Towards a Decomposition-Optimal Algorithm for Counting and Sampling Arbitrary Motifs in Sublinear Time

Amartya Shankha Biswas
CSAIL, Massachusetts Institute of Technology, Cambridge MA, USA

Talya Eden
CSAIL, Massachusetts Institute of Technology, Cambridge MA, USA

Ronitt Rubinfeld
CSAIL, Massachusetts Institute of Technology, Cambridge MA, USA

Abstract

We consider the problem of sampling and approximately counting an arbitrary given motif H in a graph G, where access to G is given via queries: degree, neighbor, and pair, as well as uniform edge sample queries. Previous algorithms for these tasks were based on a decomposition of H into a collection of odd cycles and stars, denoted $D^*(H) = \{O_{k_1}, \ldots, O_{k_q}, S_{p_1}, \ldots, S_{p_\ell}\}$. These algorithms were shown to be optimal for the case where H is a clique or an odd-length cycle, but no other lower bounds were known.

We present a new algorithm for sampling arbitrary motifs which, up to $\text{poly}(\log n)$ factors, is always at least as good, and for most graphs G is strictly better. The main ingredient leading to this improvement is an improved uniform algorithm for sampling stars, which might be of independent interest, as it allows to sample vertices according to the p-th moment of the degree distribution.

Finally, we prove that this algorithm is *decomposition-optimal* for decompositions that contain at least one odd cycle. These are the first lower bounds for motifs H with a nontrivial decomposition, i.e., motifs that have more than a single component in their decomposition.

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

Keywords and phrases Sublinear time algorithms, Graph algorithms, Sampling subgraphs, Approximate counting

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2021.55

Category RANDOM

Related Version *Full Version*: `https://arxiv.org/abs/2107.06582`

Funding *Amartya Shankha Biswas*: Big George Ventures Fund, MIT-IBM Watson AI Lab and Research Collaboration Agreement No. W1771646, NSF awards CCF-173380, CCF-2006664 and IIS-1741137.
Talya Eden: This work was supported by the NSF grant CCF-1740751, Eric and Wendy Schmidt Fund, and Ben-Gurion University.
Ronitt Rubinfeld: This work was supported the NSF TRIPODS program (awards CCF-1740751 and DMS 2022448), NSF award CCF-2006664 and by the Fintech@CSAIL Initiative.

Acknowledgements Talya Eden is thankful to Dana Ron and Oded Goldreich for their valuable suggestions regarding the presentation of the lower bound results. The authors are thankful for the anonymous reviewers for their useful comments and observations.

1 Introduction

The problems of counting and sampling small motifs in graphs are fundamental algorithmic problems with many applications. Small motifs statistics are used for the study and characterization of graphs in multiple fields, including biology, chemistry, social networks and many others (see e.g., [36, 30, 21, 33, 32, 43, 28, 35, 38, 41, 31]). From a theoretical perspective, the complexity of the best known classical algorithms for exactly enumerating small motifs such as cliques and paths of length k, grows exponentially with k [42, 9]. On the more applied side, there is an extensive study of practical algorithms for approximate motif counting (e.g., [39, 5, 34, 1, 27, 12, 7, 24]). We study the problems of approximate motif counting and uniform sampling in the *sublinear-time* setting, where sublinear is with respect to the size of the graph. We consider the *augmented query model*, introduced by [2], where the allowed queries are degree, neighbor and pair queries as well as uniform edge sample queries.[1] We note that the model which only allows for the first three types of queries is referred to as the *general graph query model*, introduced by [29].

The problems of approximate counting and uniformly sampling of *arbitrary motifs* of constant size in sublinear-time have seen much progress recently, through the results of Assadi, Kapralov and Khanna [3], and Fichtenberger, Gao and Peng [23]. The algorithms of [3, 23] both start by computing an optimal (in a sense that will be clear shortly) decomposition of the motif H into vertex-disjoint odd cycles and stars, defined next.

A decomposition into odd cycles and stars. A decomposition D of a motif (graph) H into a collection of vertex disjoint small cycles and stars $\{O_{k_1}, \ldots, O_{k_q}, S_{p_1}, \ldots, S_{p_\ell}\}$ is valid if all vertices of H belong to either a star or an odd cycle in the collection. Each decomposition can be associated with a weight function $f_D : E \to \{0, \frac{1}{2}, 1\}$ which assigns weight 1 to edges of its star components, weight 1/2 to edges of its odd cycle components and weight 0 to all other edges in H. See figure 1 for an illustration. Hence, each decomposition $\{O_{k_1}, \ldots, O_{k_q}, S_{p_1}, \ldots, S_{p_\ell}\}$ has value $\rho(D) = \sum_{e \in H} f_D(e) = \sum_{i=1}^{q} k_i/2 + \sum_{j=1}^{\ell} p_j$, where throughout the paper k_i and p_j denote the length and number of petals in the i^{th} cycle and j^{th} star, respectively, in $D^*(H)$. For every H, its optimal decomposition value is $\rho(H) = \min_D\{\rho(D)\}$, and a decomposition D is said to be *optimal* for H if $\rho(D) = \rho(H)$. We fix (one of) the optimal decomposition of H, and denote it by $D^*(H)$. In [3], it is shown that an optimal decomposition of a motif H can be computed in polynomial time in $|H|$.[2]

The algorithm in [23] has expected running time [3] $O\left(\frac{m^{\rho(H)}}{\bar{\mathbf{h}}}\right)$ for the task of uniformly

[1] Degree queries return the degree of the queried vertex, neighbor queries with index $i \leq d(v)$ return the i^{th} neighbor of the queried vertex, pair queries return whether there is an edge between the queried pair of vertices, and uniform edge queries return a uniformly distributed edge in the graph.

[2] We note that $\rho(H)$ is equal to the fractional edge cover value of H: the fractional edge cover value of a motif (graph) H is the solution to the following minimization problem. Minimize $\sum_{e \in E} f(e)$ under the constraint that for every $v \in H$, $\sum_{e \ni v} f(e) \geq 1$. In [3], the decomposition is computed by first computing an optimal fractional cover. However, as there exists a mapping between fractional edge covers to decompositions which preservers their value, we choose to define $\rho(H)$ according to the minimal valid decomposition value.

[3] Throughout the paper, unless stated otherwise, the query complexity of the mentioned sublinear-time algorithms is the same as the minimum between their running time and $\min\{n + m, m \log n\}$. This is true since any algorithm can simply query the entire graph and continue computation locally. Querying the entire graph can either be performed by querying the neighbors of all vertices (which takes $O(n+m)$ queries), or by performing $m \log n$ uniform edge samples, which, with high probability, return all edges in the graph (note that we do not care about isolated vertices, as we assume the motif H is connected). Hence, we focus our attention on the running time complexity.

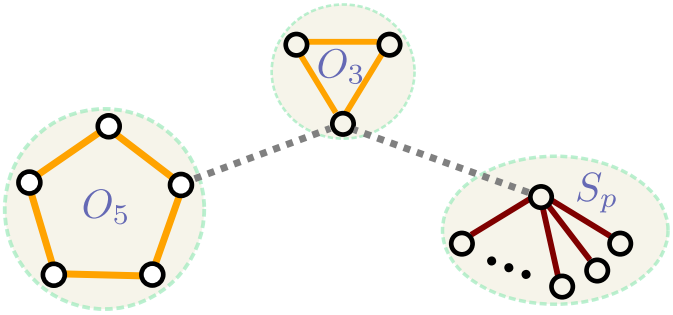

Figure 1 An example of an optimal decomposition of a motif H into odd cycles and stars. The orange edges have weight 1/2, the red edges have weight 1, and the dotted edges have zero weight.

sampling a copy of H, where $\bar{\mathtt{h}}$ is the number of copies of H in G, and m is the number of oriented edges[4] in G. The algorithm in [3] for the $(1 \pm \epsilon)$-approximation task has the same complexity up to $\text{poly}(\epsilon, |H|, \log n)$ factors, where n is the number of vertices in G.

1.1 Our results

We present improved upper and lower bounds for the tasks of estimating and sampling any arbitrary motif in a graph G in sublinear time (with respect to the size of G). First, we give a new, essentially optimal, star-sampler for graphs. We also show that with few modifications, the star-sampler can be adapted to an optimal ℓ_p sampler, which might be of independent interest. Based on this sampler, as well as an improved sampling approach, we present our main algorithm for sampling a uniformly distributed copy of any given motif H in a graph G. Our algorithm's complexity is parameterized by what we refer to as the *decomposition-cost* of H in G, denoted $\textsc{decomp-cost}(G, H, D^*(H))$. We further show that our motif sampling algorithm can be used to obtain a $(1 \pm \epsilon)$-estimate of the motif at question (with an overhead of an $O(1/\epsilon^2)$ factor). As we shall see, our result is always at least as good as previous algorithms for these problems (up to a $\log n \log\log n$ term), and greatly improves upon them for various interesting graph classes, such as random graphs and bounded arboricity graphs.

We then continue to prove that for any motif whose optimal decomposition contains at least one odd cycle, this bound is *decomposition-optimal*: we show that for every decomposition D that contains at least one odd cycle, there exists a motif H_D (with optimal decomposition D) and a family of graphs $\mathcal{G}$ so that in order to sample a uniformly distributed copy of H (or to approximate $\bar{\mathtt{h}}$) in a uniformly chosen graph in $\mathcal{G}$, the number of required queries is $\Omega(\min\{\textsc{decomp-cost}(G, H, D^*(H)), m\})$ in expectation.

We start by describing the upper bound.

1.1.1 Optimal star/ℓ_p-sampler

Our first contribution is an improved algorithm, Sample-a-Star, for sampling a (single) star uniformly at random, and its variant for sampling vertices according to the p^{th} moment. For a vertex v, we let $\bar{\mathtt{s}}_p(v) = \binom{d(v)}{p}$, if $d(v) \geq p$, and otherwise, $\bar{\mathtt{s}}_p(v) = 0$. We let $\bar{\mathtt{s}}_p = \sum_{v \in V} \bar{\mathtt{s}}_p(v)$ denote the number of p-stars in the graph. We will also be interested in the closely related value of the p^{th} moment of the degree distribution, $\bar{\mu}_p = \sum_{v \in V} d(v)^p$.

[4] Throughout the paper we think of every edge $\{u, v\}$ as two oriented edges (u, v) and (v, u), and let m denote the number of oriented edges.

▶ **Theorem 1.** *There exists a procedure,* Sample-a-Star*, that given query access to a graph G, and a constant factor estimates of $\bar{\mathsf{s}}_p$, returns a uniformly distributed p-star in G. The expected query complexity and running time of the procedure are $O\left(\min\left\{\frac{m\cdot n^{p-1}}{\bar{\mathsf{s}}_p}, \frac{m}{\bar{\mathsf{s}}_p^{1/p}}\right\}\right)$ where $\bar{\mathsf{s}}_p$ denotes the number of p-stars in G.*

We note that a constant factor estimate of $\bar{\mathsf{s}}_p$ can be obtained by invoking one of the algorithms in [17, 2], in expected query complexity $\tilde{O}\left(\min\left\{\frac{m\cdot n^{p-1}}{\bar{\mathsf{s}}_p}, \frac{m}{\bar{\mathsf{s}}_p^{1/p}}\right\}\right)$. Therefore, if such an estimate is not known in advance, then it could be computed, with probability at least $2/3$, by only incurring a $\log n$ factor to the expected time complexity.

We will also show a variant of Sample-a-Star, denoted Sublinear-ℓ_p-Sampler, that gives an optimal ℓ_p-sampler for any integer $p \geq 2$ in sublinear time. That is, Sublinear-ℓ_p-Sampler allows to sample according to the p^{th} moment of the degree distribution, so that every vertex $v \in V$ is returned by it with probability $d(v)^p/\bar{\mu}_p$. The question of sampling according to the p^{th} moment for various values of p has been studied extensively in the streaming model where ℓ_p samplers have found numerous applications, see, e.g., the recent survey by Cormode and Hossein [11] and the references therein. Therefore we hope it could find applications in the sublinear-time setting that go beyond subgraph sampling.

▶ **Theorem 2.** *There exists an algorithm,* Sublinear-ℓ_p-Sampler*, that returns a vertex $v \in V$, so that each $v \in V$ is returned with probability $d(v)^p/\bar{\mu}_p$. The expected running time of the algorithm is $O\left(\min\left\{\frac{m\cdot n^{p-1}}{\bar{\mu}_p}, \frac{m}{\bar{\mu}_p^{1/p}}\right\}\right)$.*

Observe that for every value of p, $\bar{\mathsf{s}}_p < \bar{\mu}_p$. Furthermore, Since m and $\bar{\mu}_p^{1/p}$ are simply the ℓ_1 and ℓ_p norms of the degree distribution of G, it holds that $\bar{\mu}_p^{1/p}$ is smaller than m, and could be as small as $m/n^{1-1/p}$. Therefore, $\bar{\mu}_p^{1/p} < m \Leftrightarrow \mu_p^{p-1/p} < m^{p-1}$. and it follows that

$$m \cdot \min\left\{n^{p-1}, \bar{\mathsf{s}}^{(p-1)/p}\right\} \leq m \cdot \bar{\mathsf{s}}_p^{(p-1)/p} < m \cdot \bar{\mu}_p^{(p-1)/p} \leq m \cdot m^{p-1} = m^p. \tag{1}$$

Hence, not accounting for the $O(\log n \log\log n)$ term, the expected complexity $\tilde{O}(m \cdot \min\{n^{p-1}, \bar{\mathsf{s}}_p^{(p-1)/p}\}/\bar{\mathsf{s}}_p)$ of Sample-a-Star strictly improves upon the $O(m^p/\bar{\mathsf{s}}_p)$ expected complexity of the star-sampling algorithm by [23]. Accounting for that term, our algorithm is preferable when either $d_{\mathsf{avg}} = \omega(\log n \log\log n)$ or $m/\bar{\mathsf{s}}_p^{1/p} = \omega(\log n)$.

Furthermore, the complexity of Sample-a-Star matches the complexities of the star approximation algorithms by [26, 2], thus proving that uniformly sampling and approximately counting stars in the augmented model have essentially the same complexity. Finally, the construction of the lower bound for the estimation variant by [26] proves that Sample-a-Star and Sublinear-ℓ_p-Sampler are essentially optimal.

1.1.2 An algorithm for sampling and estimating arbitrary motifs

Given the above star sampler, we continue to describe our main contribution: an algorithm, Sample-H, that for any graph G and given motif H, outputs a uniformly distributed copy of H in G.

To sample a copy of H we first sample copies of all basic components in its decomposition $D^*(H)$, and then check if they can be extended to a copy of H in G. Therefore, it will be useful to define the costs of these sampling operations.

▶ **Notation 3** (Basic components, counts and costs). *Let H be a motif, and let $D^*(H) = \{O_{k_1}, \ldots, O_{k_q}, S_{p_1}, \ldots, S_{p_\ell}\}$ be an optimal decomposition of H. We refer to the odd cycles and stars in $D^*(H)$ as the* basic components *of the decomposition (or sometimes, abusing notation, of H). We use the notation $\{C_i\}_{i\in[r]}$, to denote the set of all components in $D^*(H)$, $\{C_i\}_{i\in[r]} = D^*(H)$, where $r = q + \ell$.*

For every basic component C_i in $D^*(H) = \{C_i\}_{i\in[r]}$, we denote the number of copies of C_i in G as $\bar{\mathsf{c}}_i$ and refer to it as the count *of C_i. Similarly, $\bar{\mathsf{o}}_k$ and $\bar{\mathsf{s}}_p$ denote the number of copies of length k odd cycles and p-stars in G. respectively.*

We also define the sampling cost *(or just* cost *in short) of C_i to be:*

$$cost(C_i) = \begin{cases} m^{k/2}/\bar{\mathsf{o}}_k & C_i = O_k \\ \min\left\{\frac{m\cdot n^{p-1}}{\bar{\mathsf{s}}_p}, \frac{m}{\bar{\mathsf{s}}_p^{1/p}}\right\} & C_i = S_p \end{cases}.$$

Observe that indeed, by Theorem 1, sampling a single p-star in G takes $cost(S_p) = \min\left\{\frac{m\cdot n^{p-1}}{\bar{\mathsf{s}}_p}, \frac{m}{\bar{\mathsf{s}}_p^{1/p}}\right\}$ queries in expectation, and by [23, Lemma 3.1], sampling a single O_k odd cycle takes $cost(O_k) = m^{k/2}/\bar{\mathsf{o}}_k$ queries in expectation.

▶ **Notation 4** (Decomposition-cost). *For a motif H, an optimal decomposition $D^*(H)$ of H, and a graph G, the* decomposition cost *of H in G, denoted* DECOMP-COST$(G, H, D^*(H))$ *is*

$$\textsc{decomp-cost}(G, H, D^*(H)) = \max_{i\in[r]}\{cost(C_i)\} \cdot \frac{\prod \bar{\mathsf{c}}_i}{\bar{\mathsf{h}}}.$$

Note that the motif H determines the counts of $\bar{\mathsf{h}}$ and its decomposition $D^*(H)$ determines what are the basic component counts in G that are relevant to the sampling cost.

▶ **Theorem 5.** *Let G be a graph over n vertices and m edges, and let H be a motif such that $D^*(H) = \{O_{k_1}, \ldots, O_{k_q}, S_{p_1}, \ldots, S_{p_\ell}\} = \{C_i\}_{i\in[r]}$. There exists an algorithm,* Sample-H*, that returns a copy of H in G. With probability at least $1 - 1/\operatorname{poly}(n)$, the returned copy is uniformly distributed in G. The expected query complexity of the algorithm is*

$$O\left(\min\left\{\textsc{decomp-cost}(G, H, D^*(H)), m\right\}\right) \cdot \log n \log\log n.$$

In the full version of this paper ([8]), we prove that with slight modifications to the sampling algorithm we can obtain a $(1 \pm \epsilon)$-approximation algorithm for $\bar{\mathsf{h}}$, with the same expected query complexity and running time up to a multiplicative factor of $O(1/\epsilon^2)$.

Comparison to previous bounds. We would like to compare our algorithm's expected complexity stated in Theorem 5, to the expected complexity $O\left(\frac{m^{\rho(H)}}{\bar{\mathsf{h}}}\right)$ of the counting and sampling algorithms by [3] and [23], respectively, where recall that for an optimal decomposition $D^*(H) = \{O_{k_1}, \ldots, O_{k_q}, S_{p_1}, \ldots, S_{p_\ell}\}$ of H, $\rho(H) = \sum_{i\in[q[} k_i/2 + \sum_{i\in[\ell]} p_i$.

Recalling Equation 1, and plugging in the costs of the basic components and the decomposition cost, defined in Notations 3 and 4, respectively, we get that for any graph G and motif H,

$$\begin{aligned}\textsc{decomp-cost}(G, H, D^*(H)) &= \max_{i\in[r]}\{cost(C_i)\} \cdot \frac{\prod \bar{\mathsf{c}}_i}{\bar{\mathsf{h}}} \\ &= \max_{i\in[r]}\{cost(C_i)\} \cdot \frac{\prod_{i\in[q]} \bar{\mathsf{o}}_{k_i} \cdot \prod_{i\in[\ell]} \bar{\mathsf{s}}_{p_i}}{\bar{\mathsf{h}}}\end{aligned}$$

$$\leq \frac{\prod_{i\in[q]} m^{k_i/2} \cdot \prod_{i\in[\ell]} m \cdot (\min\{n^{p_i-1}, \bar{\mathtt{s}}_{p_i}^{(p_i-1)/p_i}\})}{\bar{\mathtt{h}}}$$
$$< \frac{\prod_{i\in[q]} m^{k_i/2} \cdot \prod_{i\in[\ell]} m^p}{\bar{\mathtt{h}}} = \frac{m^{\rho(H)}}{\bar{\mathtt{h}}}.$$

Therefore, as long as $D^*(H)$ contains at least one star, and not accounting for the term $O(\log n \log\log n)$, our algorithm is preferable to the previous one, as we save a factor of at least d_{avg}^{p-1} for each p-star in $D^*(H)$.

Moreover, the complexity of our sampling algorithm is parameterized by the *actual* counts of the basic components $O_{k_1}, \ldots, O_{k_q}, S_{p_1}, \ldots, S_{p_\ell}$ of the graph G at hand, rather than by the maximal possible counts of these components, respectively $m^{k_1/2}, \ldots m^{k_q/2}, m^{p_1}, \ldots m^{p_\ell}$, as is in previous algorithms. For example, if the max component cost is due to the odd cycle of length k_1, we get

$$O^*\left(\frac{m^{k_1/2} \cdot \bar{\mathtt{o}}_{k_2} \cdot \ldots \cdot \bar{\mathtt{o}}_{k_q} \cdot \bar{\mathtt{s}}_{p_1} \cdot \ldots \cdot \bar{\mathtt{s}}_{p_\ell}}{\bar{\mathtt{h}}}\right) \text{ vs. } O^*\left(\frac{m^{k_1/2} \cdot m^{k_2/2} \ldots \cdot m^{k_q/2} \cdot m^{p_1} \cdot \ldots \cdot m^{p_\ell}}{\bar{\mathtt{h}}}\right)$$

of the previous algorithms. Importantly, this parameterization arises *only* in the analysis, while the algorithm itself is very simple, and does not depend on prior knowledge of the actual values of these counts.

Improved results for various graph classes. Our parameterization immediately implies improved results in various interesting graph classes. For example, for sparse Erdős-Rényi random graphs $\mathcal{G}(n, d/n)$, the expected count of k-odd cycles is $\Theta(d^k)$, and of p-stars is $\Theta(n \cdot d^p)$. Hence, if we consider for example a motif H that is composed of a triangle connected to a 5-petals star, our algorithm has expected complexity $O^*\left(\frac{m^{2.5} \cdot d^4}{\bar{\mathtt{h}}}\right)$, while the algorithms in [3, 23] have expected complexity $O(\frac{m^{6.5}}{\bar{\mathtt{h}}})$. In another example, for graphs of bounded arboricity[5] α, the number of k-odd cycles is upper bounded[6] by $\alpha \cdot m^{(k-1)/2}$. Therefore, in the case that G has, e.g., constant arboricity, we save a multiplicative factor of $\sqrt{m}^q$ or $\sqrt{m}^{q-1}$, depending on whether the max cost component is due to a star or an odd cycle, respectively (recall that q is the number of odd cycles in the decomposition).

1.1.3 Lower bound for estimating and sampling general motifs

In the full version, we prove the following lower bound, which states that for every decomposition D that contains at least one odd cycle component and every realizable value of DECOMP-COST, there exists a motif H_D such that D is an optimal decomposition of H_D, and for which our upper bound is optimal.

▶ **Theorem 6.** *For any decomposition D that contains at least one odd cycle, and for every n and m and realizable value DC of DECOMP-COST, there exists a motif H_D, with optimal decomposition D, and a family of graphs $\mathcal{G}$ over n vertices and m edges, for which the following holds. For every $G \in \mathcal{G}$, DECOMP-COST$(G, H_D, D) =$ DC, and the expected query complexity of sampling (whp) a uniformly distributed copy of H_D in a uniformly chosen $G \in G$ is $\Omega(DC)$.*

[5] The arboricity of a graph G is the minimal number of forests required to cover the edge set of G.

[6] In a graph G with arboricity α there exists an acyclic ordering of the graph's vertices, such that each vertex has $O(\alpha)$ vertices exceeding it in the order. We can attribute each k-cycles in the graph to its first vertex in that ordering. It then holds that each vertex has at most $(d^+(v))^2 \cdot m^{(k-3)/2}$ attributed cycles, and it follows that $\bar{\mathtt{o}}_k \leq \alpha \cdot m^{(k-1)/2}$, where $d^+(v)$ is the number of neighbors of v that exceed it in the aforementioned ordering.

Prior to this work, the only known lower bounds for the tasks of uniformly sampling or approximately counting motifs H that were either a clique [19], a single odd cycle [3], or a single star [26, 2, 19]. The above theorem provides the first lower bounds for motifs with non-trivial decompositions. Furthermore, even though our bounds are only *decomposition-optimal* (that is, they do not hold for *any* motif H), each decomposition D corresponds to at least one motif $H_{\mathcal{D}}$ (generally, there are multiple valid ones), for which our bounds are tight.

In order to prove Theorem 6, we actually prove a stronger theorem, which relies on a technical notion of *good counts*, formally stated in Definition 17 in the full version.

▶ **Theorem 7.** *For any decomposition* $D = \{O_{k_1}, \ldots, O_{k_q}, S_{p_1}, \ldots, S_{p_\ell}\} = \{C_i\}_{i \in r}$ *that contains at least one odd cycle component, for every* n, m, $\bar{\mathtt{h}}$ *and a set of good counts,* $\{\bar{\mathtt{c}}_i\}_{i\in[r]} = \{\bar{\mathtt{o}}_{k_1}, \ldots, \bar{\mathtt{o}}_{k_q}, \bar{\mathtt{s}}_{p_1}, \ldots, \bar{\mathtt{s}}_{p_\ell}\}$*, as defined in Definition 17 of the full version, the following holds. There exists a motif* H_D*, with an optimal decomposition* D*, and a family of graphs* $\mathcal{G}$ *over* n *vertices and* m *edges, as follows. For every* $G \in \mathcal{G}$*, the basic components counts are as specified by* $\{\bar{\mathtt{c}}_i\}_{i\in[r]}$*, the number of copies of* H_D *is* $\bar{\mathtt{h}}$*, and the expected query complexity of sampling (whp) a uniformly distributed copy of* H_D *in a uniformly chosen* $G \in \mathcal{G}$ *is*

$$\Omega\left(\min\left\{\max_{i\in[r]}\{cost(C_i)\}\cdot\frac{\prod_i \bar{\mathtt{c}}_i}{\bar{\mathtt{h}}}, m\right\}\right).$$

In the full version, we first prove that Theorem 6 follows from Theorem 7. Theorem 7 is essentially a substantial refinement of Theorem 6, in the following sense. Not only that for any decomposition cost we can match the lower bound (as stated in Theorem 6), but we can match it for a large variety of *specific* setting of the basic counts (as long as they are good, as stated in Theorem 7). While Theorem 7 does not state that the lower bound holds for *any* setting of the counts $\{\bar{\mathtt{c}}_i\}_{i\in[r]}$, as we discuss in the full version, some of the constraints on these counts are unavoidable. It remains an open question whether this set of constraints can be weakened, or perhaps more interestingly, whether, given that a set of constraints that is *not good*, can a better upper bound be devised.

1.2 Organization of the paper

We give some preliminaries in Section 2. The discussion on additional related works on sublinear motif counting and sampling is deferred to Appendix A. In Section 3 we give a high level overview of our techniques. We present our algorithms for uniformly sampling stars and arbitrary motifs H in Section 4. Due to page limitation, the full details of the ℓ_p-sampler, approximation algorithm, as well as the decomposition-optimal lower bounds are deferred to the full version of this paper [8].

2 Preliminaries and Notation

Let $G = (V, E)$ be a simple undirected graph. We let n denote the number of vertices in the graph. We think of every edge $\{u, v\}$ in the graph as two *oriented* edges (u, v) and (v, u), and slightly abuse notation to let m denote the number of oriented edges, so that $m = \sum_{v\in V} d(v) = 2|E|$, and $d_{\mathsf{avg}} = m/n$. Unless explicitly stated otherwise, when we say "edge" we mean an oriented edge. We let $d(v)$ denote the degree of a given vertex. We let $[r]$ denote the set of integers 1 through r.

The augmented query model. We consider the augmented query model which allows for the following queries. (1) A degree query, $deg(v)$, returns the degree of v, $d(v)$; (2) An i^{th} neighbor query, $Nbr(v,i)$ returns the i^{th} neighbor of v if $i \leq d(v)$, and otherwise returns FAIL; (3) A pair query, $pair(u,v)$, returns whether $(u,v) \in E$; and (4) Uniform edge query returns a uniformly distributed (oriented) edge in E.

A decomposition into odd cycles and stars. Given a motif H, the result in [3] is parameterized by the *fractional edge cover number* $\rho(H)$. The fractional edge cover number is the optimal solution to the *linear programming relaxation* of the integer linear program (ILP) for the minimum edge cover of H: The ILP allows each edge to take values in $\{0,1\}$, under the constraint that the sum of edge values incident to any vertex v is at least 1. The LP relaxation allows values in $[0,1]$ instead, and $\rho(H)$ is the minimum possible sum of all the (fractional) values. In [3], the authors strengthen an existing result by Atserias, Grohe nd Marx [4], in order to prove that there always exists an optimal solution as follows. All of the weight (i.e., non zero edges) is supported on (the edges of) vertex-disjoint odd cycles and stars, where each odd cycle edge has weight $1/2$, and each star edge has weight 1. Consequently, the corresponding optimal solution of the LP for a given graph H is equivalent to a decomposition of H into a collection of vertex-disjoint odd cycles and stars, denoted $D^*(H) = \{O_{k_1}, ..., O_{k_q}, S_{p_1}, ..., S_{p_\ell}\}$. See Figure 1 for an illustration.

Generally, the motif we aim to sample (or approximate its counts) will be denoted by H, and the corresponding decomposition will be $\mathcal{D}(H) = \{O_{k_1}, ..., O_{k_q}, S_{p_1}, ..., S_{p_\ell}\} = \{C_i\}_{i \in r}$ for $r = q + \ell$. We use a convention of using O_{k_i} to refer to the i^{th} decomposition component which is an odd cycle of size k_i, and S_{p_i} to refer to the i^{th} star component, which is a star with p_i petals. We use $\bar{\mathtt{o}}_k$ and $\bar{\mathtt{s}}_p$ denote the number of k-cycles and p-stars in G respectively, and we use $\bar{\mathtt{h}}$ to denote the number of copies of H in G.

Next, we formally define the fractional edge cover of a graph (or motif), and the resulting decomposition. We note that in this paper we will be interested in the decomposition of the motif H, and not the graph G.

▶ **Definition 8** (Fractional edge cover). *A fractional edge cover of a graph is a function $f : E \to \mathbb{R}_{\geq 0}$ such that for every $v \in V$, $\sum_{e \ni v} f(e) \geq 1$. We say that the cost of a given edge cover f is $\sum_{e \in E} f(e)$. For any graph (motif) H, its fractional edge cover value is the minimum cost over all of its fractional edge covers, and we denote this value by $\rho(H)$. An* optimal *edge-cover of H is any edge cover of H with cost $\rho(H)$.*

▶ **Lemma 9** (Lemma 4 in [3]). *Any graph (motif) H admits an optimal fractional edge cover x^*, whose support, denoted $SUPP(x^*)$, is a collection of vertex-disjoint odd cycles and stars, such that:*

- *for every odd cycle $C \in SUPP(x^*)$, for every $e \in C$, $x^*(e) = 1/2$.*
- *for every $e \in SUPP(x^*)$ that does not belong to an odd cycle, $x^*(e) = 1$.*

▶ **Definition 10** (Decomposition into odd-cycles and stars). *Given an optimal fractional edge-cover x^* as in Lemma 9, let $\{O_{k_1}, ..., O_{k_q}\}$ be the odd-cycles in the support of x^*, and let $\{S_{p_1}, ..., S_{p_\ell}\}$ be the stars. We refer to $D^*(H) := \{O_{k_1}, ..., O_{k_q}, S_{p_1}, ..., S_{p_\ell}\}$ as an (optimal)* decomposition of H.

Given a graph (motif) H, its fractional edge cover value and an optimal decomposition can be computed efficiently:

▶ **Theorem 11** (Lemma 4 and Section 3 in [3]). *For any graph H, its fractional edge cover value $\rho(H)$ and an optimal decomposition $D^*(H)$ can be computed in polynomial time in $|H|$.*

3 Overview of Our Results and Techniques

We start with describing the ideas behind our upper bound result.

3.1 An algorithm for sampling arbitrary motifs

We take the same approach as that of [23], of sampling towards estimating, but improve on the query complexity of their bound using two ingredients. The first is an improved star sampler, and the second is an improved sampling approach.

Improved star sampler. The algorithm of [23] tries to sample p-stars by sampling p edges uniformly at random, and checking if they form a star (by simply checking if all p edges agree on their first endpoint). Hence, each p-star is sampled with probability $1/m^p$. Our first observation is that it is more efficient to sample a *single* edge (u, v) and then sample $p-1$ neighbors of v uniformly at random, by drawing $(p-1)$ indices $i_1, \ldots, i_p$ in $[d(v)]$ uniformly at random, and performing neighbor queries (v, i_j) for every $j \in [p-1]$. However, this sampling procedure introduces biasing towards stars that are incident to lower degree endpoints. If we were also given an upper bound d_{ub} on the maximal degree in the graph, i.e., a value d_{ub} such that $d_{max} \leq d_{ub}$, where d_{max} is the maximum degree in G, then we could overcome the above biasing, by "unifying" all the degrees in the graph to d_{ub}. Specifically, this unification of degrees is achieved by querying the i^{th} neighbor of a vertex, where i is chosen uniformly at random in $[d_{ub}]$, rather than in $[d(v)]$.[7] By repeating this process $p-1$ times, we get that each specific copy of a p-star is sampled with equal probability $\frac{1}{m \cdot (d_{ub})^{p-1}}$. Observe that this is always preferable to $1/m^p$, i.e. $\frac{1}{m \cdot (d_{ub})^{p-1}} > \frac{1}{m^p}$, since for every graph G, $d_{ub} < m$. While we are not given such a bound on the maximal degree, letting $\bar{\mathbf{s}}_p$ denote the number of p-stars in G, it always holds that $d_{max} \leq \min\{n, \bar{\mathbf{s}}_p^{1/p}\}$ (since every vertex with degree $d > p$ contributes d^p to $\bar{\mathbf{s}}_p$). Hence, we can use the existing algorithms for star approximations by [26, 2, 17] in order to first get an estimate $\hat{\mathbf{s}}_p$ of $\bar{\mathbf{s}}_p$, and then use this estimate to get an upper bound d_{ub} on d_{max} by setting $d_{ub} = \min\{n, \hat{\mathbf{s}}_p^{1/p}\}$.

An improved sampling approach. In order to describe the second ingredient for improving over the bounds of [23], we first recall their algorithm. In the first step, their algorithm simultaneously attempts to sample a copy of each odd cycle and star in the decomposition of H. Then if all individual sampling attempt succeed, the algorithm proceeds to check if the sampled copies are connected in G in a way that is consistent with the non-decomposition edges of H. However, it is easy to see that this approach is wasteful. Even if all but one of the simultaneous sampling attempts of the first step succeed, the algorithm starts over. For example, if $D^*(H)$ consists of a star and a triangle, then in the first step their algorithm attempts to sample simultaneously a star and a triangle, and in the case that, say, a triangle is sampled but the star sampling attempt fails, then the sampled triangle is discarded, and the algorithm goes back to the beginning of the first step.

To remedy this, in the first step our algorithm invokes the star- and odd-cycle samplers for every basic component in $D^*(H)$, until all samplers return an *actual* copy of of the requested component. This ensures that we proceed to the next step of verifying H only once we have actual copies of all the basic components. We then continue to check if these copies can be

[7] This is effectively equivalent to rejection sampling where first v is "kept" with probability $d(v)/d_{ub}$, and then a neighbor of v is sampled uniformly at random.

extended to a copy of H in G, as before. While this is a subtle change, it is exactly what allows us to replace the dependency in the maximum number of potential copies of the basic components, to a dependency in the actual number of copies in G.

We note that for motifs H whose decomposition has repeating smaller sub-motifs, our sampling approach can be used recursively, which can be more efficient. That is, instead of decomposing H to its most basic components, stars and odd-cycles, we can consider decomposing it to collections of more complex components. For example, if H has such a collection $H_1 \subset H$ that is repeated more than once, then it is more beneficial to first try and sample all of the copies of H_1 (as well as the other components of H) and only then try to extend these copies to H. The sampling of the H_1 copies can then be performed by a recursive call to the motif sampler. It can be shown that for any repeated motif H_1 in the decomposition of H, applying the recursive sampling process results in an improved upper bound.

From sampling to estimating

In order to obtain a $(1 \pm \epsilon)$-estimate of $\bar{\mathtt{h}}$, we can use the sampling algorithm as follows. Consider a single sampling attempt in which we first sample all basic components of $D^*(H)$ (at some cost Q), and then preform all pair queries between the components to check if the sampled components induce a copy of H (at cost $O(|H|^2)$). By the above description such an attempt succeeds with probability that depends on the counts of the basic components of $D^*(H)$ and on the count $\bar{\mathtt{h}}$. Hence we can think of the success probability of each attempt as a coin toss with bias p, where p depends only on the counts of the components and $\bar{\mathtt{h}}$. By standard concentration bounds, using $\Theta(1/(p\epsilon^2))$ sampling attempts, we can compute a $(1 \pm \epsilon)$-estimate $\hat{p}$ of p. Since we can also get $(1 \pm \epsilon)$-multiplicative estimates of the counts of each basic component without asymptotically increasing the running time, we can deduce from $\hat{p}$ a $(1 \pm \Theta(\epsilon))$-estimate of $\bar{\mathtt{h}}$. See the full version for more details.

3.2 Decomposition-optimal lower bounds

Theorem 6 follows from Theorem 7. To prove Theorems 6 and 7, we first explain why given Theorem 7, Theorem 6 follows. We then describe at a high level a family of graphs $\mathcal{G}$ in which sampling copies of a given motif is hard.

At a high level, Theorem 7 states that given (1) a decomposition D and (2) a set of good counts $\{\bar{\mathtt{c}}_i\}_{i\in[r]}$, we can construct (3) a motif H_D (such that D is an optimal decomposition of H_D) and (4) a family of graphs $\mathcal{G}$ such that expected number of queries required to sampling copies of H_D in $\mathcal{G}$ is

$$\max_{i\in[r]}\{cost(C_i)\} \cdot \frac{\prod \bar{\mathtt{c}}_i}{\bar{\mathtt{h}}} \ .$$

Theorem 6 states that given (a) a decomposition D and (b) a (realizable) decomposition cost DC, that there exists (c) a motif H_D and (d) a family of graphs for which the decomposition-cost of G, D and H_D is DC, and sampling copies of H_D in graphs of $\mathcal{G}$ requires $\Omega(\text{DC})$ queries.

To prove that Theorem 6 follows from Theorem 7, we then prove that given (a) and (b), we can specify a set of counts which both satisfies $\text{DC} = \max_{i\in[r]}\{cost(C_i)\} \cdot \frac{\prod \bar{\mathtt{c}}_i}{\bar{\mathtt{h}}}$ and which is good. Since the set of counts is good, we can invoke Theorem 7, and get that there exists a motif H_D and a family of graphs in which it is hard to sample copies of H_D. Therefore, in the rest of the section we focus our attention on the proof of Theorem 7.

Ideas behind the proof of Theorem 7. Given a graph decomposition D, values n, m, $\bar{\mathtt{h}}$ and a set of counts $\bar{\mathtt{c}}_1, \ldots, \bar{\mathtt{c}}_r$ of its basic components, our lower bound proof starts by defining a motif H_D, and a family of graphs $\mathcal{G}$ such that the following holds.

- The optimal decomposition of H_D is D;
- For every $G \in \mathcal{G}$ and $O_{k_i}, S_{p_j} \in D$, their number of copies in G is $\Theta(\bar{\mathtt{o}}_{k_i})$ and $\Theta(\bar{\mathtt{s}}_{p_j})$, respectively;
- The number of copies of H in G is $\Theta(\bar{\mathtt{h}})$
- Sampling a uniformly distributed copy of H_D in a uniformly chosen G in $\mathcal{G}$, requires $\Omega\left(\min\left\{m, \frac{m}{\bar{\mathtt{h}}}\right\}\right)$ queries in expectation.

There are several challenges in proving our lower bound. First, as they are very general and work for any given decomposition D that contains at least one odd cycle, there are many sub cases that need to be dealt with separately, depending on the mixture of components in D. Second, the lower bound term does not only depend on the different counts, but also on the relations between them, which determines the component that maximizes $cost(C_i)$. As mentioned previously, our lower bound only holds for the case that the max cost is due to an odd cycle component. It remains an open question whether a similar lower bound can be proven for the case that the max cost is due to a star, or whether in that case a better algorithm exists. The authors suspect the latter option. Third, as in most previous lower bounds for motif sampling and counting, we prove the hardness of the task by "hiding" a constant fraction of the copies of H_D, so that the existence of these copies depends on a small set of crucial edges. That is, we prove that we can construct the family of graphs $\mathcal{G}$, such that for every $G \in \mathcal{G}$, a specific set of t crucial edges, for some small t that depends on the basic counts and $\bar{\mathtt{h}}$, contributes $\Theta(\bar{\mathtt{h}})$ copies of H_D . We then prove that detecting these edges requires many queries (this is formalized by a reduction from a variant of the Set-Disjointness communication complexity problem, based on the framework of [19]). This approach of constructing many copies of H_D which all depend on small set of crucial edges, leads the construction of the graphs $\mathcal{G}$ to contain very dense components, which in turn causes correlations between the counts of the different components. A significant challenge is therefore to define the motif H_D and the graphs of $\mathcal{G}$ in a way that satisfies all given counts simultaneously.

In each graph G in the hard family $\mathcal{G}$, we have a corresponding "gadget" to each of the components of D. Let k_1 denote (one of) the maximum-cost odd-cycle components. For each odd-cycle component O_{k_i} for $k_i \neq k_1$, we define either a `few-cycles-gadget` or a `cycle-gadget` that induce $\bar{\mathtt{o}}_{k_i}$ odd cycles of length k_i according to the relation between k_i and k_1. For each star component S_{p_j} we define a `star-gadget` that induces $\bar{\mathtt{s}}_{p_j}$ many p_j-stars. The maximum-cost cycle component O_{k_1} has a different gadget, a `CC-gadget`. This gadget is used to hide the set of t crucial edges, and allows us to parameterize the complexity in terms of the cost $cost\{O_{k_1}\}$.

To formally prove the lower bound we make use the framework introduced in [19], which uses reductions from communication complexity problems to motif sampling and counting problems in order to prove hardness results of these latter tasks. This allows us to prove that one cannot, with high probability, witness an edge from the set of t hidden edges, unless $\Omega(m/t)$ queries are performed. This in turn implies that one cannot, with high probability, witness a copy of H_D contributed by these edges. Hence, we obtain a lower of $\Omega(m/t)$ for the task of outputting a uniformly sampling. Setting t appropriately gives the desired bound.

4 Upper Bounds for Sampling Arbitrary Motifs

In this section we present our improved sampling algorithm. Recall that our upper bound improvement has two ingredients, an improved star sampler, and an improved sampling approach. We start with presenting the improved star sampling algorithm.

4.1 An optimal (ℓ_p) star-sampler

Our star sampling procedure assumes that it gets as a parameter a value $\hat{\mathsf{s}}_p$ which is a constant-factor estimate of $\bar{\mathsf{s}}_p$. This value can be obtained by invoking one of the star estimation algorithm of [2, 17].

▶ **Lemma 12** ([2], Theorem 1). *Given query access to a graph G and an approximation parameter ϵ, there exists an algorithm, Moment-Estimator, that returns a value $\hat{\mathsf{s}}_p$, such that with probability at least $2/3$, $\hat{\mathsf{s}}_p \in [\bar{\mathsf{s}}_p, 2\bar{\mathsf{s}}_p]$. The expected query complexity and running time $O\left(\min\left\{m, \min\left\{\frac{m\cdot n^{p-1}}{\bar{\mathsf{s}}_p}, \frac{m}{\bar{\mathsf{s}}_p^{1/p}}\right\}\cdot \log\log n\right\}\right)$.*

Given an estimate $\hat{\mathsf{s}}_p$ on $\bar{\mathsf{s}}_p$, our algorithm sets an upper bound[8] d_{ub} on the maximal degree, $d_{ub} = \min\{n, \hat{\mathsf{s}}_p\}$. It then tries to sample a copy of a p-star as follows. In each sampling attempt it samples a single edge (v_0, v_1), and then performs $p-1$ neighbor queries $nbr(v_0, i_j)$ for $j = 2 \ldots p$, where each i_j is chosen independently and uniformly at random from $[d_{ub}]$. In order to ensure that the sampled neighbors are distinct, and to avoid multiplicity issues, a p-star is returned only if its petals are sampled in ascending order of ids. In every such sampling attempt, each specific p-star is therefore sampled with equal probability $\frac{1}{m\cdot d_{ub}^{p-1}}$. Hence, invoking the above $\frac{m\cdot d_{ub}^{p-1}}{\bar{\mathsf{s}}_p}$ times, in expectation, returns a uniformly distributed copy of a p-star.

Sample-a-Star$(p, n, \hat{\mathsf{s}}_p)$

1. Let $d_{ub} = \min\{n, (c_p\cdot\hat{\mathsf{s}}_p)^{1/p}\}$ for a value c_p as specified in the proof of Theorem 1.
2. While **TRUE**:
   a. Perform a uniform edge query, an denote the returned edge (v_0, v_1).
   b. Choose $p-1$ indices $i_2, \ldots, i_p$ uniformly at random in $[d_{ub}]$ (with replacement).
   c. For every $j \in [2..p]$, query the i_j^{th} neighbor of v_0. Let $v_2, \ldots, v_p$ be the returned vertices, if all queries returned a neighbor. Otherwise break.
   d. If $id(v_2) < id(v_2) < \ldots < id(v_p)$, then **return** $(v_0, v_1, \ldots, v_p)$.

▶ **Theorem 13.** *Assume that $\hat{\mathsf{s}}_p \in [\bar{\mathsf{s}}_p, c\cdot\bar{\mathsf{s}}_p]$ for some small constants c. The procedure* Sample-a-Star$(p, \hat{\mathsf{s}}_p)$ *returns a uniformly distributed p-star in G. The expected query complexity of the procedure is $O\left(\min\left\{\frac{m\cdot n^{p-1}}{\bar{\mathsf{s}}_p}, \frac{m}{\bar{\mathsf{s}}_p^{1/p}}\right\}\right)$.*

Proof. Let c_p denote the minimal value such that for every $k \in [n]$, $c_p \cdot \binom{k}{p} \geq k^p$ (note that $c_p = \Theta(p!)$). Then $\bar{\mathsf{s}}_p = \sum_{v\in V}\binom{d(v)}{p} > \binom{d_{max}}{p} \geq d_{max}^p/c_p$, and by the assumption on $\hat{\mathsf{s}}_p$, $d_{max} < (c_p\cdot\bar{\mathsf{s}}_p)^{1/p} \leq (c_p\cdot\hat{\mathsf{s}}_p)^{1/p}$. It follows by the setting of $d_{ub} = \min\{n, (c_p\cdot\hat{\mathsf{s}}_p)^{1/p}\}$ in Step 1, that $d_{ub} \geq d_{max}$.

[8] Observe that d_{max} is $d_{max} = \max_v d(v)$, while d_{ub} is simply a bound on d_{max}, so that $d_{max} \leq d_{ub}$.

Consider a specific copy $\bar{S}_p = (a_0, a_1, \ldots, a_p)$ of a p-star in G, where a_0 is the star center and a_1 through a_p are its petals in ascending id order. In each iteration of the while loop, the probability that $\bar{S}_p$ is returned is

$$\begin{aligned}\Pr[\bar{S}_p \text{ is returned}] &= \Pr[(a_0, a_1) \text{ is sampled in Step 2a}] \\ &\quad \cdot \Pr[a_2, \ldots, a_p \text{ are sampled in Step 2b}] \\ &= \frac{1}{m} \cdot \frac{1}{d_{ub}^{p-1}} \,. \end{aligned} \tag{2}$$

Note the the last equality crucially depends on $d(v) \leq d_{max} \leq d_{ub}$ for all $v \in V$. (Indeed, if there exists a vertex v with degree $d(v) > d_{ub}$, then some of its incident stars will have zero probability of being sampled.) Hence, each copy is sampled with equal probability, implying that the procedure returns a uniformly distributed copy of a p-star.

We now turn to bound the expected query complexity. It follows from Equation 2 and the setting of d_{ub}, that the success probability of a single invocation of the while loop is $\frac{\bar{\mathsf{s}}_p}{m \cdot d_{ub}^{p-1}}$. Hence, the expected number of invocations is $\frac{m \cdot d_{ub}^{p-1}}{\bar{\mathsf{s}}_p}$. It follows that, for a constant p, the expected number of invocations is

$$O\left(\frac{m \cdot \min\{n, (c_p \cdot \bar{\mathsf{s}}_p)^{1/p}\}^{p-1}}{\bar{\mathsf{s}}_p}\right) = O\left(\min\left\{\frac{m \cdot n^{p-1}}{\bar{\mathsf{s}}_p}, \frac{m}{\bar{\mathsf{s}}_p^{1/p}}\right\}\right).$$

Since the query complexity and running time of a single invocation of the while loop are constant, the above is also a bound on the expected query complexity and running time of the while loop. ◀

In the full version of this paper, we explain how algorithm Sample-a-Star can be slightly modified to produce an ℓ_p-sampler, Sublinear-ℓ_p-Sampler as specified in Theorem 2.

4.2 General motif sampler

Our algorithm for sampling uniform copies of a motif H in a graph G relies on the above star sampler, and the odd cycle sampler of [23].

▶ **Lemma 14** (Lemma 3.3 in [23], restated)**.** *There exists a procedure that, given a parameter k and an estimate $\hat{m} \in [m, 2m]$, samples each specific copy of an odd cycle of length k with probability $1/m^{k/2}$.*

It follows that by repeatedly invoking the procedure above until an odd cycle is returned we can get an odd cycle sampling algorithm.

▶ **Corollary 15.** *There exists a procedure,* Sample-Odd-Cycle*, that, given an estimate $\hat{m} \in [m, 2m]$, returns a uniformly distributed copy of an odd cycle of length k. The expected query complexity is $O\left(\min\left\{m \log n, n + m, \frac{m^{k/2}}{\bar{\mathsf{o}}_k}\right\}\right)$, where $\bar{\mathsf{o}}_k$ denotes the number of odd cycles of length k in G.*

We also use the following algorithm from [25] to obtain an estimate of m.

▶ **Theorem 16** ([25], Theorem 1, restated)**.** *There exists an algorithm that, given query access to a graph G, the number of vertices n, and a parameter ϵ, returns a value $\tilde{m}$, such that with probability at least $2/3$, $\tilde{m} \in [m, (1+\epsilon)m]$. The expected query complexity and running time of the algorithm is $O(n/\sqrt{m}) \cdot (\log\log n/\epsilon^2)$.*

Our motif sampling algorithm invokes the star-sampler and odd-cycles-sampler for each of the star and odd-cycles components in $D^*(H)$, respectively. Once actual copies of all the components are sampled, it checks whether they form a copy of H in G, using $O(|H|^2) = O(1)$ additional pair queries.

Sample-H (H, n)

1. Compute a 2-factor estimate $\hat{m}$ of m by invoking the algorithm of [25] with $\epsilon = 1/2$ for $10 \log n$ times, and letting $\hat{m}$ be the median of the returned values.
2. Compute an optimal decomposition of H, $D^*(H) = \{O_{k_1}, \ldots, O_{k_q}, S_{p_1}, \ldots, S_{p_\ell}\}$.
3. For every S_{p_i} in D, invoke algorithm Moment-Estimator with $\epsilon = 1/2$ and $r = p_i$ for $t = 10 \log(n \cdot \ell)$ times to get t estimates of $\bar{\mathsf{s}}_{p_i}$. Let $\hat{\mathsf{s}}_{p_i}$ be the median value among the t received estimates of each S_{p_i}.
4. While **True:**
   a. For every $i \in [q]$ do:
      i. Invoke Sample-Odd-Cycle$(k_i, \hat{m})$, and let $\bar{O}_i$ be the returned odd cycle.
   b. For every $i \in [\ell]$ do:
      i. Invoke Sample-a-Star$(p_i, n, \hat{\mathsf{s}}_{p_i})$, and let $\bar{S}_j$ be the returned s_j-star.
   c. Perform $O(|H|^2)$ pair queries to verify whether the set of components $\{\bar{O}_1, \ldots, \bar{O}_q, \bar{S}_1, \ldots, \bar{S}_\ell\}$ can be extended to a copy of H in G.
   d. If a copy of H is discovered, then **return** it.
   e. If the number of queries performed exceeds $n + \hat{m}$, then query all edges of the graph[a] and output a uniformly distributed copy of H.

[a] by either performing $n + 2m$ degree and neighbor queries, or $10m \log n$ uniform edge queries

We are now ready to prove our main upper bound theorem, which we recall here.

▶ **Theorem 17.** *Let G be a graph over n vertices and m edges, and let H be a motif such that $D^*(H) = \{O_{k_1}, \ldots, O_{k_q}, S_{p_1}, \ldots, S_{p_\ell}\} = \{C_i\}_{i \in [r]}$. There exists an algorithm,* Sample-H*, that returns a copy of H in G. With probability at least $1 - 1/\operatorname{poly}(n)$, the returned copy is uniformly distributed in G. The expected query complexity of the algorithm is*

$$O\left(\min\left\{\textsc{decomp-cost}(G, H, D^*(H)), m\right\}\right) \cdot \log n \log\log n.$$

Proof. By Theorem 16, when invoked with a value $\epsilon = 1/2$, the edge estimation algorithm of [25] returns a value $\tilde{m}$ such that, with probability at least $2/3$, $\tilde{m} \in [m, 1.5m]$. Hence, with probability at least $1 - 1/3n^2$, the median value $\hat{m}$ of the $10 \log n$ invocations is such that $\hat{m} \in [m, 1.5m]$. We henceforth condition on this event.

We next prove that with probability at least $1 - 1/3n^2$, all the computed $\hat{\mathsf{s}}_{p_i}$ values are good estimates of $\bar{\mathsf{s}}_{p_i}$. By Lemma 12, for a fixed p_i, with probability at least $2/3$, the value returned from Moment-Estimator is in $[\hat{\mathsf{s}}_{p_i}, 1.5 \cdot \hat{\mathsf{s}}_{p_i}]$. Therefore, the probability that the median value of the $t = 10 \log(n\ell)$ invocations in Step 3 is outside this range is at most $1/(3\ell n^2)$. Hence, taking a union bound over all $i \in [\ell]$, with probability at least $1 - 1/3n^2$, for every $i \in [\ell]$, $\hat{\mathsf{s}}_{p_i} \in [\bar{\mathsf{s}}_p, 1.5 \cdot \bar{\mathsf{s}}_p]$. We henceforth condition on this event as well.

Fix a copy H' of H in G, and let $O'_1, \ldots, O'_q, S'_1, \ldots, S'_\ell$ be its cycles and stars, corresponding to those of $D^*(H)$. By Corollary 15, for each O'_i, its probability of being returned in Step 4(a)i is $1/\bar{\mathsf{o}}_{k_i}$. Similarly, by Lemma 1, for each S'_i, its probability of being returned in Step 4(b)i is $1/\bar{\mathsf{s}}_{p_i}$. Therefore, in the case that the number of queries does not exceed $\hat{m}$, in every iteration

of the loop, each specific copy of H is returned with equal probability $\frac{1}{\Pi_{i=1}^{q}\bar{\mathsf{o}}_{k_i}\cdot\Pi_{i=1}^{\ell}\bar{\mathsf{s}}_{p_i}}$. [9] Hence, once a copy of H is returned, it is uniformly distributed in G. In the case that the number of queries exceeds $\hat{m}$, the algorithm either performs $n+2m$ queries to query all the neighbors of all vertices, or $10m\log n$ queries, in order to discover all edges with high probability. In the former case, the entire graph G is known. In the latter case, by the coupon collector analysis, the probability that all edges are known at the end of the process is at least $1-1/3n^2$. Hence, with probability at least $1-1/3n^2$, at the end of this process, a uniformly distributed copy of H is returned.

It remains to bound the query complexity. By Lemma 12, Step 3 takes $\sum_{p_i} t\cdot\min\left\{\frac{m\cdot n^{p_i-1}}{\bar{\mathsf{s}}_{p_i}},\frac{m}{\bar{\mathsf{s}}_{p_i}^{1/p_i}}\right\}\cdot\log n\log\log n$ queries in expectation. By the above discussion, it holds that the expected number of invocations of the while loop is $\frac{\Pi_{i=1}^{q}\bar{\mathsf{o}}_{k_i}\cdot\Pi_{i=1}^{\ell}\bar{\mathsf{s}}_{p_i}}{\bar{\mathsf{h}}}$. Furthermore, by Lemma 1, the expected query complexity of sampling each S_{p_i} is $\min\left\{\frac{m\cdot n^{p_i-1}}{\bar{\mathsf{s}}_{p_i}},\frac{m}{\bar{\mathsf{s}}_{p_i}^{1/p_i}}\right\}$. By Lemma 15, the expected running time of each invocation of the k_i-cycle sampler is $O\left(\frac{m^{k_i/2}}{\bar{\mathsf{o}}_{k_i}}\right)$. The complexity of Step 4c is $O(|H|^2)=O(1)$ queries, and is subsumed by the complexity of the other steps. Hence, the expected cost of each invocation of the while loop is

$$\max_{i\in[q]}\left\{\frac{m^{k_i/2}}{\bar{\mathsf{o}}_{k_i}}\right\}+\max_{i\in[\ell]}\left\{\min\left\{\frac{m}{\bar{\mathsf{s}}_{p_i}^{1/p_i}},\frac{m\cdot n^{p_i-1}}{\bar{\mathsf{s}}_{p_i}}\right\}\right\}=\max_{i\in[q]}\left\{\frac{m^{k_i/2}}{\bar{\mathsf{o}}_{k_i}}\right\}+\min\left\{\frac{m}{\bar{\mathsf{s}}_{p}^{1/p}},\frac{m\cdot n^{p-1}}{\bar{\mathsf{s}}_{p}}\right\},$$

where the equality holds since the maximum of the second term is always achieved by the largest star in the decomposition, S_p. Also, due to Step 4e and the assumption on $\hat{m}$, the query complexity of algorithm is always bounded by $O(\min\{m\log n, n+m\})$. Therefore, the overall expected query complexity is the minimum between $O(\min\{m\log n, n+m\})$ and

$$\begin{aligned}
&O\left(\left(\max_{i\in[q]}\left\{\frac{m^{k_i/2}}{\bar{\mathsf{o}}_{k_i}}\right\}+\min\left\{\frac{m\cdot n^{p-1}}{\bar{\mathsf{s}}_{p}},\frac{m}{\bar{\mathsf{s}}_{p}^{1/p}}\right\}\cdot\log n\log\log n\right)\cdot\frac{\prod_{i\in[r]}\bar{\mathsf{c}}_i}{\bar{\mathsf{h}}}\right)\\
&=O\left(\min\left\{\max_{i\in[r]}\{cost(C_i)\}\cdot\frac{\prod\bar{\mathsf{c}}_i}{\bar{\mathsf{h}}},m\right\}\cdot\log n\log\log n\right)\\
&=O\left(\min\left\{\textsc{decomp-cost}(G,H,D^*(H)),m,n\right\}\cdot\log n\log\log n\right),
\end{aligned}$$

as claimed. ◀

References

1. Nesreen K Ahmed, Jennifer Neville, Ryan A Rossi, and Nick Duffield. Efficient graphlet counting for large networks. In *2015 IEEE International Conference on Data Mining*, pages 1–10. IEEE, 2015.
2. Maryam Aliakbarpour, Amartya Shankha Biswas, Themis Gouleakis, John Peebles, Ronitt Rubinfeld, and Anak Yodpinyanee. Sublinear-time algorithms for counting star subgraphs via edge sampling. *Algorithmica*, 80(2):668–697, 2018.
3. Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A Simple Sublinear-Time Algorithm for Counting Arbitrary Subgraphs via Edge Sampling. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, volume 124 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:20, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ITCS.2019.6.

[9] To avoid multiplicity issues, if some components are repeated in the decomposition more than once, then we can assign ids to small components and verify they are sampled in ascending id order.

4 Albert Atserias, Martin Grohe, and Dániel Marx. Size bounds and query plans for relational joins. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 739–748. IEEE, 2008.

5 Haim Avron. Counting triangles in large graphs using randomized matrix trace estimation. In *Workshop on Large-scale Data Mining: Theory and Applications*, volume 10, pages 10–9, 2010.

6 Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge estimation with independent set oracles. *arXiv preprint arXiv:1711.07567*, 2017.

7 Suman K. Bera, Noujan Pashanasangi, and C. Seshadhri. Linear time subgraph counting, graph degeneracy, and the chasm at size six. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, pages 38:1–38:20, 2020. doi:10.4230/LIPIcs.ITCS.2020.38.

8 Amartya Shankha Biswas, Talya Eden, and Ronitt Rubinfeld. Towards a decomposition-optimal algorithm for counting and sampling arbitrary motifs in sublinear time, 2021. arXiv:2107.06582.

9 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Counting paths and packings in halves. *Algorithms - ESA 2009*, page 578–586, 2009. doi:10.1007/978-3-642-04128-0_52.

10 Xi Chen, Amit Levi, and Erik Waingarten. Nearly optimal edge estimation with independent set queries. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2916–2935, 2020. doi:10.1137/1.9781611975994.177.

11 Graham Cormode and Hossein Jowhari. L p samplers and their applications: A survey. *ACM Computing Surveys (CSUR)*, 52(1):1–31, 2019.

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

13 Talya Eden, Amit Levi, Dana Ron, and C Seshadhri. Approximately counting triangles in sublinear time. *SIAM Journal on Computing*, 46(5):1603–1646, 2017.

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

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

16 Talya Eden, Dana 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, 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 722–734, 2018. doi:10.1145/3188745.3188810.

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

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

19 Talya Eden and Will Rosenbaum. Lower bounds for approximating graph parameters via communication complexity. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, volume

116 of *LIPIcs*, pages 11:1–11:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.11.

20 Talya Eden and Will Rosenbaum. On sampling edges almost uniformly. In Raimund Seidel, editor, *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New Orleans, LA, USA*, volume 61 of *OASICS*, pages 7:1–7:9. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/OASIcs.SOSA.2018.7.

21 Patrick Eichenberger, Masaya Fujita, Shane T Jensen, Erin M Conlon, David Z Rudner, Stephanie T Wang, Caitlin Ferguson, Koki Haga, Tsutomu Sato, Jun S Liu, et al. The program of gene transcription for a single differentiating cell type during sporulation in bacillus subtilis. *PLoS biology*, 2(10):e328, 2004.

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

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

24 Jacob Fox, Tim Roughgarden, C. Seshadhri, Fan Wei, and Nicole Wein. Finding cliques in social networks: A new distribution-free model. *SIAM J. Comput.*, 49(2):448–464, 2020. doi:10.1137/18M1210459.

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

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

27 Shweta Jain and C. Seshadhri. A fast and provable method for estimating clique counts using turán's theorem. In *Conference on the World Wide Web*, pages 441–449, 2017.

28 Krzysztof Juszczyszyn, Przemysław Kazienko, and Katarzyna Musiał. Local topology of social network based on motif analysis. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 97–105. Springer, 2008.

29 Tali Kaufman, Michael Krivelevich, and Dana Ron. Tight bounds for testing bipartiteness in general graphs. *SIAM Journal on Computing*, 33(6):1441–1483, 2004. doi:10.1137/S0097539703436424.

30 Tong Ihn Lee, Nicola J Rinaldi, François Robert, Duncan T Odom, Ziv Bar-Joseph, Georg K Gerber, Nancy M Hannett, Christopher T Harbison, Craig M Thompson, Itamar Simon, et al. Transcriptional regulatory networks in saccharomyces cerevisiae. *science*, 298(5594):799–804, 2002.

31 Wenzhe Ma, Ala Trusina, Hana El-Samad, Wendell A Lim, and Chao Tang. Defining network topologies that can achieve biochemical adaptation. *Cell*, 138(4):760–773, 2009.

32 DE Nelson, AEC Ihekwaba, M Elliott, JR Johnson, CA Gibney, BE Foreman, G Nelson, V See, CA Horton, DG Spiller, et al. Oscillations in nf-κb signaling control the dynamics of gene expression. *Science*, 306(5696):704–708, 2004.

33 Duncan T Odom, Nora Zizlsperger, D Benjamin Gordon, George W Bell, Nicola J Rinaldi, Heather L Murray, Tom L Volkert, Jörg Schreiber, P Alexander Rolfe, David K Gifford, et al. Control of pancreas and liver gene expression by hnf transcription factors. *Science*, 303(5662):1378–1381, 2004.

34 Rasmus Pagh and Charalampos E Tsourakakis. Colorful triangle counting and a mapreduce implementation. *Information Processing Letters*, 112:277–281, 2012.

35 Ashwin Paranjape, Austin R Benson, and Jure Leskovec. Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 601–610. ACM, 2017.

36 Shai S Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature genetics*, 31(1):64, 2002.

37 Jakub Tětek and Mikkel Thorup. Sampling and counting edges via vertex accesses, 2021.

38 Alexandru Topirceanu, Alexandra Duma, and Mihai Udrescu. Uncovering the fingerprint of online social networks using a network motif based approach. *Computer Communications*, 73:167–175, 2016.

39 Charalampos E Tsourakakis. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *International Conference on Data Mining*, pages 608–617, 2008.

40 Jakub Tětek. Approximate triangle counting via sampling and fast matrix multiplication. *CoRR*, abs/2104.08501, 2021. `arXiv:2104.08501`.

41 John J Tyson and Béla Novák. Functional motifs in biochemical reaction networks. *Annual review of physical chemistry*, 61:219–240, 2010.

42 Virginia Vassilevska. Efficient algorithms for clique problems. *Information Processing Letters*, 109(4):254–257, 2009. `doi:10.1016/j.ipl.2008.10.014`.

43 Qiankun Zhao, Yuan Tian, Qi He, Nuria Oliver, Ruoming Jin, and Wang-Chien Lee. Communication motifs: a tool to characterize social communications. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1645–1648. ACM, 2010.

A Related Work

We note that some of the works were mentioned before, but we repeat them here for the sake of completeness. Over the past decade, there has been a growing body of work investigating the questions of approximately counting and sampling motifs in sublinear time. These questions were considered for various motifs H, classes of G, and query models.

The study of sublinear time estimation of motif counts was initiated by the works of Feige [22] and of Goldreich and Ron [25] on approximating the average degree in general graphs. Feige [22] investigated the problem of estimating the average degree of a graph, denoted d_{avg}, when given query access to the degrees of the vertices. By performing a careful variance analysis, Feige proved that $O\left(\sqrt{n/d_{\mathsf{avg}}}/\epsilon\right)$ queries are sufficient in order to obtain a $(\frac{1}{2}-\epsilon)$-approximation of d_{avg}. He also proved that a better approximation ratio cannot be achieved in sublinear time using only degree queries. The same problem was then considered by Goldreich and Ron [25]. Goldreich and Ron proved that an $(1+\epsilon)$-approximation can be achieved with $O\left(\sqrt{n/d_{avg}}\right) \cdot \mathrm{poly}(1/\epsilon, \log n)$ queries, if neighbor queries are also allowed. Building on these ideas, Gonen et al. [26] considered the problem of approximating the number of s-stars in a graph. Their algorithm only assumed neighbor and degree queries. In [2], Aliakbarpour, Biswas, Gouleakis, Peebles, and Rubinfeld and Yodpinyanee considered the same problem of estimating the number of s-stars in the augmented edqu queries model, which allowed them to circumvent the lower bounds of [26] for this problem. In [17], Eden, Ron and Seshadhari again considered this problem, and presented improved bound for the case where the graph G has bounded arboricity. In [13, 16, 18], Eden, Ron and Seshadhri considered the problems of estimating the number of k-cliques in general and in bounded arboricity graphs, in the general graph query model, and gave matching upper and lower bounds. In [40], Tětek considers both the general and the augmented query models for approximately counting triangles in the super-linear regime. In [19], Eden and Rosenbaum presented a framework for proving motif counting lower bounds using reduction from communication complexity, which allowed them to reprove the lower bounds for all of the variants listed above.

In [20, 14], Eden and Rosenbaum and Ron has initiated the study of sampling motifs (almost) uniformly at random. They considered the general graph query model, and presented upper and matching lower bounds up to $\mathrm{poly}(\log n/1/\epsilon)$ factors, for the task of sampling edges almost uniformly at random, both for general graphs and bounded arboricity graphs. Recently, Tětek and Thorup [37] presented an improved analysis which reduced the dependency in

ϵ to $\log(1/\epsilon)$. This result implies that for all practical applications, the edge sampler is essentially as good as a truly uniform sampler. They also proved that given access to what they refer to as hash-based neighbor queries, there exists an algorithm that samples from the exact uniform distribution. The authors of [14] also raised the question of approximating vs. sampling complexity, and gave preliminary results that there exists motifs H (triangles) and classes of graphs G (bounded arboricity graphs) in which approximating the number of H's is strictly easier than sampling an almost uniformly distributed copy of H. This question was very recently resolved by them, proving a separation for the tasks of counting and uniformly sampling cliques in bounded arboricity graphs [15].

A significant result was achieved recently, when Assadi, Kapralov and Khanna gave an algorithm for approximately counting the number of copies of any given general H, in the edge queries augmented query model. They also gave a matching lower bound for the case that H is an odd cycle. Fichtenberger, Gao and Peng presented a cleaner algorithm with a mich simplified analysis for the same problem, that also returns a uniformly distributed copy of H.

Another query model was suggested recently by Beame et al. [6], which assumes access to only *independent set* (IS) queries or *bipartite independent set* (BIS) queries . Inspired by group testing, IS queries allow to ask whether a given set A is an independent set, and BIS queries allow to ask whether two sets A and B have at least one edge between them. In this model they considered the problem of estimating the average degree and gave an $O(n^{2/3}) \cdot \text{poly}(\log n)$ algorithm using IS queries, and $\text{poly}(\log n)$ algorithm using BIS queries. Chen, Levi and Waingarten [10] later improved the first bound to $O(n/\sqrt{m}) \cdot \text{poly}(\log n)$ and also proved it to be optimal.