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

Amartya Shankha Biswas
CSAIL, MIT, Cambridge MA, USA

Talya Eden
CSAIL at MIT, USA

Ronitt Rubinfeld
CSAIL at MIT, USA

Abstract

Counting and uniformly sampling motifs in a graph are fundamental algorithmic tasks with numerous applications across multiple fields. Since these problems are computationally expensive, recent efforts have focused on devising sublinear-time algorithms for these problems. We consider the model where the algorithm gets a constant size motif H and query access to a graph G, where the allowed queries are degree, neighbor, and pair queries, as well as uniform edge sample queries. In the sampling task, the algorithm is required to output a uniformly distributed copy of H in G (if one exists), and in the counting task it is required to output a good estimate to the number of copies of H in G.

Previous algorithms for the uniform sampling task 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, for any motif H whose decomposition contains at least two components or at least one star, is always preferable. 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. We further show how to use our sampling algorithm to get an approximate counting algorithm, with essentially the same complexity.

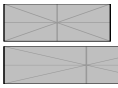
Finally, we prove that this algorithm is *decomposition-optimal* for decompositions that contain at least one odd cycle. That is, we prove that for any decomposition D that contains at least one odd cycle, there exists a motif H_D with decomposition D, and a family of graphs $\mathcal{G}$, so that in order to output a uniform copy of H in a uniformly chosen graph in $\mathcal{G}$, the number of required queries matches our upper bound. 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...

Funding *Amartya Shankha Biswas*: Big George Ventures Fund, MIT-IBM Watson AI Lab and Research Collaboration Agreement No. W1771646, NSF awards CCF-1733808 and IIS-1741137
Talya Eden: This work was supported by the National Science Foundation under Grant No. CCF-1740751, Eric and Wendy Schmidt Fund for Strategic Innovation, and Ben-Gurion University of the Negev.
Ronitt Rubinfeld: This work was supported by the National Science Foundation under Grants No. CCF-2006664, CCF-1740751, IIS-1741137, and by the Fintech@CSAIL Initiative.



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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., [35, 29, 20, 32, 31, 42, 27, 34, 37, 40, 30]). 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 [41, 8]. On the more applied side, there is an extensive study of practical algorithms for approximate motif counting (e.g., [38, 5, 33, 1, 26, 11, 7, 23]). 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 [28].

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 [22]. The algorithms of [3, 22] 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 [22] has expected running time [3] $O\left(\frac{m^{\rho(H)}}{\bar{\mathtt{h}}}\right)$ for the task of uniformly sampling a copy of H, where $\bar{\mathtt{h}}$ is the number of copies of H in G, and m is the number of

[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.

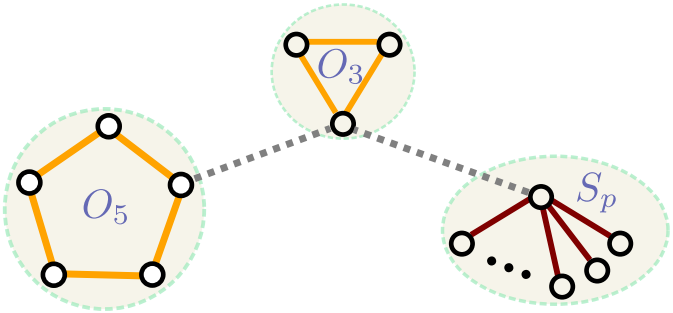


Figure 1 An example of an optimal decomposition of a motif H into odd cycles and stars. The orange edges have weight 1/2, the red edges have weight 1, and the dotted edges have zero weight.

oriented edges[4] in G. The algorithm in [3] for the estimation task has the same complexity up to $\text{poly}(\epsilon, |H|, \log n)$ factors.

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 $\text{DECOMP-COST}(G, H, D^*(H))$. 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 $\Omega(\min\{\text{DECOMP-COST}(G, H, D^*(H)), m\})$ 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 [16, 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 [10] 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 [22]. 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 [25, 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 [25] 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 13, 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 [22, 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 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 [22], 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 $O(\log n \log\log n)$ term, 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

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

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, 22] 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 [18], a single odd cycle [3], or a single star [25, 2, 18]. 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 Section 5.1, some of the constraints on these counts (detailed in Definition 17) 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.

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 $D^*(H) = \{O_{k_1}, \ldots, O_{k_q}, S_{p_1}, \ldots, S_{p_\ell}\}$. 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 $\mathcal{D}(H) = \{O_{k_1}, \ldots, O_{k_q}, S_{p_1}, \ldots, S_{p_\ell}\} = \{C_i\}_{i \in r}$ 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 $\{O_{k_1}, \ldots, O_{k_q}\}$ be the odd-cycles in the support of x^*, and let $\{S_{p_1}, \ldots, S_{p_\ell}\}$ be the stars. We refer to $D^*(H) := \{O_{k_1}, \ldots, O_{k_q}, S_{p_1}, \ldots, S_{p_\ell}\}$ 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 [22], 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 [22] 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 [25, 2, 16] 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 [22], 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 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

[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.

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.

3.1.0.1 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. In order to prove Theorem 6, we first prove Theorem 7 (in Section 5), and then prove that Theorem 6 follows from Theorem 7 (in Section 5.4). We first explain the intuition as to why Theorem 6 follows from Theorem 7.

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. We formalize this argument in Lemma 24, and 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 $\Omega\left(\min\left\{m, \text{DC}\right\}\right)$ 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 [18]). 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 [18], 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, 16].

▶ **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 13.
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 $d_{ub} \geq d_{max}$.

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

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

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

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 [22].

▶ **Lemma 14** (Lemma 3.3 in [22], 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 [24] to obtain an estimate of m.

▶ **Theorem 16** ([24], 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.

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

Sample-H (H, n)

1. Compute a 2-factor estimate $\hat{m}$ of m by invoking the algorithm of [24] 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 degree queries and $2m$ neighbor queries, or $10m \log n$ uniform edge queries

▶ **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*

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

Proof. By Theorem 16, when invoked with a value $\epsilon = 1/2$, the edge estimation algorithm of [24] 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 13, 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 $\frac{1}{\prod_{i=1}^{q} \bar{\mathsf{o}}_{k_i} \cdot \prod_{i=1}^{\ell} \bar{\mathsf{s}}_{p_i}}$. [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

[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.

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 13, 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. ◀

5 Lower Bounds

In this section we prove our main lower bounds statements, Theorem 6 and Theorem 7. We defer the proof that the former follows from the latter to Section 5.4, and start with proving Theorem 7, stated here again for the sake of convenience.

▶ **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, $\bar{\mathsf{h}}$ and a set of good counts, $\{\bar{\mathsf{c}}_i\}_{i\in[r]} = \{\bar{\mathsf{o}}_{k_1}, \ldots, \bar{\mathsf{o}}_{k_q}, \bar{\mathsf{s}}_{p_1}, \ldots, \bar{\mathsf{s}}_{p_\ell}\}$, 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 $\{\bar{\mathsf{c}}_i\}_{i\in[r]}$, the number of copies of H_D is $\bar{\mathsf{h}}$, and the expected query complexity of sampling (whp) a uniformly distributed copy of H_D in a uniformly chosen $G \in \mathcal{G}$ is*

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

We next formalize the definition of *good counts*.

5.1 Good counts

▶ **Definition 17** (Good counts). *We say that a set of counts n, m and $\bar{\mathsf{o}}_{k_1}, \ldots, \bar{\mathsf{o}}_{k_q}, \bar{\mathsf{s}}_{p_1}, \ldots, \bar{\mathsf{s}}_{p_\ell}, \bar{\mathsf{h}}$ is* good *if the following hold.*

1. *The counts are realizable; that is, there exist a graph G and a motif H_D with optimal decomposition D that realize these counts.*
2. *The max component cost is due to an odd cycle component. That is, $argmax_{i\in[r]}\{cost(C_i)\} = O_{k_i}$ for some odd cycle component $O_{k_i} \in D$. Assume without loss of generality that O_{k_1} is the odd cycle that maximizes $\max_{i\in[r]}\{cost(C_i)\}$.*
3. *$\forall k_j > k_i$, if $\bar{\mathsf{o}}_{k_i} \leq \sqrt{m}^{k_i-1}$, then , $\left(\bar{\mathsf{o}}_{k_j}\right)^{1/(k_j-1)} \geq \left(\bar{\mathsf{o}}_{k_i}\right)^{1/(k_i-1)}$. Otherwise, if $\bar{\mathsf{o}}_{k_i} > \sqrt{m}^{k_i-1}$, $\left(\bar{\mathsf{o}}_{k_j}\right)^{1/k_j} \geq \left(\bar{\mathsf{o}}_{k_i}\right)^{1/k_i}$.*
4. *For every $j \in [\ell]$, $\bar{\mathsf{s}}_{p_j} \geq \sqrt{m}^{p_j+1}$.*
5. *At least one of the followings hold.*
   a. *Let k_* be the index of the O_k that maximizes $\bar{\mathsf{o}}_k^{1/k}$. There exists at least one star S_p in D with $\bar{\mathsf{s}}_p = \omega\left(m \cdot (\bar{\mathsf{o}}_{k_*})^{(p+1)/k_*}\right)$. Observe that it always holds that $\bar{\mathsf{o}}_{k_*}^{1/k_*} \leq \sqrt{m}$, so if $\bar{\mathsf{s}}_p = \omega(\sqrt{m}^{p+3})$ then this constraint holds.*
   b. *For every $k_i \leq k_1$, $\bar{\mathsf{k}}_i \leq \sqrt{m}^{k_i-1}$.*
6. *At least one of the followings hold.*
   a. *For at least one of the cycles O_k, it holds that $\bar{\mathsf{o}}_k \leq \sqrt{m}^{k-1}$, and for every p, $\bar{\mathsf{s}}_p \geq n^p$.*
   b. *The $\bar{\mathsf{s}}_{p_i}$ counts are such there exists a set A of $\sqrt{m}$ integers $a_1, \ldots, a_{\sqrt{m}}$ so that $\forall i$, $a_i \leq n$, $\sum_i a_i \leq m$, and $\sum_i a_i^{p_i} = \bar{\mathsf{s}}_{p_i}$.*

As discussed in the introduction, some of the above constraints are unavoidable, and some arise due to the way we construct the graphs G in the hard family $\mathcal{G}$. Details follow.

1. Constraint 1 simply states that the given counts can be realized by some graph and is therefore unavoidable.
2. Constraint 2 implies that our upper bound is tight only in the case that the max cost is due to an odd cycle and not due to a star component. We leave it as an open question whether for the case that the max component cost is due to a star, a new lower bound can be designed or an improved algorithm can be devised.

The rest of the constraints arise from the way we construct the basic structure of the graphs in the "hard" family of graphs in the proof of the lower bound.

3. Constraint 3: for each cycle O_{k_i} such that $\bar{\mathsf{o}}_{k_i \geq \sqrt{m}^{k_i-1}}$, we "pack" the $\Theta(\bar{\mathsf{o}}_{k_i})$ k_i length odd cycles in a k_i-partite subgraph. This inadvertently results in the creation of $\Theta((\bar{\mathsf{o}}_{k_i})^{k_j/k_i})$ odd-cycles for any $k_j \geq k_i$ length odd cycle component.
4. Constraint 4: Recall that in order to prove the lower bound we "hide" as set of t crucial edges which create $\Theta(\bar{\mathsf{h}})$ of the copies of H_D. To hide the edges, we use a subgraph with density $\Theta(\sqrt{m})$, which again inadvertently induces $\Theta(\sqrt{m}^{p+1})$ p-stars for every $p \in [\sqrt{m}]$.
5. Constraint 5: Let k' denote the min length odd cycle component in D. If for example $\bar{\mathsf{o}}_{k'} = \sqrt{m}^{k'}$, then our gadget for creating $\bar{\mathsf{o}}_k$ odd cycles also maximizes (up to constant factors) the counts of all odd cycles for every k_i, and therefore might induce too many copies of H_D. To avoid such a scenario, we require that either there exists at least one star in D with counts strictly greater than what could be created by a cycle gadget (in 5a); or that the number of short cycles, i.e., cycles of length $k_i \leq k_1$, does not exceed $\sqrt{m}^{k_i-1}$ (in 5b). In the latter case the corresponding gadget can have a single vertex which is incident to all cycles, and therefore, no two vertex-disjoint odd cycles can be formed, so that no copies of H_D are formed solely by this gadget.

6. Constraint 6 arises from the way we connect the odd cycles and stars in the graphs of $\mathcal{G}$. The first item, 6a, simply states that the count of one of the cycles which is not the max cost cycle is not maximized. In such a case the corresponding cycle gadget will have one part with a single vertex, which will allow us to connect it to a set of n vertices that induce the $\bar{\mathtt{s}}_p$ counts in the corresponding star gadget. The second item, item 6b, states that there exists a set A of $|A| \leq \sqrt{m}$ (rather than n) integers (that will later determine the degrees of $|A|$ vertices), so that for every p, $\sum_{a_i \in A} a_i^p = \bar{\mathtt{s}}_p$. [10]

We note that while there are indeed many constraints required by our construction, these constraints are satisfiable by many sets of possible counts. Indeed in order prove that Theorem 6 follows from Theorem 7 (see proof of Lemma 24), we show that for *every* realizable value of DECOMP-COST$(G, H, D^*(H))$, *there exists* a set a set of good counts $\{\bar{\mathtt{c}}_i\}_{i\in[r]}$, which satisfies all of the constraints of Definition 17.

We continue to describe the different ingredients required for our proof. We make use of the framework for proving graph estimation lower bounds via communication complexity reductions given in [18]. The framework makes use of the following communication problem.

▶ **Theorem 18.** *In the t-Set-Disjointness variant of the Set-Disjointness problem, Alice and Bob are given $\{0,1\}$-matrices $\vec{x}, \vec{y} \in \{0,1\}^N \times \{0,1\}^N$, respectively. Under the promise that either there exists t pairs of indices such that $x_{i,j} = y_{i,j} = 1$, or that there exists 0 such indices. The goal of Alice and Bob is then to distinguish between these two cases. We will denote the set of intersections by $\vec{z}$, where $\vec{z}_{i,j} = \vec{x}_{i,j} \wedge \vec{y}_{i,j}$.*

The idea is to construct an embedding of the t-Set-Disjointness communication problem to a graph $G_{\vec{z}}$, such that the following holds. First, every query performed on $G_{\vec{z}}$ can be answered by exchanging B bits of communication for a constant B. Second, one can solve the given t-Set-Disjointness instance by sampling uniformly distributed copies of H_D in $G_{\vec{z}}$. The parameter t in the t-Set-Disjointness problem is set according to $m, \bar{\mathtt{h}}$ and the counts of the basic components of D, to ensure that the lower bound on the communication complexity problem implies the desired lower bound specified in Theorem 7.

▶ **Theorem 19** (Corollary 2.7 in [18]). *The communication complexity of t-Set-Disjointness is $\Omega(N^2/t)$.*

We shall prove that the problem of t-Set-Disjointness can be reduced to the problem of estimating the number of copies of H in a graph $G_{\vec{z}}$, such that each query in $G_{\vec{z}}$ can be answered in constant time. Namely, we prove that for a given $\bar{\mathtt{h}}$, the graph $G_{\vec{z}}$ consists of several gadgets, that are independent of the instance $(\vec{x}, \vec{y})$, and a `CC-gadget` gadget that embeds the instance $(\vec{x}, \vec{y})$ to the graph $G_{\vec{z}}$ as follows. If $(\vec{x}, \vec{y})$ intersect, then at least a constant factor of the copies of H_D in $G_{\vec{z}}$ are contributed by this gadget, and otherwise this gadget contributes no copies. The family of graphs $\mathcal{G}$ is then defined to be the collection of graphs $\{G_{\vec{z}}\}$ for all possible $\vec{z}$ that are the intersection of an t-Set-Disjointness instance.

[10] Note that indeed there exists many *valid* counts (ones which can be realized by some graph) that satisfy this constraint. Consider first a bipartite graph $G_0 = A \cup B$ with $|A| = \sqrt{m}$, $|B| = n$, where each vertex in A has degree $\Theta(\sqrt{m})$, and each vertex in B has degree $O(\sqrt{m})$. Then in this graph, all star counts are exactly $\bar{\mathtt{s}}_p = \sqrt{m}^{p+1}$ as required by the second constraint. To get higher values of the counts $\bar{\mathtt{s}}_p$, we can simply move edges around, one edge at a time, as to skew the set of degrees of the vertices of A. Let G_t denote the graph resulting from the above process at time t. This process ends after r steps, with a graph $G_r = A' \cup B'$ as follows. A' has d_{avg} vertices with degree n, and $\sqrt{m} - d_{\mathsf{avg}}$ vertices of degree 0, and B has n vertices with degree d_{avg}. This graph maximizes the $\bar{\mathtt{s}}_p$ counts, $\bar{\mathtt{s}}_p = d_{\mathsf{avg}} \cdot n^p$ for any p. At each time step t, the set of counts $\bar{\mathtt{s}}_{p_1}, \ldots, \bar{\mathtt{s}}_{p_\ell}$ of the p_i-stars in G_t satisfies constraint 6b.

Thus by uniformly sampling copies of H_D, one can distinguish between the case that $\vec{x}, \vec{y}$ are disjoint to the case where they intersect (by sampling a constant number of copies and checking if some are contributed by the `CC-gadget`). It follows that for every N and t, $\Omega(N^2/t)$ queries are required in order to sample uniform copies of H.

Our lower bound theorem is very generic as it works for any decomposition that contains at least one cycle, and for a variety of plausible basic component counts (those that meet the constraints specified in Definition 17). Hence, we shall start with a (sketched) proof for a specific easy basic case. The ideas in proving the general case will be the same, however due to the generality of the statement, many technical difficulties arise in satisfying all counts simultaneously. Hence, we defer that analysis of the general case to Subsection 5.3.

5.2 Warm up: a lower bound for a decomposition $D = \{O_3, S_p\}$

In this section we prove the first term in our lower bound for a specific decomposition, $D = \{O_3, S_p\}$ and for the case that lower bound is sublinear in m, and the max cost in the bound is due to the O_3 component.

▶ **Theorem 20.** *Let $D = \{O_3, S_p\}$ be a decomposition and assume that we are given the counts $n, m, \bar{\mathtt{o}}_3, \bar{\mathtt{s}}_p$ and $\bar{\mathtt{h}}$. Further assume that the counts are such that $\bar{\mathtt{s}}_p \geq \sqrt{m}^{p+1}$, $\max\{cost(O_3), cost(S_p)\} = cost(O_3)$ and $\bar{\mathtt{h}} \geq \sqrt{m} \cdot \bar{\mathtt{s}}_p$. Then there exist a motif H_D with decomposition D, and a family of graphs $\mathcal{G}$ such that for every $G \in \mathcal{G}$ the counts are as above (up to constant factors), and such that sampling a uniformly distributed copy of H_D in a uniformly chosen $G \in \mathcal{G}$ requires*

$$\Omega\left(\max_i(cost(C_i)) \cdot \frac{\bar{\mathtt{o}}_3 \cdot \bar{\mathtt{s}}_p}{\bar{\mathtt{h}}}\right) = \Omega\left(\frac{m^{3/2} \cdot \bar{\mathtt{s}}_p}{\bar{\mathtt{h}}}\right)$$

queries in expectation.

Proof Sketch. By the above it holds that $\bar{\mathtt{h}} \leq \prod_i \bar{\mathtt{c}}_i$, we let $\alpha = \prod_i \bar{\mathtt{c}}_i / \bar{\mathtt{h}}$ so that $\alpha > 1$. We shall rearrange the lower bound:

$$\frac{m^{3/2} \cdot \bar{\mathtt{s}}_p}{\bar{\mathtt{h}}} = \frac{m^{3/2}}{\bar{\mathtt{o}}_3} \cdot \frac{\bar{\mathtt{o}}_3 \cdot \bar{\mathtt{s}}_p}{\bar{\mathtt{h}}} = \frac{m^{3/2}}{\bar{\mathtt{o}}_3} \cdot \alpha = \frac{m^{3/2}}{\bar{\mathtt{o}}_3/\alpha}.$$

The family $\mathcal{G}$ is the set of graphs $\{G_{\vec{z}}\}$ for all possible vectors $\vec{z} = \vec{x} \cdot \vec{y}$ where $(\vec{x}, \vec{y})$ are instances of the t-Set-Disjointness problem, for a value t that will be set shortly. Fix an instance $(\vec{x}, \vec{y})$ of t-Set-Disjointness and let $\vec{z} = \vec{x} \cdot \vec{y}$. We shall describe an embedding from $\vec{z}$ to $G_{\vec{z}}$ so that sampling a uniformly distributed copy of H_D in $G_{\vec{z}}$ solves t-Set-Disjointness on $(\vec{x}, \vec{y})$. We set $t = \lfloor |T|/\sqrt{m} \rfloor = \lfloor \bar{\mathtt{o}}_3/(\sqrt{m} \cdot \alpha) \rfloor$ so that $\frac{m^{3/2}}{o_3/\alpha} = \frac{m}{t}$ and we consider the case that $N = \sqrt{m}$, so that $\Omega(N^2/t) = \Omega(m/t) = \Omega(m^{3/2}/(o_3/\alpha))$. Observe that this setting is valid since, by the assumption that the complexity is sublinear in m, it holds that $\frac{m^{3/2} \cdot \bar{\mathtt{s}}_p}{\bar{\mathtt{h}}} \leq m$, implying that $h \geq \sqrt{m} \cdot \bar{\mathtt{s}}_p$. Therefore, $\frac{\bar{\mathtt{o}}_3 \cdot \bar{\mathtt{s}}_p}{\alpha} \geq \sqrt{m} \cdot \bar{\mathtt{s}}_p$, and it follows that $o_3/(\sqrt{m} \cdot \alpha) \geq 1$ so that $t \geq 1$.

We let H_D be the motif of a triangle connected by a single edge to a star S_p. To describe the graph $G_{\vec{z}}$, we describe a corresponding gadget to each of the components O_3 and S_p in D. The gadget corresponding to the star is a bipartite graph over two sets R_1, R_2 such that $|R_1| = 1$ and $|R_2| = \bar{\mathtt{s}}_p^{1/p}$ (if $\bar{\mathtt{s}}_p > n^p$, then we can modify R_1 to be of size $\lfloor \bar{\mathtt{s}}_p/n \rfloor$ and R_2 to be of size n). There is a complete bipartite graph between R_1 and R_2.

The gadget used to create the $|T|$ odd cycles of length 3 has $3 + 2 = 5$ sets $R_1, R_2, R_3, R'_1, R'_2$, each of size $\sqrt{m}$. There is a complete bipartite graph between the

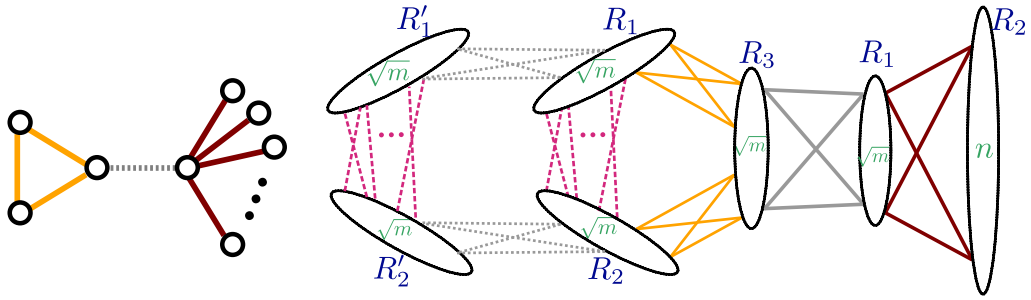


Figure 2 (a) The motif H_D for $D = \{O_3, S_p\}$ (b) The graph $G \setminus G'$. Orange/red crossed lines indicate a complete bipartite graph of intra-gadget edges, gray crossed lines indicate a complete bipartite graph of inter-gadget edges, and pink dotted lines indicate "potential" edges – i.e., ones whose existence depends on the t-Set-Disjointness instance $\vec{x}, \vec{y}$.

sets R_1 and R_3 and R_2 and R_3. The edges between the sets R_1, R_2, R'_1, R'_2 are determined according to the t-Set-Disjointness instance $\vec{x}, \vec{y}$ as follows. For every pair of indices $i, j \in \sqrt{m}$, if $\vec{x}_{ij} = \vec{y}_{ij} = 1$ then we add the edge (r_1^i, r_2^j) and let as the $(j-i)^{\text{th}}$ edge of r_1^i and r_2^j, and the edge (r_1^j, r_2^i) as the $(j-i)^{\text{th}}$ edge of r_1^j and r_2^i. We also add the edges $((r')_1^i, (r')_2^j)$ and $((r')_1^j, (r')_2^i)$ and label them as the $(j-i)^{\text{th}}$ edge of their endpoints. Otherwise, we add the edges $(r_1^i, (r'_1)^j)$, $(r_1^j, (r'_1)^i)$, $(r_2^i, (r'_2)^j)$ and $(r_2^j, (r'_2)^i)$ to the gadget, and label them as the $(j-i)^{\text{th}}$ edge of their endpoints. Hence, if $(\vec{x}, \vec{y})$ is a YES instance we get that the CC-gadget has $t \cdot \sqrt{m}^{k-2}$ odd cycles, and if it is a NO instance then the gadget induces no cycles. See Figure 2(b) for an illustration. Furthermore, in both cases, the degrees of all vertices in the gadget are exactly $2 \cdot \bar{\mathtt{o}}_k^{1/k}$, and the "gadget edges" of the vertices in R_1, R_2 are their first $\sqrt{m}$ edges (in terms of edge labels). We furthermore add a complete bipartite graph between the two R_1 sets of the two gadgets. Observe that at this point, the count $\bar{\mathtt{o}}_3$ is not satisfied as G only contains $|T| < \bar{\mathtt{o}}_3$ triangles. As the set of counts is valid, there exists a graph G' for which they are all satisfied. To finalize the construction, we add the graph G' to G as a subgraph as a disconnected component.

By the construction of the gadgets, there are $\Theta(\bar{\mathtt{s}}_p + \sqrt{m}^{p+1}) = \Theta(\bar{\mathtt{s}}_p)$ copies of S_p in the graph, as well as $\bar{\mathtt{o}}_3$ triangles, $\Theta(n)$ vertices and $\Theta(m)$ edges. Hence, the basic counts are satisfied (up to constant factors).

By construction of the O_3 gadget, we have that if $\vec{x} \cdot \vec{y} = 0$, then the graph $G \setminus G'$ is bipartite, and otherwise it contains $t\sqrt{m} \cdot \bar{\mathtt{s}}_p = (\bar{\mathtt{o}}_3/\alpha) \cdot \bar{\mathtt{s}}_p = \bar{\mathtt{h}}$ many copies of H_D. Hence, given an algorithm $\mathcal{A}$ that samples uniformly distributed copies of H_D, to solve the given t-Set-Disjointness instance Alice and Bob proceed as follows. First they implicitly construct the graph $G_{\vec{z}}$ as described. Then, Alice and Bob both invoke $\mathcal{A}$ using their shared randomness as the randomness of $\mathcal{A}$ (so that $\mathcal{A}$ is now deterministic and Alice and Bob see the same queries during $\mathcal{A}$'s run). Whenever $\mathcal{A}$ queries $G_{\vec{z}}$, they either answer the query themselves (in case it does not depend on the input instance) or communicate $O(B)$ bits to answer it. They repeat this process for 10 times. Once all invocations of $\mathcal{A}$ conclude, if all the returned copies of H_D are from G' then Alice and Bob respond that the input matrices are disjoint, and otherwise, they respond that the matrices intersect. In case the matrices intersect, 1/2 of the copies of H_D are in $G \setminus G'$, and therefore, Alice and Bob respond incorrectly with probability $1/2^{10}$. If however the sets do not intersect, Alice and Bob respond correctly with probability 1.

Assume that each query can be answered by Alice and Bob exchanging $O(B)$ bits of

communication. Then the number of expected number of queries Q performed by $\mathcal{A}$ is lower bounded by $Q \cdot B = \Omega(m/t \cdot B)$, and for $B = O(1)$ we get $Q = \Omega\left(\frac{m^{3/2} \cdot \bar{\mathtt{s}}_p}{B \cdot \bar{\mathtt{h}}}\right)$.

It remains to bound B. Here we only sketch the proof, as the full proof is identical for this case and the general one, and it is given in Lemma 23. First observe that the degrees of all vertices are determined independently of the input instance to t-Set-Disjointness. Indeed all vertices in the cycle gadget have degrees $2\sqrt{m}$ and the structure of the star gadget does not depend on $\vec{x}, \vec{y}$. For a pair query (u, v), unless both vertices belong to $R_1 \cup R_2 \cup R'_1 \cup R'_2$ the answer is independent to the input instance. Otherwise, assume for example that $u = r_i^1 \in R_1$ and $v = r_j^2 \in R_2$. Then to answer the query, Alice and Bob send each other the bits x_{ij}, y_{ij}, and if they intersect they answer that the pair is an edge, and otherwise it is not. Other pair queries within these sets can be answered similarly, and so does neighbor queries on vertices in these sets. Hence, each query can be answered by exchanging $O(B) = O(1)$ bits of communication, and we get $Q = \Omega\left(\frac{m^{3/2} \cdot \bar{\mathtt{s}}_p}{\bar{\mathtt{h}}}\right)$, as required. ◀

5.3 Proof of Theorem 7

Let $D = \{O_{k_1}, \ldots, O_{k_q}, S_{p_1}, \ldots, S_{p_\ell}\}$. To prove the lower bound of Theorem 7, we first construct a graph H_D with optimal decomposition is D. We then construct a family of graphs $\mathcal{G}$ such that each $G \in \mathcal{G}$ satisfies all the counts and constraints of the theorem, and so that sampling a uniformly distributed copy of H_D in a uniformly chosen $G \in \mathcal{G}$ requires $\Omega\left(\min\left\{cost(O_k) \cdot \frac{\prod_i \bar{\mathtt{c}}_i}{\bar{\mathtt{h}}}, m\right\}\right)$ samples.

Constructing the motif H_D. Given a decomposition D we construct the graph H_D as follows. Recall that O_{k_1} denotes the odd cycle with maximum cost, and denote its vertices by $v_1^{k_1}, \ldots v_{k_1}^k$. If there exists a star S_p in D with count $\bar{\mathtt{s}}_p > |H| \cdot \sqrt{m}^{p+1}$, then we connect its star center to one of the vertices of O_{k_1}. If for at least one of the cycles in D, $\bar{\mathtt{o}}_k \leq \sqrt{m}^{k-1}$, then we connect to it all the stars of D, except for the one that is connected to O_{k_1}. We connect the rest of the components of D with a single edge to O_{k_1} , where stars are connected through their star center, and odd cycles are connected through arbitrary vertices in each of the cycles.

Constructing the graph family of graphs $\mathcal{G}$. The basic structure of all graphs G in the family $\mathcal{G}$ will be the same, except for a small set of edges which will be determined according to the t-Set-Disjointness instance $(\vec{x}, \vec{y})$, or more specifically, according to $\vec{z} = \vec{x} \cdot \vec{y}$. To construct the family of graphs $\{G_{\vec{z}}\}$, we first define gadgets that correspond to the stars and odd cycles of D.

We differentiate between *short* odd cycles of length k_i for $k_i \leq k_1$ (if such exist in D), and those with higher lengths than k_1. The reason is that we want the gadgets corresponding to short odd cycles to create $\bar{\mathtt{o}}_k$ odd cycles, while not creating "too many" k_1 odd cycles. (This is also the reason behind constraint 5b.)

- `cycle-gadget`: Given O_k and $\bar{\mathtt{o}}_k$ such that $\bar{\mathtt{o}}_k > \sqrt{m}^{k-1}$, this gadget is a complete k-partite graph, comprising of sets of vertices $R_1, R_2, \cdots, R_k$, each of size $\Theta(\bar{\mathtt{o}}_k^{1/k})$. Each adjacent pair $R_i, R_{i+1(\mod k)}$ induces a complete bipartite graph. (Observe that for every graph $\bar{\mathtt{o}}_k \leq m^{k/2}$ and therefore for every $i \in [k]$, $|R_i| \leq \sqrt{m}$.)
- `few-cycles-gadget`: Given O_k and $\bar{\mathtt{o}}_k$ such that $\bar{\mathtt{o}}_k \leq \sqrt{m}^{k-1}$, this gadget has a set R_1 consisting of a single vertex v_1 and $k_1 - 1$ sets R_i for $i \in [2, k-1]$, each of size $\bar{\mathtt{o}}_{k_1}^{1/(k_1-1)}$. The sets form a k_1-tripartite motif.
- `star-gadget`: Recall that we assume that the counts $\bar{\mathtt{s}}_{p_i}$ are either such that there exists a cycle O_k with length $\bar{\mathtt{o}}_k \leq \sqrt{m}^{k-1}$, or that each count $\bar{\mathtt{s}}_p$ can be satisfied by a set A of

$\sqrt{m}$ numbers, $a_1, \ldots, a_{\sqrt{m}}$. That is, $\bar{\mathtt{s}}_p = \sum_{i \in A} (a_i)^p$.
In the former case, the star gadget is a bipartite motif $R_1 \cup R_2$, where $|R_1| = |R_2| = n$ and the degrees of the vertices in R_1 are such that $\sum_{v \in R_1} d(v)^p = \bar{\mathtt{s}}_p$. Due to constraint 1, such a setting of degrees exists. The edges going from R_1 to R_2 are spread evenly among the vertices of R_2, so that $\forall r_i^2 \in R_2, ; d(r_2) \leq d_{\mathsf{avg}}$.
In the latter case, the star gadget is a bipartite motif $R_1 \cup R_2$, where $|R_1| = \sqrt{m}$ and $\forall r_i^1 \in R_1,\ d(r_i^1) = a_i$. The set R_2 is of size n, and the edges from R_1 are distributed evenly among the vertices of R_2.

To embed the t-Set-Disjointness instance to $G_{\vec{z}}$, we use the following `CC-gadget` that corresponds to O_{k_1} which is (one of) the maximum cost odd cycle in D. Since this gadget is used to distinguish the two families of graphs, it appears in two forms, corresponding to the YES and NO instance of the problem.

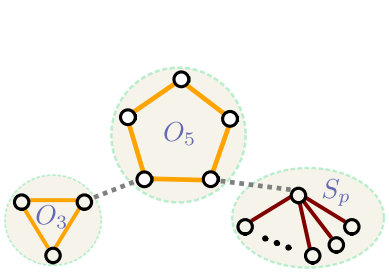
- `CC-gadget`: This gadget will correspond to the odd cycle of length k_1 in H_D (a maximum cost odd cycle). The gadget contains k_1 sets $R_1, \ldots, R_{k_1}$ and two additional sets R'_1, R'_2, all of size $\sqrt{m}$. Between every pair of sets $R_i, R_{i+1(\ mod \sqrt{m})}$, except between the pair R_1, R_2, there is a complete bipartite set. The edges between the sets $R_1, R_2, R'1, R'_2$ are determined according to the instance $(\vec{x}, \vec{y})$ as follows.
  For every pair of indices $i, j \in \sqrt{m}$, if $\vec{x}_{ij} = \vec{y}_{ij} = 1$ then we add the edge (r_1^i, r_2^j) as the $(j-i)^{\text{th}}$ edge of r_1^i, and the edge (r_1^j, r_2^i) as the $(j-i)^{\text{th}}$ edge of r_1^j and r_2^i. We also add the edges $((r')_1^i, (r')_2^j)$ and $((r')_1^j, (r')_2^i)$ and label them as the $(j-i)^{\text{th}}$ edge of their endpoints. Otherwise, $\vec{x}_{ij} = \vec{y}_{ij} = 0$, and we add the edges $(r_1^i, (r'_1)^j)$, $(r_1^j, (r'_1)^i)$, $(r_2^i, (r'_2)^j)$ and $(r_2^j, (r'_2)^i)$ to the gadget, and label them as the $(j-i)^{\text{th}}$ edge of their endpoints. Hence, if $(\vec{x}, \vec{y})$ is a YES instance we get that there are t edges between R_1 and R_2, and so the `CC-gadget` has $t \cdot \sqrt{m}^{k-2}$ many k_1 cliques. Otherwise, there are no edges between R_1 and R_2, and so the gadget is bipartite and induces no odd cycles.

See Figure 3(b) for an illustration of the different gadgets corresponding to the basic components of H_D.

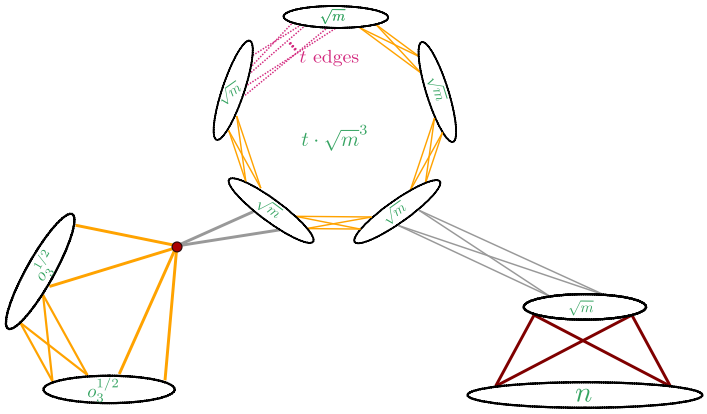
Fix an input instance $\vec{x}, \vec{y}$ and let $\vec{z} = \vec{x} \cdot \vec{y}$. The graph $G_{\vec{z}}$ contains one `CC-gadget` that corresponds to the O_{k_1} component. For any other $O_k, k \leq k_1$, if $k \leq k_1$ or $\bar{\mathtt{o}}_k \leq \sqrt{m}^{k-1}$, the graph contains a corresponding `few-cycles-gadget`, and otherwise, the graph contains a `cycle-gadget`. For all stars S_{p_j} we add a `star-gadget`. To connect the different gadgets, for each edge between two odd cycles, or between an odd cycle ant a star in H_D, we add a complete bipartite graph between the two sets R_1 of the corresponding gadgets. The way that the components of D are connected, and the construction of the gadgets of $G_{\vec{z}}$, ensure that this can be performed without exceeding $\Theta(m)$ edges between any two sets in $G_{\vec{z}}$. (Since all sets of odd cycle gadgets are of size $\sqrt{m}$, and since R_1 sets of star gadgets with $|R_1| = n$ are only connected to sets R_1 of odd cycles for which $|R_1| = 1$.) Finally, we add to $G_{\vec{z}}$ a graph G' for which all of the given counts are satisfied (recall there exists such a graph as we assume that the counts are valid). See Figure 3 for an illustration of a graph $G_{\vec{z}}$ for some $|\vec{z}| = t$ and motif H_D.

Proving the lower bound. We first consider the case that the $\max_{i \in [r]} \{cost(C_i)\} \cdot \frac{\prod \bar{\mathtt{c}}_i}{\bar{\mathtt{h}}} \leq m$. As in the warm up case, we shall prove the lower bound by "hiding" $\bar{\mathtt{h}}$ copies of H_D using a hidden set T of $|T|$ k_1-odd cycles. That is, these $|T|$ odd cycles will be added to the graph if and only if the matrices $\vec{x}$ and $\vec{y}$ intersect, and in turn they will create a constant number of copies of H_D to $G_{\vec{z}}$.

We start by rearranging the lower bound terms and determining the values of $|T|$ and t. Let $\alpha = \prod_i c_i / \bar{\mathtt{h}}$ so that $\alpha \geq 1$. By the assumption that the lower bound is sublinear in m,



(a) motif H_D for $D = \{O_3, O_5, S_p\}$



(b) Complete lower bound construction of $G \setminus G'$ using all three types of gadgets: clockwise from top, we have `CC-gadget`, `star-gadget`, and `few-cycles-gadget`.

Figure 3 Orange/red crossed lines indicate a complete bipartite graph of intra-gadget edges, gray crossed lines indicate a complete bipartite graph of inter-gadget edges, and pink dotted lines indicate "potential" edges – i.e., ones whose existence depends on the t-Set-Disjointness instance $\vec{x}, \vec{y}$.

we have that $cost(O_{k_1}) \cdot \alpha \leq m$, implying $\frac{m^{k_1/2}}{\bar{\mathsf{o}}_{k_1}} \cdot \alpha \leq m \Leftrightarrow \bar{\mathsf{o}}_{k_1} \geq \alpha m^{(k_1-2)/2}$. Let

$$|T| = \bar{\mathsf{o}}_{k_1}/\alpha \qquad \text{and} \qquad t = \left\lfloor |T|/\sqrt{m}^{(k_1-2)/2} \right\rfloor = \left\lfloor \bar{\mathsf{o}}_{k_1}/(\alpha\sqrt{m}^{(k_1-2)/2}) \right\rfloor$$

so that $|T| \geq m^{(k-2)/2}$ and $t \geq 1$. The lower bound we aim for is then

$$\max_{i\in[r]}\{cost(C_i)\} \cdot \frac{\prod_i \bar{\mathsf{c}}_i}{\bar{\mathsf{h}}} = cost(O_{k_1}) \cdot \alpha = \frac{m^{k_1/2}}{\bar{\mathsf{o}}_{k_1}/\alpha} = \frac{m^{k_1/2}}{|T|} = \frac{m}{t}.$$

In order to prove the lower bound, we first prove that all the given motif counts of the basic components are indeed as specified. That is, we prove the following lemma.

▶ **Lemma 21.** *Let $G_{\vec{z}}$ be as above. Then for any $\vec{z}$, $G_{\vec{z}}$ contains $\Theta(n)$ vertices, $\Theta(m)$ edges, and $\Theta(\bar{\mathsf{c}}_i)$ copies of each component C_i in D.*

Proof. The graph G' ensure that all counts are at least as specified. It remains to prove that the counts are not exceeded.

Fix an odd cycle O_{k_i}. We shall verify that is count as is required.

1. By construction, the gadget corresponding to O_{k_i} contributes $\Theta(\bar{\mathsf{o}}_{k_i})$ copies of O_{k_i}.
2. Now consider contributions from gadgets O_{k_j} for $j \neq i$.
   - If $k_j > k_i$ then such gadgets do not contribute to $\bar{\mathsf{o}}_{k_i}$, as a k_j-partite graph induces no k_i odd cycles for $k_i < k_j$.
   - If $k_j \leq k_i$ and $\bar{\mathsf{o}}_{k_j} \leq \sqrt{m}^{k_j-1}$, then by the construction of the `few-cycles-gadget` and by constraint 3, it contributes $(\bar{\mathsf{o}}_{k_j})^{(k_i-1)/(k_j-1)} \leq \bar{\mathsf{o}}_{k_i}$ odd cycles of length k_i.
   - Finally, if $k_j \leq k_i$ and $\bar{\mathsf{o}}_{k_j} > \sqrt{m}^{k_j-1}$, then by the construction of the `cycle-gadget` and by constraint 3, it contributes $(\bar{\mathsf{o}}_{k_j})^{(k_i)/(k_j)} \leq \bar{\mathsf{o}}_{k_i}$ odd cycles of length k_i.
3. If $z = \vec{0}$, the `CC-gadget` does not contribute any odd cycles, as it is bipartite. Otherwise, when $z \neq \vec{0}$, and $k_1 > k_i$, the `CC-gadget` also contributes 0 odd cycles of length $k_i < k_1$.

If $k_i \geq k_1$, then the gadget contributes $\Theta(t \cdot \sqrt{m}^{k_j-2})$ odd cycles of length k_i. Since the O_{k_1} component is the odd cycle component with maximum cost, we have that

$$m^{k_1/2}/\bar{\mathtt{o}}_{k_1} \geq m^{k_i/2}/\bar{\mathtt{o}}_{k_i} \quad \Leftrightarrow \quad \bar{\mathtt{o}}_{k_i} \geq \frac{\bar{\mathtt{o}}_{k_1}}{m^{k_1/2}} \cdot m^{k_i/2} \quad \Leftrightarrow \quad \bar{\mathtt{o}}_{k_i} \geq t \cdot m^{k_i/2+1}$$

where the last inequality is by the setting of $t = \bar{\mathtt{o}}_{k_1}/\sqrt{m}^{k_1-2}$.

Hence, summing over all contributions from all the components, we get that the number of copies of O_{k_i} is $\Theta(\bar{\mathtt{o}}_{k_i})$.

Now fix a star component S_p. The vertices of the odd cycle gadgets contributes at most $\sqrt{m}^{p+1} \leq \bar{\mathtt{s}}_p$ to the number of copies of S_p in $G_{\vec{z}}$. All star gadgets contribute $\Theta(\bar{\mathtt{s}}_p)$ contribute $\Theta(\bar{\mathtt{s}}_p)$ copies of S_p. Hence, the number of S_p stars in $G_{\vec{z}}$ is $\Theta(\bar{\mathtt{s}}_p)$. ◀

▶ **Lemma 22.** *Let $\mathcal{G}$ be the family of all graphs $G_{\vec{z}}$ such that $\vec{z} = \vec{x} \cdot \vec{y}$ for $\vec{x}, \vec{y}$ that are instances of t-Set-Disjointness. Let B be an upper bound on the number of bits it takes Alice and Bob to communicate in order to answer queries on any graph $G_{\vec{z}} \in \mathcal{G}$. Then for any $\bar{\mathtt{h}}$, and any algorithm that with high success probability samples a uniformly distributed copy of H_D from a uniformly chosen $G_{\vec{z}} \in \mathcal{G}$, the number of required queries is*

$$\Omega\left(\frac{m^{k_1/2} \cdot \prod_{i>1} \bar{\mathtt{c}}_i}{B \cdot \bar{\mathtt{h}}}, m/B\right)$$

in expectation, where $\bar{\mathtt{h}}$ denotes the number of copies of H_D in $G_{\vec{z}}$.

Proof. First assume that the first term achieves the minimum. In that case we have that $h \geq m^{k_1/2-1} \cdot \prod_{i>2} \bar{\mathtt{c}}_i$ and we aim to prove a lower bound of $\Omega\left(\frac{1}{B} \cdot \max_{i\in[r]}\{cost(C_i)\} \cdot \frac{\prod \bar{\mathtt{c}}_i}{\bar{\mathtt{h}}}\right)$. We let $t = \left\lfloor \left(m^{(k_1-2)/2} \cdot \prod_{i>1} \bar{\mathtt{c}}_i\right)/\bar{\mathtt{h}} \right\rfloor$. This t is the one which determines the t-Set-Disjointness communication problem we consider. Given a t-Set-Disjointness instance with inputs $\vec{x}$ and $\vec{y}$, we construct $G_{\vec{z}}$ as described above, where recall that $\vec{z} = \vec{x} \cdot \vec{y}$ determines the `CC-gadget`.

We first consider the case that $|\vec{z}| = t$, and argue that the number of copies of H_D in $G \setminus G'$ is $\Omega(\bar{\mathtt{h}})$. Since $|\vec{z}| = t$, the `CC-gadget` corresponding to O_{k_1} contains $t \cdot \sqrt{m}^{(k_1-2)/2}$ odd cycles of length k_1 (since fixing an edge t, one can complete it to a k_1 length cycle by choosing one vertex (out of the possible $\sqrt{m}$) in each of the sets R_i for $i \in [3, k_1]$). By choosing one odd cycle or star from every odd cycle and star gadgets in G, it holds that the number of copies of H_D in $G_{\vec{z}} \setminus G$ is at least $t \cdot m^{(k_1-2)/2} \cdot \prod_{i>1} \bar{\mathtt{c}}_i = \left\lfloor |T| \cdot \prod_{i>1} \bar{\mathtt{c}}_i \right\rfloor$. Observe that by the construction of G, the edges between the odd cycles and stars of different components agree with the non-decomposition edges of H_D. Hence, the number of copies of H_D in $G_{\vec{z}} \setminus G'$ is at least $\Omega(\bar{\mathtt{h}})$.

We now turn to the case that $\vec{z} = \vec{0}$. and argue that the graph $G \setminus G'$ contains less $o(\bar{\mathtt{h}})$ copies H_D. We deal separately the two potential cases due to constraint 5, that is, that either there is at least one star with $\bar{\mathtt{s}}_p > |H|\sqrt{m}^{p+1}$, or that that for all odd cycle components O_{k_i} for $k_i \leq k_1$, there are a few of them ($\bar{\mathtt{o}}_{k_i} \leq \sqrt{m}^{k-1}$). (Recall that this constraint is to prevent short cycle gadgets from creating too many copies of H_D within themselves.)

Assume first that there exists at least one star S_p in D with $\bar{\mathtt{s}}_p = \omega(m \cdot (\bar{\mathtt{o}}_{k_*})^{(p+1/k_*)})$, where recall that k_* is the index of the O_k component that maximizes $\bar{\mathtt{o}}_k^{1/k}$. Recall that by the construction of the motif H_D, S_p is connected to O_{k_1}. Also recall that in that case, the `few-cycles-gadget` is identical to the `cycle-gadget`, and it holds that a `cycle-gadget` can potentially create at most $k_i \cdot (\bar{\mathtt{o}}_{k_i})^{1/k_i} \cdot (\bar{\mathtt{o}}_{k_i})^{p/k_i} = k_i \cdot (\bar{\mathtt{o}}_{k_i})^{(p+1)/k_i}$ copies of S_p. Also,

for all other $S_{p_j} \in D$, at most $\sqrt{m}^{p+1}$ copies of S_{p_j} are created Hence, each `cycle-gadget` creates at most

$$(\bar{\mathtt{o}}_{k_i})^{(p+1)/k_i} \cdot \prod_{j\in[q]} (\bar{\mathtt{o}}_{k_i})^{k_j/k_i} \cdot \prod_{j\in[\ell], S_{p_j}\neq S_p} \sqrt{m}^{p_j+1} \leq (\bar{\mathtt{o}}_{k_i})^{(p+1)/k_i} \cdot \prod_{j\in[q]} \bar{\mathtt{o}}_{k_j} \cdot \prod_{j\in[\ell], S_{p_j}\neq S_p} \bar{\mathtt{s}}_{p_j} \ ,$$

where the last equality is due to constraint 4. Also, since $\bar{\mathtt{o}}_{k_1} \in [\sqrt{m}^{k_1-2}, \sqrt{m}^{k_1}]$, and $t \geq 1$, it holds that $t \cdot \sqrt{m}^{k-2} \geq \bar{\mathtt{o}}_{k_1}/m$. Hence,

$$h = t \cdot \sqrt{m}^{k-2} \cdot \prod_{i>1} \bar{\mathtt{o}}_{k_i} \cdot \prod_{j\in[\ell]} \bar{\mathtt{s}}_{p_j} \geq \frac{1}{m} \prod_{i\in[q]} \bar{\mathtt{o}}_{k_i} \cdot \prod_{j\in[\ell]} \bar{\mathtt{s}}_{p_j} = \frac{1}{m} \cdot \bar{\mathtt{s}}_p \cdot \prod_{i\in[q]} \bar{\mathtt{o}}_{k_i} \cdot \prod_{j\in[\ell], S_{p_j}\neq S_p} \bar{\mathtt{s}}_{p_j} \ .$$

Since $\bar{\mathtt{s}}_p = \omega(m \cdot (\bar{\mathtt{o}}_{k_i})^{(p+1)/k_i})$, it holds that the number of copies created by the `cycle-gadget` of O_{k_i} is $o(\bar{\mathtt{h}})$. Therefore, in that case the number of copies of H_D in $G \setminus G'$ is $o(\bar{\mathtt{h}})$.

In the case that there is no star S_p with sufficiently many copies as above, we have that constraint 5b holds. In that case, for every O_{k_i}, either (1) $k_i \leq k_1$, and so O_{k_i} has a `few-cycles-gadget` with a part R_1 consisting of a single vertex; or (2) $k_i > k_1$ and O_{k_i} has an `cycle-gadget`. In case (1), since the part R_1 of the `few-cycles-gadget` has a single vertex no copies of H_D can be created. In case (2), since $k_i > k_1$, no copies of odd length cycles of length k_1 are formed, and again no copies of H_D can be created. Also, no copies of H_D can be created by combining odd cycles of different gadgets, since each `few-cycles-gadget` can contribute at most one odd cycle, and `cycle-gadget` cannot contribute short cycles, and so at least one odd cycle of length $k_i < k_1$ will be missing.

Therefore, in both cases of constraint 5, the number of copies of H_D in $G \setminus G'$ is $o(\bar{\mathtt{h}})$, as claimed.

Now let $\mathcal{A}$ be any algorithm that samples returns a uniformly distributed copy of H_D. Then Alice and Bob can invoke $\mathcal{A}$ on the (implicit) graph $G_{\vec{z}}$ and whenever $\mathcal{A}$ performs a query, by the assumption of the lemma, Alice and Bob can communicate B bits to answer it. Alice and Bob repeat the above for 10 times. Let Q denote the number of queries each invocation of $\mathcal{A}$ performs. After $\mathcal{A}$ concludes all its runs, if $\mathcal{A}$ returns any copy of H_D from $G_{\vec{z}} \setminus G'$, then Alice declares that x and y intersect, and otherwise she declares they do not. Since the number of copies of H_D from $G_{\vec{z}} \setminus G'$ is at least $1/2$ of the number of copies in $G_{\vec{z}}$, each invocation of $\mathcal{A}$ should return a copy of H_D from $G_{\vec{z}} \setminus G'$ with probability at least $2/3$. Hence, the probability that $\vec{z} \neq \vec{0}$ and no copy from $G_{\vec{z}} \setminus G'$ is returned is at most $(2/3)^{10}$. Therefore, Alice and Bob can with high probability solve the t-Set-Disjointness instance using $O(Q \cdot B)$ bits of communication. By the $\Omega(m/t)$ expected communication lower bound for t-Set-Disjointness, it follows that $Q = \Omega(m/(t \cdot B))$. Since $t = \Theta(\bar{\mathtt{h}}/(m^{k_1/2-1} \cdot \prod_{i>1} \bar{\mathtt{c}}_i))$, we get an

$$\Omega\left(\frac{m}{t \cdot B}\right) = \Omega\left(\frac{m^{k_1/2} \cdot \prod_{i>1} \bar{\mathtt{c}}_i}{B \cdot \bar{\mathtt{h}}}\right)$$

lower bound, as claimed.

For the case that the minimum in the lower bound is due to the term m, we use the same proof, but with adjusted values of $|T|, t$ and the sizes of the sets in the `CC-gadget` of O_{k_1}. All other arguments remain the same. Recall that $\bar{\mathtt{h}} = \prod_i \bar{\mathtt{c}}_i/\alpha$, and so in this case we have that $\frac{m^{k_1/2}}{\bar{\mathtt{o}}_k} \cdot \alpha \geq m \Rightarrow \bar{\mathtt{o}}_k \leq \alpha \cdot m^{(k_1-2)/2}$. Let $\beta > 1$ be $\beta = \alpha m^{(k_1-2)/2}/\bar{\mathtt{o}}_{k_1} \Rightarrow \bar{\mathtt{o}}_k = \alpha \cdot (m/\beta)^{(k_1-2)/2}$. We change the `CC-gadget` that corresponds to O_{k_1} by changing the sizes

of its sets $R_3, \ldots, R_{k_1}$ to be of size $\sqrt{m}/\beta$ instead of $\sqrt{m}$. We now let

$$|T| = \bar{\mathtt{o}}_{k_1}/\alpha = (m/\beta)^{(k_1-2)/2} \qquad \text{and} \qquad t = \left\lfloor |T|/\sqrt{m}^{(k_1-2)/2} \right\rfloor = 1.$$

By the same arguments as for the previous case, we have that if $\vec{z} = 0$, then all copies of H_D are in G', and otherwise, $G_{\vec{z}} \setminus G'$ has $t \cdot (\sqrt{m}/\beta)^{k_1-2} \cdot \prod_{i>1} \bar{\mathtt{c}}_i = (\bar{\mathtt{o}}_{k_1}/\alpha) \cdot \prod_{i>1} \bar{\mathtt{c}}_i = \prod_i \bar{\mathtt{c}}_i/\alpha = \bar{\mathtt{h}}$ many copies of H_D. Therefore, the proof continues as before and we get a lower bound of $\Omega(m/B \cdot t) = \Omega(m/B)$ on the expected query complexity of any algorithm that returns a uniformly distributed copy of H_D. ◀

It remains to prove that queries on $G_{\vec{z}}$ can be answered by Alice efficiently.

▶ **Lemma 23.** *Alice can answer any query to $G_{\vec{z}}$ using $O(1)$ bits of communication between Alice and Bob. That is $B = O(1)$.*

Proof. We consider each of the possible queries.

Answering degree queries and uniform edge sample queries. Observe that all the vertices' degrees in the graph are set regardless of the (x, y) instance. Therefore, Alice knows the degree sequence and can produce a uniform edge sample and answer a degree query with zero communication.

Pair queries. Pair queries that include at most one vertex from the sets R_1, R_2, R'_1, R'_2, of the `CC-gadget` can be answered with zero communication. Pair queries (u, v) where say $u = r_1^i$ and $v = r_2^j$, are answered as follows. Bob sends to Alice the bit $y_{i,j}$. If the two bits intersect then the answer to the pair query is positive and otherwise, it is negative. Queries on other pairs with both endpoints in R_1, R_2, R'_1, R'_2 are answered similarly.

Answering i^{th} neighbor queries. First, any neighbor queries for vertices outside `CC-gadget` can be answered with zero communication. Let (v, j) be an j^{th} neighbor query for some v in the `CC-gadget`. If $v \notin R_1 \cup R_2$ or $j > \sqrt{m}$ then again the query can be answered with no communication. Therefore, assume without loss of generality that $v = r_1^i$ for some $r_1^i \in R_1$ and that $j \leq \sqrt{m}$. In this case Bob will send the bit y_{j+i} to Alice (recall that both Alice and Bob invoke the same algorithm using their shared randomness, so that the queries are known to both without communication). If $x_{i,i+k} \cdot y_{i,i+k} = 1$, then Alice answers $r_2^{i+j \mod \sqrt{m}}$. Otherwise, Alice answers $(r')_1^j$. Neighbor queries on vertices in R_2, R'_1 and R'_2 are answered similarly. ◀

Theorem 7 follow from Lemma 22 and Lemma 23.

5.4 From Theorem 7 to Theorem 6

▶ **Lemma 24.** *Theorem 6 follows from Theorem 7.*

Proof. Assume that Theorem 7 holds. Fix D to be a decomposition that contains at least one odd cycle component and a unique minimum odd length cycle, and fix n, m and a realizable value of DC. We would like to argue that there exists a motif H_D with optimal decomposition D, and a hard family of graphs $\mathcal{G}$ over n vertices, m edges and with decomposition cost DC, such that sampling a uniformly distributed copy of H in graphs uniformly chosen in $\mathcal{G}$ takes $\Omega(\min\{\text{DC}, m\})$. In order to do so we shall specify a set of good counts. We set the counts depending on the value of DC. If $\text{DC} \geq m$, then we set the odd cycle counts as follows: for every $O_{k_i} \in D$,

$$\begin{cases} \bar{\mathtt{o}}_k = \lceil m^{k/2}/\text{DC} \rceil & \text{if } k_i = k \\ o_{k_i} = \sqrt{m}^{k_i}, \text{ if } k_i > k & \text{if } k_i < k \end{cases}.$$

If $\text{DC} < m$, then we set the odd cycle counts as follows. Let $O_{k'}$ be the minimum length odd cycle in D. for every $O_{k_i} \in D$,

$$\begin{cases} \bar{\mathtt{o}}_{k_i} = \lceil m^{k'/2}/\text{DC} \rceil \text{ if } k_i = k' \\ o_{k_i} = \sqrt{m}^{k_i - 1} \qquad\qquad\qquad \text{if } k_i > k' \end{cases} .$$

Observe that by the assumption that there is only one odd cycle component of minimum length, indeed for every k_i, either $k_i = k'$ or $k_i > k'$. In both cases we also set $\bar{\mathtt{s}}_p = d_{\mathsf{avg}} \cdot n^p$ We also set $\bar{\mathtt{h}} = \prod_{i \in [r]} \bar{\mathtt{c}}_i$.

In order to be able to invoke Theorem 7, we argue that these counts are good, as defined in Definition 17. First, to see that the counts are realizable, consider a graph G which has a `few-cycles-gadget` for every $O_{k_i} \in D$ such that $k_i \leq k$, and a `cycle-gadget` for every $O_{k_i} \in D$ such that $k_i > k$. For every $S_p \in D$ we have a `star-gadget`. We let H_D be the components of D that are connected is some tree like manner, and we connect the gadgets of G by a complete bipartite graph between any two gadgets whose corresponding components in H_D are connected. It holds that the number of copies of H_D in G is $\bar{\mathtt{h}} = \pi \bar{\mathtt{c}}_i$. One can verify that in both cases of possible values of DC, the rest of the constraints of Definition 17 also hold.

Finally, in case that that $\text{DC} > m$, $\max_{i \in [r]} cost(C_i) = m^{k/2}/\bar{\mathtt{o}}_k = \Theta(\text{DC})$, and otherwise $\max_{i \in [r]} cost(C_i) = m^{k'/2}/\bar{\mathtt{o}}_{k'} = \Theta(\text{DC})$. Hence, we get that in both cases,

$$\text{DECOMP-COST}(G, H_D, D) = \max_{i \in [r]} cost(C_i) \cdot \frac{\prod_{i \in [r]} \bar{\mathtt{c}}_i}{\bar{\mathtt{h}}} = \Theta(\text{DC}).$$

Therefore, we can invoke Theorem 6, and the theorem follows. ◀

6 Acknowledgments

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.

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. URL: `http://drops.dagstuhl.de/opus/volltexte/2018/10099`, `doi:10.4230/LIPIcs.ITCS.2019.6`.

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

9 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.

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

11 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.

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

13 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.

14 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.

15 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.

16 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.

17 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.

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

19 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.

20 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.

21 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.

22 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.

23 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.

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

25 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.

26 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.

27 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.

28 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.

29 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.

30 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.

31 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.

32 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.

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

34 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.

35 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.

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

37 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.

38 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.

39 Jakub Tětek. Approximate triangle counting via sampling and fast matrix multiplication. *CoRR*, abs/2104.08501, 2021. URL: https://arxiv.org/abs/2104.08501, arXiv:2104.08501.

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

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

42 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 [21] and of Goldreich and Ron [24] on approximating the average degree in general graphs. Feige [21] 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 [24]. 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. [25] 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 [25] for this problem. In [16], Eden, Ron and Seshadhari again considered this problem, and presented improved bound for the case where the graph G has bounded arboricity. In [12, 15, 17], 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 [39], Tětek considers both the general and the augmented query models for approximately counting triangles in the super-linear regime. In [18], 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 [19, 13], 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 [36] 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 [13] 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 [14].

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 [9] later improved the first bound to $O(n/\sqrt{m}) \cdot \text{poly}(\log n)$ and also proved it to be optimal.