_1,\ldots,q_\ell)=1 \mid (q_1,\ldots,q_\ell)=\mathbf{0}_\ell\right]$ or $\Pr_{G\sim\mathcal{G}_0}\left[O(q_1,\ldots,q_\ell)=0\right]$ must be 0. This is because in both cases, $(q_1,\ldots,q_\ell)=\mathbf{0}_\ell$ and hence $O(q_1,\ldots,q_\ell)$ is fixed to be either 0 or 1 at this point. As a result,

$$\Pr_{G\sim\mathcal{G}}\left[\mathcal{A}' \text{ is correct on } G\right] \leq \frac{1}{2} + \frac{\ell}{2m} = \frac{1}{2} + o(1).$$

This contradicts the fact that $\mathcal{A}'$ outputs the correct answer with probability at least $2/3$, implying that ℓ needs to be $\Omega(m)$. ▌

Proof of Theorem 4

We can now finalize the proof of Theorem 4 using Proposition 5.2 and Lemma 5.3.

Proof of Theorem 4. Firstly, any algorithm that can provide any multiplicative-approximation to the number of colorful copies of H in graphs G must necessarily distinguish between the graphs chosen from distributions $\mathcal{G}_0$ and $\mathcal{G}_1$ because by Part (i) of Proposition 5.2, graphs in $\mathcal{G}_0$ contain no colorful copies of H while graphs in $\mathcal{G}_1$ contain $m^{\rho(H)-1}$ colorful copies of H. Moreover, in the graphs chosen from $\mathcal{G}_1$, $\#H_c = m^{\rho(H)-1}$. The lower bound of $\Omega(m^{\rho(H)}/\#H_c)$ on the query complexity of algorithms now follows from the $\Omega(m)$ lower bound of Lemma 5.3. ▌

Acknowledgements

We are thankful to the anonymous reviewers of ITCS 2019 for many valuable comments.

References

[1] N. K. Ahmed, J. Neville, and R. R. Kompella. Network sampling: From static to streaming graphs. *TKDD*, 8(2):7:1–7:56, 2013.

[2] M. Aliakbarpour, A. S. Biswas, T. Gouleakis, J. Peebles, R. Rubinfeld, and A. Yodpinyanee. Sublinear-time algorithms for counting star subgraphs via edge sampling. *Algorithmica*, 80(2):668–697, 2018.

[3] N. Alon. On the number of subgraphs of prescribed type of graphs with a given number of edges. *Israel Journal of Mathematics*, 1981.

[4] A. Atserias, M. Grohe, and D. Marx. Size bounds and query plans for relational joins. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 739–748. IEEE Computer Society, 2008.

[5] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. Information theory methods in communication complexity. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity, Montréal, Québec, Canada, May 21-24, 2002*, pages 93–102, 2002.

[6] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA.*, pages 623–632, 2002.

[7] S. K. Bera and A. Chakrabarti. Towards tighter space bounds for counting triangles and other substructures in graph streams. In *34th Symposium on Theoretical Aspects of Computer Science, STACS 2017, March 8-11, 2017, Hannover, Germany*, pages 11:1–11:14, 2017.

[8] E. Blais, J. Brody, and K. Matulef. Property testing lower bounds via communication complexity. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8-10, 2011*, pages 210–220, 2011.

[9] E. Blais, C. L. Canonne, and T. Gur. Distribution testing lower bounds via reductions from communication complexity. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, pages 28:1–28:40, 2017.

[10] E. Bloedorn, N. Rothleder, D. DeBarr, and L. Rosen. Relational Graph Analysis with Real-World Constraints: An Application in IRS Tax Fraud Detection. In *AAAI*, 2005.

[11] V. Braverman, R. Ostrovsky, and D. Vilenchik. How hard is counting triangles in the streaming model? In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 244–254, 2013.

[12] L. S. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, and C. Sohler. Counting triangles in data streams. In *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*, pages 253–262, 2006.

[13] S. Burt. Structural Holes and Good Ideas. *The American Journal of Sociology*, 110(2):349–399, 2004.

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

[15] N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. Comput.*, 14(1):210–223, 1985.

[16] G. Cormode and H. Jowhari. A second look at counting triangles in graph streams (corrected). *Theor. Comput. Sci.*, 683:22–30, 2017.

[17] 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 J. Comput.*, 35(1):91–109, 2005.

[18] A. Czumaj and C. Sohler. Estimating the weight of metric minimum spanning trees in sublinear-time. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 175–183, 2004.

[19] T. Eden, A. Levi, D. Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 614–633, 2015.

[20] T. Eden, D. Ron, and C. Seshadhri. Sublinear time estimation of degree distribution moments: The degeneracy connection. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 7:1–7:13, 2017.

[21] T. Eden, D. Ron, and C. Seshadhri. On approximating the number of k-cliques in sublinear time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 722–734, 2018.

[22] T. Eden and W. Rosenbaum. Lower bounds for approximating graph parameters via communication complexity. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, pages 11:1–11:18, 2018.

[23] T. Eden and W. Rosenbaum. On sampling edges almost uniformly. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, pages 7:1–7:9, 2018.

[24] U. Feige. On sums of independent random variables with unbounded variance, and estimating the average degree in a graph. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 594–603, 2004.

[25] E. Friedgut and J. Kahn. On the number of copies of one hypergraph in another. *Israel Journal of Mathematics*, 1998.

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

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

[28] M. Gonen, D. Ron, and Y. Shavitt. Counting stars and other small subgraphs in sublinear time. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 99–116, 2010.

[29] A. Hassidim, J. A. Kelner, H. N. Nguyen, and K. Onak. Local graph partitions for approximation and testing. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 22–31, 2009.

[30] M. Jha, C. Seshadhri, and A. Pinar. A space efficient streaming algorithm for triangle counting using the birthday paradox. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 589–597, 2013.

[31] H. Jowhari and M. Ghodsi. New streaming algorithms for counting triangles in graphs. In *Computing and Combinatorics, 11th Annual International Conference, COCOON 2005, Kunming, China, August 16-29, 2005, Proceedings*, pages 710–716, 2005.

[32] J. Kallaugher, M. Kapralov, and E. Price. The sketching complexity of graph and hypergraph counting. *To appear in FOCS*, 2018.

[33] J. Kallaugher and E. Price. A hybrid sampling scheme for triangle counting. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1778–1797, 2017.

[34] B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discrete Math.*, 5(4):545–557, 1992.

[35] D. M. Kane, K. Mehlhorn, T. Sauerwald, and H. Sun. Counting arbitrary subgraphs in data streams. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, pages 598–609, 2012.

[36] T. Kaufman, M. Krivelevich, and D. Ron. Tight bounds for testing bipartiteness in general graphs. *SIAM J. Comput.*, 33(6):1441–1483, 2004.

[37] J. Lee and J. Pfeffer. Estimating centrality statistics for complete and sampled networks: Some approaches and complications. In *48th Hawaii International Conference on System Sciences, HICSS 2015, Kauai, Hawaii, USA, January 5-8, 2015*, pages 1686–1695, 2015.

[38] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 631–636, 2006.

[39] A. McGregor, S. Vorotnikova, and H. T. Vu. Better algorithms for counting triangles in data streams. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 401–411, 2016.

[40] 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, October 2002.

[41] H. Q. Ngo, E. Porat, C. Ré, and A. Rudra. Worst-case optimal join algorithms. *J. ACM*, 65(3):16:1–16:40, 2018.

[42] H. N. Nguyen and K. Onak. Constant-time approximation algorithms via local improvements. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 327–336, 2008.

[43] 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 Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1123–1131, 2012.

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

[45] A. A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992.

[46] O. Simpson, C. Seshadhri, and A. McGregor. Catching the head, tail, and everything in between: A streaming algorithm for the degree distribution. In *2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015*, pages 979–984, 2015.

[47] J. Ugander, L. Backstrom, and J. Kleinberg. Subgraph frequencies: Mapping the empirical and extremal geography of large graph collections. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1307–1318, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[48] A. C. Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 209–213, 1979.

[49] A. C. Yao. Lower bounds by probabilistic arguments (extended abstract). In *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983*, pages 420–428, 1983.

[50] Y. Yoshida, M. Yamamoto, and H. Ito. An improved constant-time approximation algorithm for maximum matchings. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 225–234, 2009.

A Missing Details and Proofs

A.1 Proof of Proposition 2.2

Proposition 2.2 (restated here for convenience of the reader) follows from standard graph theory facts (see, e.g. Lemma 2 in [15]). We give a self-contained proof here for completeness.

Proposition (Proposition 2.2 in Section 2)**.** *For any graph G, $\sum_{(u,v)\in E}\min(d_u,d_v) \leq 5m\sqrt{m}$.*

Proof. Let V^+ be the set of vertices with degree more than $\sqrt{m}$ and $V^- := V \setminus V^+$.

$$\begin{aligned}\sum_{(u,v)\in E}\min(d_u,d_v) &= \frac{1}{2}\cdot\sum_{u\in V}\sum_{v\in N(u)}\min(d_u,d_v) \leq \frac{1}{2}\cdot\left(\sum_{u\in V^-}\sum_{v\in N(u)}\sqrt{m} + \sum_{u\in V^+}\sum_{v\in N(u)}\min(d_u,d_v)\right)\\ &\leq m\sqrt{m} + \frac{1}{2}\cdot\sum_{u\in V^+}\left(\sum_{v\in N(u):d_v<d_u} d_v + \sum_{v\in N(u):d_v\geq d_u} d_u\right)\\ &\leq m\sqrt{m} + \frac{1}{2}\cdot\sum_{u\in V^+}\left(2m + \frac{2m}{d_u}\cdot d_u\right) \leq m\sqrt{m} + \sqrt{m}\cdot 4m,\end{aligned}$$

where the second last inequality is because, sum of degrees of vertices in $N(u)$ is $\leq 2m$, and number of vertices with degree more than d_u is $\leq 2m/d_u$ and the last inequality is by $|V^+| \leq 2\sqrt{m}$. ∎

A.2 Proof of Lemma 3.1

We now provide a self-contained proof of Lemma 3.1 (restated below) for completeness.

Lemma (Lemma 3.1 in Section 3)**.** *Any subgraph H admits an optimal fractional edge-cover x^* such that the support of x^*, denoted by $\textsc{supp}(x^*)$, is a collection of vertex-disjoint odd cycles and star graphs, and,*

1. *for every odd cycle $C \in \textsc{supp}(x^*)$, $x^*_e = 1/2$ for all $e \in C$;*
2. *for every edge $e \in \textsc{supp}(x^*)$ that does* not *belong to any odd cycle, $x_e = 1$.*

To prove Lemma 3.1, we first state a basic property of LP (2).

Proposition A.1. *LP (2) admits a half-integral optimum solution $x^* \in \left\{0, \frac{1}{2}, 1\right\}^{|E(H)|}$. Moreover, if H is bipartite, then LP (2) admits an integral optimum solution.*

Proof. Suppose first that H is bipartite and $x \in [0,1]^{|E(H)|}$ is some optimal solution of LP (2). We perform a simple cycle-canceling on x to make it integral. In particular, let $e_1, \ldots, e_{2k}$ for some integer $k \geq 2$ be a cycle in the support of x (as H is bipartite length of this cycle is necessarily even). We can alternatively increase the value on one edge and decrease the value on the next one by the same amount and continue along the cycle until the value on an edge drops to zero. This operation clearly preserves the feasibility as well as the value of the solution. By doing this, we can cancel all cycles in the support of x without changing the value of LP or violating the feasibility. At this point, support of x is a forest and can be turned into an integral solution using a standard deterministic rounding in a bottom up approach from the leaf-nodes of the forest (see the proof of Lemma 3.1 for more details on this standard procedure).

Now suppose H is a non-bipartite graph. Create the following bipartite graph H' where $V(H')$ consists of two copies of vertices in H, i.e., for any vertex $a \in V(H)$, there are two copies, say, a^L and a^R in $V(H')$. Moreover, for any edge $e := (a,b) \in E(H)$ there are two edges $e_1 := (a^L, b^R)$ and $e_2 := (a^R, b^L)$ in $E(H')$. It is easy to see that any edge cover y of H' can be translated to an edge cover x of H by setting $x_e = \frac{y_{e_1} + y_{e_2}}{2}$. As by the first part, H' admits an integral optimum solution, we immediately have that H admits a half-integral optimum solution. ∎

Proof of Lemma 3.1. Let x^* be a half-integral optimum solution for the graph H that is guaranteed to exist by Proposition A.1. Let C be any cycle (odd or even length) in $\textsc{supp}(x^*)$. For any edge $e \in C$, $x^*_e = 1/2$ as otherwise by decreasing x^*_e from 1 to 1/2 (recall that x^* is half-integral and $x^*_e \neq 0$), we can reduce the optimal solution without violating the feasibility. Moreover, if C is of even length, then we can perform a standard cycle canceling (by adding and subtracting 1/2 to the value of x^* on the alternate edges of C) and remove the cycle. Now suppose C is of odd length; we argue that for each vertex $a \in C$, the only edges in $\textsc{supp}(x^*)$ that are incident on a are edges in C.

Suppose by contradiction that there exists an edge e with $x^*_e \geq 1/2$ which is incident on a vertex a in an odd cycle C (in $\textsc{supp}(x^*)$). Perform a cycle canceling as follows: subtract 1/2 from every other edge starting from an edge incident to a and add 1/2 to every other edge *plus* the edge e. As C is an odd cycle, the total number of addition and subtractions are equal and hence does not change the value of x^*. It is also easy to verify that the new x^* is still feasible as $x^*_e \geq 1$ now and hence a is covered still. Thus, by repeatedly applying the above argument, we can change x^* so that $\textsc{supp}(x^*)$ consists of a vertex-disjoint union of odd cycles (with $x^*_e = 1/2$) and forests. We now turn the forests into a collection of starts using a simple deterministic rounding.

For each tree T in this forest, we root the tree arbitrarily at some degree one vertex. Any edge e incident on leaf-nodes of this tree clearly has $x^*_e = 1$. Let f be a parent edge e and z be a parent of f (if these edges do not exist, e belongs to a cycle and we are already done). Let $x^*_z \leftarrow x^*_z + x^*_f$ and $x^*_f \leftarrow 0$. This preserves both the value of x^* and its feasibility, and further partition this tree into a forest and a star. By repeatedly applying this argument, we can decompose every forest into a collection of stars, finalizing the proof of Lemma 3.1. ∎

A.3 An Alternate Analysis of the Variance of the Estimator for Stars

Recall that in Lemma 4.3, we upper bounded the variance of the random variable X associated with `star-sampler`(G, S_ℓ) with $\mathbb{V}\text{ar}\left[X\right] \leq 2m^\ell \cdot \mathbb{E}\left[X\right]$. Using this analysis in our Theorem 2 results in an upper bound of $O(\frac{m^\ell}{\#S_\ell})$ on the query complexity of counting stars which is suboptimal. We now show that a slightly improved analysis of the variance in fact results in an algorithm with $O^*(\frac{m}{(\#S_\ell)^{1/\ell}})$ query complexity which is optimal by a result of [2].

Lemma A.2. *For the random variable X associated with* `star-sampler`(G, S_ℓ)*,*

$$\mathbb{E}\left[X\right] = (\#S_\ell), \qquad \mathbb{V}\text{ar}\left[X\right] \leq 4m \cdot \ell^{2\ell} \cdot (\#S_\ell)^{2-1/\ell}.$$

Proof. The bound on the expectation is already establish in Lemma 4.3. We now prove the bound on variance. This proves the desired bound on the exception. We now bound $\mathbb{V}\text{ar}\left[X\right]$.

$$\begin{aligned}
\mathbb{V}\text{ar}\left[X\right] \leq \mathbb{E}\left[X^2\right] &= \sum_{v \in V} \sum_{\boldsymbol{w} \in N(v)^\ell} \Pr\left(\mathsf{label}[\alpha_r] = v\right) \cdot \Pr\left(\mathsf{label}[\alpha_l] = \boldsymbol{w}\right) \\
&\qquad\qquad\qquad \cdot \mathbb{I}((v, \boldsymbol{w}) \text{ forms a copy of } S_\ell) \cdot (\mathsf{value}[\alpha_r] \cdot \mathsf{value}[\alpha_l])^2 \\
&= \sum_v \frac{d_v}{2m} \cdot \sum_{\boldsymbol{w}} \frac{1}{\binom{d_v}{\ell}} \cdot \mathbb{I}((v, \boldsymbol{w}) \text{ forms a copy of } S_\ell) \cdot \left((2m/d_v) \cdot \binom{d_v}{\ell}\right)^2 \\
&= \sum_v \sum_{\boldsymbol{w}} \mathbb{I}((v, \boldsymbol{w}) \text{ forms a copy of } S_\ell) \cdot (2m/d_v) \cdot \binom{d_v}{\ell} \\
&\leq 2m \cdot \sum_v \sum_{\boldsymbol{w}} \mathbb{I}((v, \boldsymbol{w}) \text{ forms a copy of } S_\ell) \cdot {d_v}^{\ell-1} && \text{(since } \tbinom{d_v}{\ell} \leq {d_v}^\ell\text{)} \\
&= 2m \cdot \sum_{v : d_v \geq \ell} \binom{d_v}{\ell} \cdot {d_v}^{\ell-1} \\
&\text{(since } \textstyle\sum_{\boldsymbol{w}} \mathbb{I}((v, \boldsymbol{w}) \text{ forms a copy of } S_\ell) = \binom{d_v}{\ell} \text{ for any } v \text{ with } d_v \geq \ell \text{ and is 0 otherwise)} \\
&= 2m \cdot \sum_{v : d_v \geq \ell} \left({d_v}^\ell\right)^{2-1/\ell} && \text{(since } \tbinom{d_v}{\ell} \leq {d_v}^\ell\text{)} \\
&\leq 2m \cdot \Big(\sum_{v : d_v \geq \ell} {d_v}^\ell\Big)^{2-1/\ell} && \text{(since } \textstyle\sum_a a^b \leq (\sum_a a)^b \text{ for } b \geq 1\text{)} \\
&\leq 2m \cdot \Big(\sum_{v : d_v \geq \ell} \ell^\ell \cdot \binom{d_v}{\ell}\Big)^{2-1/\ell} && \text{(since } \tbinom{d_v}{\ell} \cdot \ell^\ell \geq {d_v}^\ell\text{)} \\
&\leq 4m \cdot \ell^{2\ell} \cdot (\#S_\ell)^{2-1/\ell},
\end{aligned}$$

where the last inequality is because $\sum_{v : d_v \geq \ell} \binom{d_v}{\ell}$ is equal to $\#S_\ell$ when $\ell > 1$ and is equal to $2 \cdot (\#S_\ell)$ when $\ell = 1$. ∎

Using this lemma, the only change we need to do with our algorithm in Theorem 2 for improving its performance when counting stars is that instead of taking average of $O(\frac{m^\ell}{\#S_\ell})$ estimators, we only need to take average of $O(\frac{m}{\#S_\ell^{1/\ell}})$ many of them. The proof of correctness now follows exactly as before by Chebyshev's inequality (Proposition 2.1) as ℓ is a constant. With this minor modification, our algorithm then needs $O(\frac{m}{\#S_\ell^{1/\ell}})$ queries to compute a $(1 \pm \varepsilon)$-approximation of the number of occurrences of the star S_ℓ in any given graph G.