# Sampling Arbitrary Subgraphs Exactly Uniformly in Sublinear Time

**Hendrik Fichtenberger**
Department of Computer Science, TU Dortmund, Germany
hendrik.fichtenberger@tu-dortmund.de

**Mingze Gao**
Department of Computer Science, University of Sheffield, UK
noahgao0015@gmail.com

**Pan Peng**
Department of Computer Science, University of Sheffield, UK
p.peng@sheffield.ac.uk

## Abstract

We present a simple sublinear-time algorithm for sampling an arbitrary subgraph $H$ *exactly uniformly* from a graph $G$, to which the algorithm has access by performing the following types of queries: (1) uniform vertex queries, (2) degree queries, (3) neighbor queries, (4) pair queries and (5) edge sampling queries. The query complexity and running time of our algorithm are $\tilde{O}(\min\{m, \frac{m^{\rho(H)}}{\#H}\})$ and $\tilde{O}(\frac{m^{\rho(H)}}{\#H})$, respectively, where $\rho(H)$ is the fractional edge-cover of $H$ and $\#H$ is the number of copies of $H$ in $G$. For any clique on $r$ vertices, i.e., $H = K_r$, our algorithm is almost optimal as any algorithm that samples an $H$ from any distribution that has $\Omega(1)$ total probability mass on the set of all copies of $H$ must perform $\Omega(\min\{m, \frac{m^{\rho(H)}}{\#H\cdot(cr)^r}\})$ queries.

Together with the query and time complexities of the $(1 \pm \varepsilon)$-approximation algorithm for the number of subgraphs $H$ by Assadi et al. [3] and the lower bound by Eden and Rosenbaum [12] for approximately counting cliques, our results suggest that in our query model, approximately counting cliques is "equivalent to" exactly uniformly sampling cliques, in the sense that the query and time complexities of exactly uniform sampling and randomized approximate counting are within polylogarithmic factor of each other. This stands in interesting contrast to an analogous relation between approximate counting and almost uniformly sampling for self-reducible problems in the polynomial-time regime by Jerrum, Valiant and Vazirani [18].

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

**Keywords and phrases** Graph sampling, Graph algorithms, Sublinear-time algorithms

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2020.45

**Category** Track A: Algorithms, Complexity and Games

**Acknowledgements** We would like to thank the anonymous reviewers for their detailed comments. In particular, we would like to thank an anonymous reviewer for their suggestion to improve the presentation of the proof of Theorem 2 and their comment on applications, which we included as future work.

## 1 Introduction

"*Given a huge real graph, how can we derive a representative sample?*" is a first question asked by Leskove and Faloutsos in their seminal work on graph mining [20], which is motivated by the practical concern that most classical graph algorithms are too expensive for massive graphs (with millions or billions of vertices), and graph sampling seems essential for lifting the dilemma.


47th International Colloquium on Automata, Languages, and Programming (ICALP 2020).
Editors: Artur Czumaj, Anuj Dawar, and Emanuela Merelli; Article No. 45; pp. 45:1–45:13
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In this paper, we study the question of how to sample a subgraph $H$ uniformly at random from the set of all subgraphs that are isomorphic to $H$ contained in a large graph $G$ in *sublinear time*, where the algorithm is given query access to the graph $G$. That is, the algorithm only probes a small portion of the graph while still returning a sample with provable performance guarantee. Such a question is relevant for statistical reasons: we might need a few representative and unbiased motifs from a large network [21], or edge-colored subgraphs in a structured database [4], in a limited time. A subroutine for extracting a uniform sample of $H$ is also useful in streaming (e.g., [1]), parallel and distributed computing (e.g., [15]) and other randomized graph algorithms (e.g., [17]).

Currently, our understanding of the above question is still rather limited. Kaufman et al. gave the first algorithm for sampling an edge almost uniformly at random [19]. Eden and Rosenbaum gave a simpler and faster algorithm [13]. Both works considered the *general graph model*, where an algorithm is allowed to perform the following queries, where each query will be answered in constant time:

- **uniform vertex query:** the algorithm can sample a vertex uniformly at random;
- **degree query:** for any vertex $v$, the algorithm can query its degree $d_v$;
- **neighbor query:** for any vertex $v$ and index $i \leq d_v$, the algorithm can query the $i$-th neighbor of $v$;
- **pair query:** for any two vertices $u, v$, the algorithm can query if there is an edge between $u, v$.

In [13], Eden and Rosenbaum gave an algorithm that takes as input a graph with $n$ vertices and $m$ edges (where $m$ is unknown to the algorithm), uses $\tilde{O}(n/\sqrt{m})$ queries[1] in expectation and returns an edge $e$ that is sampled with probability $(1 \pm \varepsilon)/m$ (i.e., almost uniformly at random). This is almost optimal in the sense that any algorithm that samples an edge from an almost-uniform distribution requires $\Omega(n/\sqrt{m})$ queries. In their sublinear-time algorithm for approximately counting the number cliques [10] (see below), Eden, Ron and Seshadhri use a procedure to sample cliques incident to a suitable vertex subset $S$ almost uniformly at random. However, for an arbitrary subgraph $H$, it is still unclear how to obtain an almost uniform sample in sublinear time.

**Approximate counting in sublinear-time.** In contrast to sampling subgraphs (almost) uniformly at random, the very related line of research on approximate counting the number of subgraphs in sublinear time has made some remarkable progress in the past few years. Feige gave a $(2 + \varepsilon)$-approximation algorithm with $\tilde{O}(n/\sqrt{m})$ queries for the average degree, which is equivalent to estimating the number of edges, of a graph in the model that only uses vertex sampling and degree queries [14]. He also showed that any $(2 - o(1))$-approximation for the average degree using only vertex and degree queries requires $\Omega(n)$ queries. Goldreich and Ron then gave a $(1 + \varepsilon)$-approximation algorithm with $\tilde{O}(n/\sqrt{m})$ queries for the average degree in the model that allows vertex sampling, degree and neighbor queries [16].

Eden et al. recently gave the first sublinear-time algorithm for $(1 \pm \varepsilon)$-approximating the number of triangles [7]. Later, Eden, Ron and Seshadhri generalized it to $(1 \pm \varepsilon)$-approximating the number of $r$-cliques $K_r$ [10] in the general graph model that allows vertex sampling, degree, neighbor and vertex-pair queries. The query complexity and running time of their algorithms for $r$-clique $K_r$ counting are $\tilde{O}(\frac{n}{(\#K_r)^{1/3}} + \min\{m, \frac{m^{r/2}}{\#K_r}\})$ and $\tilde{O}(\frac{n}{(\#K_r)^{1/3}} + \frac{m^{r/2}}{\#K_r})$ respectively, for any $r \geq 3$, where $\#K_r$ is the number of copies of $K_r$ in $G$. Furthermore, in boths works it was proved that the query complexities of the respective algorithms are optimal up to polylogarithmic dependencies on $n, \epsilon$ and $r$.

[1] Throughout the paper, we use $\tilde{O}(\cdot)$ to suppress any dependencies on the parameter $\varepsilon$, the size of the corresponding subgraph $H$ and $\log(n)$-terms.

Later, Assadi et al. [3] gave a sublinear-time algorithm for $(1 \pm \varepsilon)$-approximating the number of copies of an arbitrary subgraph $H$ in the *augmented general graph model* [2]. That is, besides the aforementioned vertex sampling, degree, neighbor and pair queries, the algorithm is allowed to perform the following type of queries:

- **edge sampling query** the algorithm can sample an edge uniformly at random.

The algorithm in [3] uses $\tilde{O}(\min\{m, \frac{m^{\rho(H)}}{\#H}\})$ queries and $\tilde{O}(\frac{m^{\rho(H)}}{\#H})$ time, where $\rho(H)$ is the fractional edge-cover of $H$ and $\#H$ is the number of copies of $H$ in $G$. For the special case $H = K_r$, their algorithm performs $\tilde{O}(\min\{m, \frac{m^{r/2}}{\#K_r}\})$ queries and runs in $\tilde{O}(\frac{m^{r/2}}{\#K_r})$ time, which do not have the additive term $\frac{n}{(\#K_r)^{1/3}}$ in the query complexity and running time of the algorithms in [7, 10]. Eden and Rosenbaum provided simple proofs that most of the aforementioned results are nearly optimal in terms of their query complexities by reducing from communication complexity problems [12]. Further investigation of sampling an edge and estimating subgraphs in low arboricity graphs [8, 9] and approximately counting stars [2] has also been performed.

**Relation of approximate counting and almost uniform sampling.** One of our original motivations is to investigate the relation of approximate counting and almost uniform sampling in the sublinear-time regime. That is, we are interested in the question whether *in the sublinear-time regime, is almost uniform sampling "computationally comparable" to approximate counting, or is it strictly harder or easier, in terms of the query and/or time complexities for solving these two problems?* Indeed, in the polynomial-time regime, Jerrum, Valiant and Vazirani showed that for self-reducible problems (e.g., counting the number of perfect matchings of a graph), approximating counting is "equivalent to" almost uniform sampling [18], in the sense that the time complexities of almost uniform sampling and randomized approximate counting are within polynomial factor of each other. Such a result has been instrumental for the development of the area of approximate counting (e.g., [23]). It is natural to ask if similar relations between approximate counting and sampling hold in the sublinear-time regime.

## 1.1 Our Results

In this paper, we consider the problem of (almost) uniformly sampling a subgraph in the augmented general graph model. As mentioned above, this model has been studied in [2, 3], in which the authors find that "allowing edge-sample queries results in considerably simpler and more general algorithms for subgraph counting and is hence worth studying on its own". On the other hand, allowing edge sampling queries is also natural in models where neighbor queries are allowed, e.g., in the well-studied bounded-degree model and the general model: most graph representations that allow efficient neighbor queries (e.g., GEXF, GML or GraphML) store edges in linear data structures, which often allows efficient (nearly) uniformly sampling of edges. We refer to [3] for a deeper discussion on allowing edge sampling queries from both theoretical and practical perspectives.

We prove the following upper bound on sampling subgraphs (exactly) uniformly at random and provide a corresponding algorithm in Section 3.

▶ **Theorem 1.** *Let $H$ be an arbitrary subgraph. Let $G = (V, E)$ be a graph with $n$ vertices and $m$ edges. There exists an algorithm in the augmented general graph model that uses $\tilde{O}(\min\{m, \frac{m^{\rho(H)}}{\#H}\})$ queries in expectation, and with probability at least $2/3$, returns a copy of $H$, if $\#H > 0$. Each returned $H$ is sampled according to the uniform distribution over all copies of $H$ in $G$. The expected running time of the algorithm is $\tilde{O}(\frac{m^{\rho(H)}}{\#H})$.*

We stress that our sampler is an exactly uniform sampler, i.e., the returned $H$ is sampled from the uniform distribution, while to the best of our knowledge, the previous sublinear-time subgraph sampling algorithms are only *almost* uniform samplers. That is, they return an edge or a clique that is sampled from a distribution that is *close* to the corresponding uniform distribution. Indeed, it has been cast as an open question if it is possible to sample an edge exactly uniformly at random in the general graph model in [11].

Our algorithm is based on one idea from [3] (see also [4]) that uses the fractional edge cover to partition a subgraph $H$ into stars and odd cycles (i.e., Lemma 7). The authors of [3] also provided a scheme called *subgraph-sampler trees* for recursively sampling stars and odd cycles that compose $H$, while the resulting distribution is not (almost) uniform distribution. Instead, we show that one can sample stars and odd cycles by using rejection sampling in parallel (or, more precisely, sequentially but independently of each other) and check whether they form a copy of $H$.

To complement our algorithmic result, we give a lower bound on the query complexity for sampling a clique in sublinear time by using a simple reduction from [12]. We show the following theorem and present its proof in Section 4.

▶ **Theorem 2.** *Let $r \geq 3$ be an integer. Suppose $\mathcal{A}$ is an algorithm in the augmented general graph model that for any graph $G = (V, E)$ on $n$ vertices and $m$ edges returns an arbitrary $r$-clique $K_r$, if one exists; furthermore, each returned clique $K_r$ is sampled according to a distribution $\mathcal{D}$, such that the total probability mass of $\mathcal{D}$ on the set of all copies of $K_r$ is $\Omega(1)$. Then $\mathcal{A}$ requires $\Omega(\min\{m, \frac{m^{r/2}}{\#K_r \cdot (cr)^r}\})$ queries, for some absolute constant $c > 0$.*

Note that the above theorem gives a lower bound for sampling $K_r$ from almost every non-trivial distribution $\mathcal{D}$. In particular, it holds if $\#K_r > 0$ and $\mathcal{D}$ is a distribution that is only supported on the set of all copies of $K_r$, e.g., the (almost) uniform distribution on these copies. Together with the query and time complexities of the $(1 \pm \varepsilon)$-approximation algorithm for the number of subgraphs $H$ by Assadi et al. [3] and the lower bound by Eden and Rosenbaum [12] for approximately counting cliques, our Theorems 1 and 2 imply that in the augmented general graph model, *approximately* counting the number of cliques is equivalent to *exactly* sampling cliques in the sense that the query and time complexities of them are within a polylogarithmic factor of each other.

**Future Work.** Considering real-world applications, it would be interesting to relax the guarantees of the queries available to the algorithm. In particular, one may not be able to sample vertices or edges *exactly* uniformly at random, but only *approximately* uniformly. For example, there exist works that consider weaker query models in which even uniform vertex query is disallowed, and instead they sample vertices almost uniformly at random by performing random walks from some fixed vertex (see, e.g., [5, 6]). Implementing these changes in the model would result in a weaker guarantee for the distribution of sampled subgraphs in Theorem 1 but would be potentially more practical.

## 2 Preliminaries

Let $G = (V, E)$ be a simple graph with $|V| = n$ vertices and $|E| = m$ edges. For a vertex $v \in V$, we denote by $d_v$ the degree of the vertex, by $\Gamma_v$ the set of all the neighbors of $v$, and by $E_v$ the set of edges incident to $v$. We fix a total order on vertices denoted by $\prec$ as follows:

▶ **Definition 3.** *For any two vertices $u$ and $v$, we say that $u \prec v$ if $d_u < d_v$ or $d_u = d_v$ and $u$ appears before $v$ in the lexicographic order.*

For any two vertices, we denote by $\Gamma_{uv}$ the set of the shared neighbors of $u$ and $v$ that are larger than $u$ with respect to "$\prec$", i.e., $\Gamma_{uv} = \{w \mid w \in \Gamma_u \cap \Gamma_v \wedge u \prec w\}$. Sometimes, we view our graph $G = (V, E)$ as a directed graph $(V, \vec{E})$ by treating each undirected edge $e = \{u, v\} \in E$ as two directed edges $\vec{e}_1 = (u, v)$ and $\vec{e}_2 = (v, u)$. The following was proven in [7].

▶ **Lemma 4** ([7]). *For any vertex $v$, the number of neighbors $w$ of $v$ such that $v \prec w$ is at most $\sqrt{2m}$.*

Given a graph $H$, we say that a subgraph $H'$ of $G$ is a *copy* or an *instance* of $H$ if $H'$ is isomorphic to $H$. An isomorphism-preserving mapping from $H$ to a copy of $H$ in $G$ is called an *embedding* of $H$ in $G$.

**Rejection Sampling.** Given a starting distribution $\vec{p}$ and a target distribution $\vec{q}$ supported on a set $R$, let $M := \max_{a \in R} \frac{\vec{q}(a)}{\vec{p}(a)}$. Algorithm 1 is called *rejection sampling*.

**Algorithm 1** Rejection sampling with starting distribution $\vec{p}$ and target distribution $\vec{q}$.

```
procedure RejectionSampling(p⃗, q⃗)
    M ← max_{a∈R} q⃗(a)/p⃗(a)
    while true do
        sample a from p⃗.
        sample a number t ∈ [0, 1] uniformly at random.
        if t ≤ q⃗(a)/(M·p⃗(a)) then
            return a
```

Observe that when the algorithm terminates, the probability that $a$ is returned is $\vec{q}(a)$ for every $a \in R$. The following lemma is known.

▶ **Lemma 5** ([22]). *The expected number of iterations of* RejectionSampling*($\vec{p}, \vec{q}$) is $M$.*

**Edge Cover and Graph Decomposition.** We use the following definition of the fractional edge cover of a graph and a decomposition result based on it by Assadi et al. [3].

▶ **Definition 6** (Fractional Edge-Cover Number). *A fractional edge-cover of $H(V_H, E_H)$ is a mapping $\psi : E_H \to [0, 1]$ such that for each vertex $v \in V_H$, $\sum_{e \in E_H, v \in e} \psi(e) \geq 1$. The fractional edge-cover number $\rho(H)$ of $H$ is the minimum value of $\sum_{e \in E_H} \psi(e)$ among all fractional edge-covers $\psi$.*

Let $C_k$ denote the cycle of length $k$. Let $S_k$ denote a star with $k$ petals, i.e., $S_k = (\{u, v_1, \ldots, v_k\}, \cup_{i \in [k]} \{u, v_k\})$. Let $K_k$ denote a clique on $k$ vertices. It is known that $\rho(C_{2k+1}) = k + 1/2$, $\rho(S_k) = k$ and $\rho(K_k) = k/2$.

▶ **Lemma 7** ([3]). *Any subgraph $H$ can be decomposed into a collection of vertex-disjoint odd cycles $\overline{C_1}, \ldots, \overline{C_o}$ and star graphs $\overline{S_1}, \ldots, \overline{S_s}$ such that*

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

By a result of Atserias, Grohe and Marx [4], the number of instances of $H$ in a graph $G$ with $m$ edges is $O(m^{\rho(H)})$.

# 3 Sampling an Arbitrary Subgraph $H$

In this section, we present sampling algorithms for odd cycles and stars and show how to combine them to obtain a sampling algorithm for arbitrary subgraphs. Note that we do not need to know the exact number of edges $m$ to run our algorithm; it is sufficient to have a constant approximation $\hat{m}$ of $m$ so that $m \leq \hat{m} \leq cm$ for some $c > 1$. Such an approximation can be obtained by using the algorithm from [14, 16]. This increases the query complexity only by a constant factor. For the sake of simplicity, we will continue to use $m$ in the following.

## 3.1 Sampling an Odd-Length Cycle

We describe our algorithm SampleOddCycle for sampling a uniformly random odd-length $k$-cycle. For any instance of $C_{2k+1}$ in the input graph, our goal is to guarantee that it will be sampled with probability $\frac{1}{m^{k+1/2}}$. Let $e_1, \dots, e_{2k+1}$ be a sequence of edges that represents a cycle of length $2k+1$. While we can use edge sampling to sample every second edge of the first $2k$ edges sequentially, i.e., $e_1, e_3, \dots, e_{2k-1}$, and query the edges inbetween, i.e., $e_2, \dots, e_{2k-2}$, by vertex pair queries, we use a different strategy to sample $e_{2k}$ and $e_{2k+1}$. Let $\{u, v\} = e_1$. If $u$ has low degree, i.e., $d_u \leq \sqrt{2m}$, we can afford to sample each neighbor of $u$ with probability $1/\sqrt{2m}$ and fail if no neighbor is sampled. In particular, we need that a distinguished neighbor $x_1$ of $u$ is sampled with probability at least $1/\sqrt{2m}$. However, if $d_u \geq \sqrt{2m}$, this is too costly. Instead, we invoke rejection sampling with the following starting distribution and target distribution.

▶ **Definition 8.** *Let $u, v$ be two vertices such that $d_u > \sqrt{2m}$. Let $\vec{p}_u$ be a (starting) distribution with support $\Gamma_u$ such that:*

$$\vec{p}_u(w) = \frac{1}{d_u}, \quad w \in \Gamma_u \tag{1}$$

*Let $\vec{q}_u$ be a (target) distribution with support $\Gamma_u$ such that:*

$$\vec{q}_u(w) = \begin{cases} \frac{1}{\sqrt{2m}}, & w \in \Gamma_{uv} \\ \left(1 - \frac{|\Gamma_{uv}|}{\sqrt{2m}}\right) \cdot \frac{1}{d_u - |\Gamma_{uv}|}, & w \notin \Gamma_{uv} \end{cases} \tag{2}$$

We note that by Lemma 4, it always holds that $|\Gamma_{uv}| \leq \sqrt{2m}$. Furthermore,

$$\begin{aligned} \sum_{w \in \Gamma_u} \vec{q}_u(w) &= \sum_{w \in \Gamma_{uv}} \frac{1}{\sqrt{2m}} + \sum_{w \notin \Gamma_{uv}} \left(1 - \frac{|\Gamma_{uv}|}{\sqrt{2m}}\right) \cdot \frac{1}{d_u - |\Gamma_{uv}|} \\ &= \frac{|\Gamma_{uv}|}{\sqrt{2m}} + (d_u - |\Gamma_{uv}|)\left(1 - \frac{|\Gamma_{uv}|}{\sqrt{2m}}\right) \cdot \frac{1}{d_u - |\Gamma_{uv}|} = 1. \end{aligned}$$

Thus the distribution $\vec{q}_u$ is well-defined. Let $M_u = \max_{w \in \Gamma_u} \frac{\vec{q}_u(w)}{\vec{p}_u(w)}$ (as in Algorithm 1). Then, $M_u$ is bounded as follows.

▶ **Lemma 9.** *Let $M_u$ be defined as above. Recall that $d_u > \sqrt{2m}$. Then $M_u = \frac{d_u}{\sqrt{2m}}$.*

**Proof.** If $w \in \Gamma_{uv}$, we have that $\frac{\vec{q}_u(w)}{\vec{p}_u(w)} = \frac{d_u}{\sqrt{2m}}$. If $w \notin \Gamma_{uv}$, we have that

$$\frac{\vec{q}_u(w)}{\vec{p}_u(w)} = \frac{d_u(1 - \frac{|\Gamma_{uv}|}{\sqrt{2m}})}{d_u - |\Gamma_{uv}|} = \frac{d_u(\sqrt{2m} - |\Gamma_{uv}|)}{\sqrt{2m}(d_u - |\Gamma_{uv}|)} \leq \frac{d_u}{\sqrt{2m}}, \tag{3}$$

where the last inequality uses the fact that $d_u > \sqrt{2m}$. ◀

**Algorithm 2** Sampling a wedge.

**procedure** SampleWedge($G, u, v$)
  **if** $d_u \leq \sqrt{2m}$ **then**
    sample a number $i \in \{1, \dots \sqrt{2m}\}$ uniformly at random
    **if** $i > d_u$ **then**
      **return Fail**
    $w \leftarrow i^{th}$ neighbor of $u$
  **else**
    $w \leftarrow$ RejectionSampling($\vec{p}_u, \vec{q}_u$) ▷ see Definition 8
  **return** $w$

**Algorithm 3** Sampling a cycle of length $2k+1$.

**procedure** SampleOddCycle($G, 2k+1$)
  sample $k$ directed edges $(u_1, v_1), \dots, (u_k, v_k)$ u.a.r. and i.i.d.
  **if** $u_1, v_1, \dots, u_k, v_k$ is a path of length $2k-1$ and $u_1 \prec v_1$, $\forall i > 1 : u_1 \prec u_i, v_i$ **then**
    **if** SampleWedge($G, u_1, v_k$) returns $w$ and $w \prec v_1$ **then**
      **return** $\{(u_1, v_1), \dots, (u_k, v_k)\} \cup \{(v_k, w), (w, u_1)\}$
  **return Fail**

As there exists a linear number of automorphisms for every cycle, it is crucial in our algorithm to define a unique embedding based on the order of vertices for every instance of a $k$-cycle. Otherwise, bounding the probability that an instance is sampled *exactly* uniformly is hard as some instance might be sampled less likely because, e.g., its edges participate in many overlapping cycles. We take care of this by enforcing that only uniquely defined embeddings are sampled in SampleOddCycle. In particular, we sample $k$ directed edges $(u_1, v_1), \dots, (u_k, v_k)$ independently and uniformly at random and require that (i) they induce a path $u_1, v_1, u_2, \dots, v_k$ and (ii) for the first edge $(u_1, v_1)$, $u_1$ is the smallest vertex according to the order "$\prec$" among all $u_i, v_i, i \geq 1$. Then, we call SampleWedge on the two ends $u_1, v_k$ of this path to close a cycle and define an orientation of this cycle by requiring that $w \prec v_1$, where for $(v_k, w) = e_{2k+1}$. If any of these requirements is not met, we have not sampled the uniquely defined embedding we are looking for, and the algorithm fails.

▶ **Lemma 10.** *For any instance of an odd cycle $C_{2k+1}$ in $G$, the probability that it will be returned by* SampleOddCycle($G, 2k+1$) *is* $\frac{1}{(2m)^{k+1/2}}$.

**Proof.** Let $\mathcal{C}_{2k+1}$ be any instance of a cycle of odd length $2k+1$ in $G$. Let $x_0$ be the smallest vertex on $\mathcal{C}_{2k+1}$ according to the total order "$\prec$". Let $x_1, x_{2k}$ be the two neighbors of $x_0$ on $\mathcal{C}_{2k+1}$ such that $x_1 \prec x_{2k}$. Then, we let $x_i$ denote the vertices on $\mathcal{C}_{2k+1}$ such that $(x_i, x_{i+1}) \in E(\mathcal{C}_{2k+1})$ for $0 \leq i \leq 2k-1$ and $(x_{2k}, x_0) \in E(\mathcal{C}_{2k+1})$. Note that for any $\mathcal{C}_{2k+1}$, there is a *unique* way of mapping its vertices to $x_i$, for $0 \leq i \leq 2k$. Thus, SampleOddCycle returns $\mathcal{C}_{2k+1}$ if and only if

1. $u_1 = x_0$ and $v_1 = x_{2k}$;
2. $u_i = x_{2k-2i+3}$ and $v_i = x_{2k-2i+2}$ for $2 \leq i \leq k$;
3. SampleWedge($G, u_1, v_k$) returns $x_1$.

Event 1 occurs with probability $1/(2m)$, and event 2 occurs with probability $1/(2m)^{k-1}$, as each directed edge is sampled with probability $1/(2m)$.

Now we bound the probability of event 3. In the call to SampleWedge, let $u := u_1$ and $v := v_k$, which satisfies that $u \prec v$. We first note that if $d_u < \sqrt{2m}$ in SampleWedge$(G, u_1, v_k)$, then the vertex $x_1$ will be sampled with probability $1/\sqrt{2m}$. Now we consider the case that $d_u \geq \sqrt{2m}$. Then, RejectionSampling$(\vec{p}_u, \vec{q}_u)$ will return $x_1$ with probability $\vec{q}_{u_1}(x_1) = \frac{1}{\sqrt{2m}}$, as $x_1$ is a common neighbor of $u_1, v_k$ and $u_1 \prec x_1$. Thus in both cases, the probability that event 3 occurs is $\frac{1}{\sqrt{2m}}$. Therefore, the probability that SampleOddCycle returns $\mathcal{C}_{2k+1}$ is $\frac{1}{\sqrt{2m}} \cdot \frac{1}{2m} \cdot (\frac{1}{2m})^{k-1} = \frac{1}{(2m)^{k+1/2}}$. ◀

## 3.2 Sampling a Star

Similarly to odd cycles, we observe that every $k$-star admits an exponential number of automorphisms. Therefore, we enforce a unique embedding of every instance of a $k$-star in our sampling algorithm SampleStar. Let $e_1, \ldots, e_k$ be the petals of an instance of a $k$-star. We sample $e_1, \ldots, e_k$ sequentially. If these edges form a star, we output it only if the leaves where sampled in ascending order with respect to "$\prec$".

**Algorithm 4** Sampling a star with $k$ petals.

1: **procedure** SampleStar$(G, k)$
2: Sequentially sample $k$ directed edges $\{(u_1, v_1), \ldots, (u_k, v_k)\}$ u.a.r. and i.i.d.
3: **if** $u_1 = u_2 = \ldots = u_k$ and $v_1 \prec v_2 \prec \ldots \prec v_k$ **then**
4: **return** $(u_1, v_1, \ldots, v_k)$
5: **return Fail**

▶ **Lemma 11.** *For any instance of a $k$-star $S_k$ in $G$, the probability that it will be returned by the algorithm SampleStar$(G, k)$ is $\frac{1}{(2m)^k}$.*

**Proof.** Consider any instance of $S_k$ with root $x$ and petals $y_1, \ldots, y_k$ such that $y_1 \prec \ldots y_k$. Note that it will be returned by SampleStar if and only if all the directed edges $(x, y_1), \ldots, (x, y_k)$ are sequentially sampled, which occurs with probability $1/(2m)^k$. ◀

## 3.3 Sampling $H$

Let $H$ be a subgraph. It can be decomposed into collections of $o$ odd cycles $\overline{C_i}$ and $s$ stars $\overline{S_j}$ as given in Lemma 7. We say that $H$ has a (decomposition) *type* $\overline{T} = \{\overline{C_1}, \ldots, \overline{C_o}, \overline{S_1}, \ldots, \overline{S_s}\}$.

▶ **Definition 12.** *Given a graph $G$, for each potential* instance *$\mathcal{H}$ of $H$, we say that $\mathcal{H}$ can be decomposed into* configurations *$\mathcal{T} = \{\mathcal{C}_1, \ldots, \mathcal{C}_o, \mathcal{S}_1, \ldots, \mathcal{S}_s\}$ with respect to type $\overline{T} = \{\overline{C_1}, \ldots, \overline{C_o}, \overline{S_1}, \ldots, \overline{S_s}\}$, if*

1. *$\mathcal{C}_i \cong \overline{C_i}$ for any $1 \leq i \leq o$, and $\mathcal{S}_j \cong \overline{S}_j$, for any $1 \leq i \leq s$*
2. *all the remaining edges of $H$ between vertices specified in $\mathcal{T}$ all are present in $G$.*

*We let $f_{\overline{T}}(H)$ denote the number of all possible configurations $\mathcal{T}$ into which $H$ can be decomposed with respect to $\overline{T}$.*

**Algorithm 5** Sampling a copy of subgraph $H$.

**procedure** SAMPLESUBGRAPH$(G, H)$
    Let $\overline{T} = \{\overline{C_1}, \ldots, \overline{C_o}, \overline{S_1}, \ldots, \overline{S_s}\}$ denote a (decomposition) type of $H$.
    **for all** $i = 1 \ldots o$ **do**
        **if** SAMPLEODDCYCLE$(G, |E(\overline{C}_i)|)$ returns a cycle $\mathcal{C}$ **then**
            $\mathcal{C}_i \leftarrow \mathcal{C}$
        **else**
            **return Fail**
    **for all** $j = 1 \ldots s$ **do**
        **if** SAMPLESTAR$(G, |V(\overline{S}_j)| - 1)$ returns a star $\mathcal{S}$ **then**
            $\mathcal{S}_j \leftarrow \mathcal{S}$
        **else**
            **return Fail**
    Query all edges $(\bigcup_{i \in [o]} V(\mathcal{C}_i) \cup \bigcup_{j \in [s]} V(\mathcal{S}_j))^2$
    **if** $S := (\mathcal{C}_1, \ldots, \mathcal{C}_o, \mathcal{S}_1, \ldots, \mathcal{S}_s)$ forms a copy of $H$ **then**
        flip a coin and with probability $\frac{1}{f_{\overline{T}}(H)}$: **return** $S$
    **return Fail**

▶ **Lemma 13.** *For any instance of a subgraph $H$ in $G$, the probability that it will be returned by the algorithm* SAMPLESUBGRAPH$(G, H)$ *is* $\frac{1}{(2m)^{\rho(H)}}$.

**Proof.** For any instance $\mathcal{H}$ of $H$ in $G$, and any configuration $\mathcal{T} = \{\mathcal{C}_1, \ldots, \mathcal{C}_O, \mathcal{S}_1, \ldots, \mathcal{S}_s\}$ of $\mathcal{H}$ with respect to $\overline{T}$, $\mathcal{H}$ will be returned by SAMPLESUBGRAPH$(G, H)$ if and only if

1. $\mathcal{C}_i$ is returned in Algorithm 5 for each $1 \le i \le o$, and $\mathcal{S}_j$ is returned in Algorithm 5 for any $1 \le j \le s$;
2. the configuration is returned with probability $\frac{1}{f_{\overline{T}}(H)}$ in Algorithm 5.

By Lemma 10, each $\mathcal{C}_i$ will be returned with probability $\frac{1}{(2m)^{|E(\overline{C_i})|/2}} = \frac{1}{(2m)^{\rho(\overline{C_i})}}$. By Lemma 11 each $\mathcal{S}_j$ will be returned with probability $\frac{1}{(2m)^{|V(\overline{S_j})|-1}} = \frac{1}{(2m)^{\rho(\overline{S_j})}}$. Thus, $\mathcal{T}$ will be returned with probability

$$\prod_{i=1}^{o} \frac{1}{(2m)^{\rho(\overline{C_i})}} \cdot \prod_{j=1}^{s} \frac{1}{(2m)^{\rho(\overline{S_j})}} \cdot \frac{1}{f_{\overline{T}}(H)} = \frac{1}{(2m)^{\rho(H)}} \cdot \frac{1}{f_{\overline{T}}(H)}.$$

Finally, since there are $f_{\overline{T}}(H)$ configurations of $\mathcal{H}$ with respect to $\overline{T}$, the instance will be returned with probability $f_{\overline{T}}(H) \cdot \frac{1}{(2m)^{\rho(H)}} \cdot \frac{1}{f_{\overline{T}}(H)} = \frac{1}{(2m)^{\rho(H)}}$. ◀

## 3.4 The Final Sampler

Let $X_H$ be an estimate of $\#H$. Such an estimate can be obtained by, e.g., the subgraph counting algorithm of Assadi et al. [3] in expected time $\tilde{O}(m^{\rho(H)}/\#H)$. We show that by sufficiently many calls to SAMPLESUBGRAPH, we can obtain a uniformly random sample of an instance of $H$ with constant probability.

**Algorithm 6** Sampling a copy of subgraph $H$ uniformly at random.

---

**procedure** SAMPLESUBGRAPHUNIFORMLY$(G, H, X_H)$
  **for all** $j = 1, \ldots, q = 10 \cdot (2m)^{\rho(H)}/X_H$ **do**
    Invoke SAMPLESUBGRAPH$(G, H)$
    **if** a subgraph $H$ is returned **then return** $H$
  **return Fail**

---

▶ **Lemma 14.** *If $\#H \le X_H \le 2\#H$, then Algorithm SAMPLESUBGRAPHUNIFORMLY$(G, H, X_H)$ returns a copy $H$ with probability at least $2/3$. The distribution induced by the algorithm is (exactly) uniform over the set of all instances of $H$ in $G$.*

**Proof.** Since $\#H \le X_H \le 2\#H$, the probability that no instance of $H$ is returned in $q = 10 \cdot (2m)^{\rho(H)}/X_H$ invocations is at most

$$\left(1 - \frac{\#H}{(2m)^{\rho(H)}}\right)^q \le e^{-\frac{\#H}{(2m)^{\rho(H)}} \cdot q} < \frac{1}{3}$$

by Lemma 13. Let $\mathcal{H}$ be an instance of $H$. By Lemma 13, the probability that SAMPLESUBGRAPH$(H)$ returns $\mathcal{H}$ is $\frac{1}{(2m)^{\rho(H)}}$. Thus, the probability that SAMPLESUBGRAPHUNIFORMLY$(G, H)$ successfully output an instance of $H$ is

$$\frac{\#H}{(2m)^{\rho(H)}}.$$

Conditioned on the event that SAMPLESUBGRAPHUNIFORMLY$(G, H)$ succeeds, the probability that any specific instance $\mathcal{H}$ will be returned is

$$p_{\mathcal{H}} = \frac{\frac{1}{(2m)^{\rho(H)}}}{\frac{\#H}{(2m)^{\rho(H)}}} = \frac{1}{\#H}.$$

That is, with probability at least $\frac{2}{3}$, an instance $\mathcal{H}$ is sampled from the uniform distribution over all the instances of $H$ in $G$. ◀

Finally, we prove the expected query and time complexity of SAMPLESUBGRAPHUNIFORMLY.

▶ **Lemma 15.** *The expected query and time complexity of SAMPLESUBGRAPHUNIFORMLY$(G, H, X_H)$ is $O(m^{\rho(H)}/X_H)$.*

**Proof.** We analyze the query complexity of SAMPLEODDCYCLE$(G, 2k+1)$ for $d_{u_1} < \sqrt{2m}$ and $d_{u_1} \ge \sqrt{2m}$ separately. The probability that $d_{u_1} < \sqrt{2m}$ is at most 1, and the query complexity is at most $O(1)$ in this case.

To bound the probability that SAMPLEWEDGE$(G, u_1, v_k)$ is invoked such that $d_{u_1} \ge \sqrt{2m}$, recall that sampling an edge uniformly at random is equivalent to sampling a vertex proportionally to its degree and selecting a neighbor uniformly at random. The probability to sample a neighbor $x$ of $u_1$ is $1/d_{u_1}$. There at most $2m/\sqrt{2m} = \sqrt{2m}$ vertices that have degree at least $\sqrt{2m}$, so the probability that a uniformly random neighbor $v_1$ of $u_1$ has degree at least $\sqrt{2m}$ is at most $\sqrt{2m}/d_{u_1}$. Therefore, the probability that $v_1$ has degree at least $\sqrt{2m}$, which is implied by the check $u_1 \prec v_1$ in line 3, is bounded by $\sqrt{2m}/d_{u_1}$. By Lemmas 5 and 9, the expected number of queries in SAMPLEWEDGE$(G, u_1, v_k)$ is at most $M \le d_{u_1}/\sqrt{2m}$ if $d_{u_1} \ge \sqrt{2m}$. Thus, the expected query complexity of SAMPLEODDCYCLE$(G, 2k+1)$ is bounded by

$$\sum_{\substack{u_1 \in V\\ d_{u_1} < \sqrt{2m}}} \frac{d_{u_1}}{2m} \cdot O(1) + \sum_{\substack{u_1 \in V\\ d_{u_1} \geq \sqrt{2m}}} \frac{d_{u_1}}{2m} \cdot \frac{\sqrt{2m}}{d_{u_1}} \cdot \frac{d_{u_1}}{\sqrt{2m}} \leq O(1) + \sum_{\substack{u_1 \in V\\ d_{u_1} \geq \sqrt{2m}}} \frac{d_{u_1}}{2m} = O(1).$$

The expected query complexity of SampleStar$(G, k)$ is bounded by $k \in O(1)$. It follows that the expected query complexity of SampleSubgraph$(G, H)$ is at most $(o + s + |H|^2) \cdot O(1) \subseteq |H| \cdot O(1)$. The expected query complexity of SampleSubgraphUniformly$(G, H)$ is $O((2m)^{\rho(H)}/X_H \cdot |H|^2) = \tilde{O}((2m)^{\rho(H)}/X_H)$. To bound the expected running time, we observe that every loop in our algorithm issues at least one query, and we only perform isomorphism checks on subgraphs of constant size. Thus the running time is still $\tilde{O}((2m)^{\rho(H)}/X_H)$. ◀

The proof of Theorem 1 follows almost directly from Lemmas 14 and 15.

**Proof of Theorem 1.** For the case that $m \geq m^{\rho(H)}/\#H$, the claim follows from Lemmas 14 and 15. If $m < m^{\rho(H)}/\#H$, we can query the whole graph, which requires $O(m)$ degree and neighbor queries, store the graph and answer the queries of the algorithm from this internal memory. ◀

## 4 Proof of Theorem 2

In this section, we give the proof of Theorem 2, which follows by adapting the proofs for the lower bounds on the query complexity for approximate counting subgraphs given by Eden and Rosenbaum [12].

▶ **Theorem 16** (see Theorems 4.7 and B.1 in [12]). *For any choices of $n, m, r, c_r > 0$, there exist families of graphs with $n$ vertices and $m$ edges, $\mathcal{F}_0$ and $\mathcal{F}_1$, such that*

- *all graphs in $\mathcal{F}_0$ are $K_r$-free,*
- *all graphs in $\mathcal{F}_1$ contain at least $c_r$ copies of $K_r$,*
- *and any algorithm in the augmented general graph model that distinguishes a graph $G \in \mathcal{F}_0$ from $G \in \mathcal{F}_1$ with probability $\Omega(1)$ requires $\Omega(\min\{m, m^{r/2}/c_r(cr)^r\})$ queries for some constant $c > 0$.*

Now we prove our Theorem 2.

**Proof of Theorem 2.** Let $\mathcal{A}$ be an algorithm that for any graph $G = (V, E)$ on $n$ vertices and $m$ edges returns an arbitrary $r$-clique $K_r$, if one exists; and each $K_r$ is sampled according to $\mathcal{D}$, using $f(m, r, \#K_r) \in o(\min\{m, \frac{m^{r/2}}{\#K_r \cdot (cr)^r}\})$ neighbor, degree, pair and edge sampling queries.

Let $n, m, c_r > 0$ and let $\mathcal{F}_0, \mathcal{F}_1$ be the families from Theorem 16. Consider the following algorithm $\mathcal{A}'$: run $\mathcal{A}$ on a graph from $\mathcal{F}_0 \cup \mathcal{F}_1$ and terminate $\mathcal{A}$ if it did not produce a $K_r$ after $f(m, r, c_r)$ queries. If it output a clique, $\mathcal{A}'$ claims that $G \in \mathcal{F}_1$, otherwise it claims that $G \in \mathcal{F}_0$. By the assumption, $\mathcal{A}$ returns a clique after at most $f(m, r, c_r)$ queries with probability $\Omega(1)$ if $G \in \mathcal{F}_1$ because then $G$ contains at least $c_r$ copies of $K_r$ and the probability mass of $\mathcal{D}$ on the set of all copies of $K_r$ is $\Omega(1)$. Otherwise, $G \in \mathcal{F}_0$, which implies that $G$ contains no triangle. Therefore, $\mathcal{A}$ cannot output a triangle from $G$.

It follows that $\mathcal{A}'$ can distinguish $\mathcal{F}_0$ and $\mathcal{F}_1$, which is a contradiction to Theorem 16. ◀

## References


1 Nesreen K Ahmed, Nick Duffield, Theodore L Willke, and Ryan A Rossi. On sampling from massive graph streams. *Proceedings of the VLDB Endowment*, 10(11), 2017. doi:10.14778/3137628.3137651.

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*, 2017. doi:10.1007/s00453-017-0287-3.

3 Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A Simple Sublinear-Time Algorithm for Counting Arbitrary Subgraphs via Edge Sampling. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 124. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2019.6.

4 Albert Atserias, Martin Grohe, and Dániel Marx. Size bounds and query plans for relational joins. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2008. doi:10.1109/FOCS.2008.43.

5 Anna Ben-Hamou, Roberto I Oliveira, and Yuval Peres. Estimating graph parameters via random walks with restarts. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2018. doi:10.1137/1.9781611975031.111.

6 Flavio Chiericetti, Anirban Dasgupta, Ravi Kumar, Silvio Lattanzi, and Tamás Sarlós. On sampling nodes in a network. In *Proceedings of the 25th International Conference on World Wide Web*, 2016. doi:10.1145/2872427.2883045.

7 Talya Eden, Amit Levi, Dana Ron, and C Seshadhri. Approximately counting triangles in sublinear time. *SIAM Journal on Computing*, 46(5), 2017. doi:10.1137/15M1054389.

8 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)*, volume 132. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.ICALP.2019.52.

9 Talya Eden, Dana Ron, and C. Seshadhri. Faster sublinear approximations of k-cliques for low arboricity graphs. *CoRR*, abs/1811.04425, 2018. arXiv:1811.04425.

10 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. doi:10.1145/3188745.3188810.

11 Talya Eden and Will Rosenbaum. On sampling edges almost uniformly. *arXiv preprint*, 2017. arXiv:1706.09748.

12 Talya Eden and Will Rosenbaum. Lower Bounds for Approximating Graph Parameters via Communication Complexity. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 116. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.11.

13 Talya Eden and Will Rosenbaum. On Sampling Edges Almost Uniformly. In *1st Symposium on Simplicity in Algorithms (SOSA)*, volume 61. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/OASIcs.SOSA.2018.7.

14 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), 2006. doi:10.1137/S0097539704447304.

15 Weiming Feng, Yuxin Sun, and Yitong Yin. What can be sampled locally? In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2017. doi:10.1007/s00446-018-0332-8.

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

17 Pili Hu and Wing Cheong Lau. A survey and taxonomy of graph sampling. *arXiv preprint*, 2013. arXiv:1308.5865.

**18** Mark R Jerrum, Leslie G Valiant, and Vijay V Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43, 1986. doi:10.5555/11534.11537.

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

**20** Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006. doi:10.1145/1150402.1150479.

**21** R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594), 2002. doi:10.1126/science.298.5594.824.

**22** Von Neumann. Various techniques used in connection with random digits. *National Bureau of Standards, Applied Math Series*, 12, 1950.

**23** Alistair Sinclair and Mark Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation*, 82(1), 1989. doi:10.1016/0890-5401(89)90067-9.