Sampling Multiple Edges Efficiently

Talya Eden
CSAIL at MIT, USA

Saleet Mossel
CSAIL at MIT, USA

Ronitt Rubinfeld
CSAIL at MIT, USA

Abstract

We present a sublinear time algorithm that allows one to sample multiple edges from a distribution that is pointwise ϵ-close to the uniform distribution, in an *amortized-efficient* fashion. We consider the adjacency list query model, where access to a graph G is given via degree and neighbor queries.

The problem of sampling a single edge in this model has been raised by Eden and Rosenbaum (SOSA 18). Let n and m denote the number of vertices and edges of G, respectively. Eden and Rosenbaum provided upper and lower bounds of $\Theta^*(n/\sqrt{m})$ for sampling a single edge in general graphs (where $O^*(\cdot)$ suppresses $\text{poly}(1/\epsilon)$ and $\text{poly}(\log n)$ dependencies). We ask whether the query complexity lower bound for sampling a single edge can be circumvented when multiple samples are required. That is, can we get an improved amortized per-sample cost if we allow a preprocessing phase? We answer in the affirmative.

We present an algorithm that, if one knows the number of required samples q in advance, has an overall cost that is sublinear in q, namely, $O^*(\sqrt{q} \cdot (n/\sqrt{m}))$, which is strictly preferable to $O^*(q \cdot (n/\sqrt{m}))$ cost resulting from q invocations of the algorithm by Eden and Rosenbaum.

Subsequent to a preliminary version of this work, Tětek and Thorup (arXiv, preprint) proved that this bound is essentially optimal.

2012 ACM Subject Classification Theory of computation → Sketching and sampling

Keywords and phrases Sampling edges, graph algorithm, sublinear algorithms

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Category RANDOM

Related Version *Full Version*: https://arxiv.org/abs/2008.08032

Funding *Talya Eden*: This work was supported by the NSF Grant CCF-1740751, the Eric and Wendy Schmidt Fund, and Ben-Gurion University.
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.

1 Introduction

The ability to select edges uniformly at random in a large graph or network, namely *edge sampling*, is an important primitive, interesting both from a theoretical perspective in various models of computation (e.g., [19, 2, 3, 1, 13, 12, 7, 4, 15]), and from a practical perspective in the study of real-world networks (e.g., [20, 22, 31, 6, 27]). We consider the task of outputting edges from a distribution that is close to uniform; more precisely, the output distribution on edges will be *pointwise* ϵ-close to the uniform distribution, so that each edge will be returned with probability in $[\frac{1-\epsilon}{m}, \frac{1+\epsilon}{m}]$. Note that this is a stronger notion than the more standard notion of ϵ-close to uniform in *total variation distance* (TVD).[1] We consider this task in the

[1] See Section 1.1.0.2 for a detailed discussion comparing TVD-closeness to pointwise closeness.

sublinear setting, specifically, in the adjacency list query model, where the algorithm can perform uniform vertex queries, as well as degree and neighbor queries.

Three recent algorithms have been presented for this problem in the adjacency list model. The first, by Eden and Rosenbaum [13], is an $O^*(n/\sqrt{m})$ query complexity[2] algorithm that works in general graphs.[3] This was later refined by Eden, Ron, and Rosenbaum [7] to an $O^*(m\alpha/n)$ algorithm for graphs that have arboricity[4] at most α (where it is assumed that α is given as input to the algorithm). Finally, in [26], Tětek and Thorup combined techniques from the previous two works and presented the state of the art algorithm for sampling a single edge. This algorithm exponentially improves on the dependency in $1/\epsilon$ compared to the algorithm by [13]. All of these algorithms were also shown to be essentially optimal if one is interested in outputting a *single* edge sample. Naively, to sample q edges in general graphs, one can invoke the [26] algorithm q times, with expected complexity $O^*(q \cdot (n/\sqrt{m}))$. In this paper, we prove that this query complexity can be improved to $O^*(\sqrt{q} \cdot (n/\sqrt{m}))$. That is, we prove that there exists an algorithm with a better *amortized* query complexity.

1.1 Results

We present an algorithm that returns an edge from a distribution that is pointwise ϵ-close to uniform, and efficiently supports many edge sample invocations. Assuming one knows in advance the number of required edge samples q, the overall cost of q edge samples is $O^*(q \cdot (n/\sqrt{m}) + q) = O^*(q \cdot (n/\sqrt{m}))$, where the equality is since we can assume that $q = O(n^2/m)$.[5] Subsequent to a preliminary version of this work, Tětek and Thorup [26, Theorem 15] proved that the above result is essentially optimal.

Our algorithm is based on two procedures: a preprocessing procedure that is invoked once, and a sampling procedure which is invoked whenever an edge sample is requested. There is a trade-off between the preprocessing cost and per-sample cost of the sampling procedure. Namely, for a trade-off parameter $x \geq 1$, which can be given as input to the algorithm, the preprocessing query complexity is $O^*(n^2/(m \cdot x))$ and the per-sample cost of the sampling procedure is $O(x/\epsilon)$.

▶ **Theorem 1.1** (Informal.). *Let G be a graph over n vertices and m edges. Assume access to G is given via the adjacency list query model. There exists an algorithm that, given an approximation parameter ϵ and a trade-off parameter x, has two procedures: a preprocessing procedure, and a sampling procedure. The sampling procedure outputs an edge from a distribution that is pointwise ϵ-close to uniform. The preprocessing procedure has $O^*(n^2/(m \cdot x))$ expected query complexity, and the expected per-sample query complexity of the sampling procedure is $O(x/\epsilon)$.*

As mentioned previously, this result is essentially optimal, due to a lower bound by Tětek and Thorup [26].

---

[2] We note that in all the mentioned algorithms the running time is asymptotically equal to the query complexity, and therefore we limit the discussion to query complexity.

[3] Throughout the paper $O^*(\cdot)$ is used to suppresses $\text{poly}(\log n/\epsilon)$ dependencies.

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

[5] Observe that if the number of required samples q exceeds n^2/m, then one an simply perform $O(n^2 \log n/m)$ uniform pair queries and with high probability recover all edges in the graph. Hence, we can assume that $q \leq n^2/m$, and so the term q does not asymptotically affect the complexity.

▶ **Theorem 1.2** (Theorem 15 in [26], restated). *Let ϵ be some small constant $0 < \epsilon < 1$. Any algorithm that samples q edges from a distribution that is pointwise ϵ-close to uniform in the adjacency list query model must perform $\Omega(\sqrt{q} \cdot (n/\sqrt{m}))$ queries.*

To better understand how the complexity of our upper bound compares to what was previously known, we give some possible instantiations. First, setting $x = n/\sqrt{m}$ implies a preprocessing phase with $O^*(n/\sqrt{m})$ queries and a cost of $O(n/\sqrt{m})$ per sample, thus recovering the bounds of [13]. Second, setting $x = 1$ implies a preprocessing phase with $O(n^2/m)$ queries and a cost of $O(1/\epsilon)$ per sample. This can be compared to the naive approach of querying the degrees of all the vertices in the graph, and then sampling each vertex with probability proportional to its degree and returning an edge incident to the sampled vertex.[6] Hence, the naive approach yields an $O(n)$ preprocessing cost and $O(1)$ per-sample cost while our algorithm with $x = 1$ yields an $O^*(n^2/m) = O^*(n/d_{\mathsf{avg}})$ preprocessing and $O(1/\epsilon)$ per-sample cost, where d_{avg} denotes the average degree of the graph.

For a concrete example, consider the case where $m = \Theta(n)$ and $q = O(\sqrt{n})$ edge samples are required. Setting $x = n^{1/4}$ gives an overall cost of $n^{3/4}$ for sampling q edges, where previously this would have required $O(n)$ queries (by either the naive approach, or performing $O(\sqrt{n})$ invocations of the $O^*(n/\sqrt{m}) = O^*(\sqrt{n})$ algorithm of [26]). In general, if the number of queries q is known in advance, then setting $x = \frac{n/\sqrt{m}}{\sqrt{q}}$, yields that sampling q edges has an overall cost of $O^*(\sqrt{q} \cdot (n/\sqrt{m}))$, where previously this would have required $O^*(q \cdot (n/\sqrt{m}))$ queries resulting from q invocations of the algorithm by [26]. We discuss some more concrete applications in the following section.

1.1.0.1 From the augmented model to the general query model.

Recently, it has been suggested by Aliakbarpour et al. [3] to consider query models that also provide queries for uniform edge samples, and multiple algorithms have since been developed for this model, e.g., [4, 15, 5, 28].

Currently, for "transferring" results in models that allow uniform edge samples back to models that do not allow such queries in a black-box manner,[7] one must either (1) pay a multiplicative cost of $O^*(n/\sqrt{m})$ per query (replacing each edge sample query in an invocation of the [13] algorithm for sampling edges), (2) pay an additive cost of $O(n)$ (using the naive approach described above), or (3) pay an additive cost of $O^*(n^2/m)$ if pair queries[8] are allowed.[9]

For example, the works by Assadi, Kapralov and Khanna [4], Fichtenberger, Gao and Peng [15], and Biswas, Eden and Rubinfeld [5] give algorithms that rely on edge samples for the tasks of approximately counting and uniformly sampling arbitrary subgraphs in sublinear time. Specifically, these works assume the *augmented* query model which allows for vertex, degree, neighbor, pair as well as uniform edge samples queries. When only vertex, degree, neighbor and pair queries (without uniform edge samples) are provided, this is referred to as the *general* query model [21]. Currently, there are no dedicated algorithms for these tasks in the general model, that does not allow edge samples. For approximating the number of 4-cycles, denoted $\#C_4$, the algorithms of [4, 15] have query complexity of $O^*(m^2/\#C_4)$.

[6] Indeed, the naive approach returns an edge from a distribution that is *exactly* uniform.
[7] This is true for results for which pointwise-close to uniform edge samples are sufficient, as in the case in all the current sublinear results that rely on edge samples (that we know of).
[8] Pair queries return whether there is an edge between two vertices in the graph.
[9] As one can sample *all* edges in the graph with high probability using $O^*(n^2/m)$ uniform pair queries (by the coupon collector's argument), and then return from the set of sampled edges.

For a graph with $m = O(n)$ edges and $\#C_4 = \Theta(n^{3/2})$ 4-cycles, this results in an $O^*(\sqrt{n})$ query complexity in the augmented model. Using our algorithm, we can set $q = O(\sqrt{n})$, and approximately count the number of $\#C_4$'s in $O^*(n^{3/4})$ queries in the general query model, where previously to our results this would have cost $O(n)$ queries. We note that this "black-box" transformation from the augmented model to the general query model is not guaranteed to be optimal in terms of the resulting complexity in the general model. Indeed, dedicated algorithms for counting and sampling stars and cliques in the general model, prove that this is not the case [18, 9, 11, 10, 8, 28]. Nonetheless, to the best of our knowledge, no other results are currently known for subgraphs apart from stars or cliques, and so this approach provides the only known algorithms for arbitrary subgraph counting and sampling in the general model.

1.1.0.2 Pointwise vs. TVD.

A more standard measure of distance between two distributions P and Q is the total variation distance (TVD), $d_{TV}(P,Q) = \frac{1}{2}\sum_{x\in\Omega}|P(x) - Q(x)|$. Observe that this is a strictly weaker measure. That is, pointwise-closeness implies closeness in TVD. Thus our algorithm immediately produce a distribution that is TVD close to uniform. However, being close to a distribution in TVD, does not imply pointwise-closeness.[10] Furthermore, in various settings, this weaker definition is not sufficient, as is the case in some of the applications we mentioned previously. For instance, the uniform edge samples in the algorithms of [4, 15] cannot be replaced in a black-box manner by edge samples that are only guaranteed to be close to uniform in TVD. For a concrete example, consider the task of approximately counting the number of triangles. Let $G = A \cup B$ be a graph, where A is a bipartite subgraph over $(1-\epsilon)m$ edges, and B is a clique over ϵm edges. An algorithm that returns a uniformly distributed edge in A is close in TVD to uniform over the entire edge set of G. However, it does not allow one to correctly approximate the number of triangles in G, as the algorithm will never return an edge from the clique, which is where all the triangles reside.

1.2 Technical Overview

Sampling (almost) uniformly distributed edges is equivalent to sampling vertices with probability (almost) proportional to their degree $\frac{d(v)}{2m}$.[11] Hence, from now on we focus on the latter task.

Consider first the following naive procedure for sampling vertices with probability proportional to their degree. Assume that $d_{\max}$, the maximum degree in the graph is known. Query a vertex uniformly at random and return it with probability $\frac{d(v)}{d_{\max}}$; otherwise, return fail. Then each vertex is sampled with probability $\frac{d(v)}{n\cdot d_{\max}}$. Therefore, if we repeatedly invoke the above until a vertex is returned, then each vertex is returned with probability $\frac{d(v)}{2m}$, as desired. However, the expected number of attempts until a vertex is returned is $O(\frac{n\cdot d_{\max}}{m})$ (since the overall success probability of a single attempt is $\sum_{v\in V}\frac{d(v)}{n\cdot d_{\max}} = \frac{2m}{n\cdot d_{\max}}$), which could be as high as $O(\frac{n^2}{m})$ when $d_{\max} = \Theta(n)$.

[10] E.g., a distribution that ignores $\epsilon/2$-fraction of the edges and is uniform on the rest is close in TVD to uniform, but clearly it is not pointwise close.

[11] Since if every v is sampled with probability in $(1\pm\epsilon)\frac{d(v)}{2m}$, performing one more uniform neighbor query from v implies that each specific edge (v,w) in the graph is sampled with probability in $(1\pm\epsilon)\cdot\frac{1}{2m}$.

Our idea is to partition the graph vertices into *light* and *heavy*, according to some degree threshold τ, that will play a similar role to that of $d_{\max}$ in the naive procedure above. Our algorithm has two procedures, a preprocessing procedure and a sampling procedure. The preprocessing procedure is invoked once in the beginning of the algorithm, and the sampling procedure is invoked every time an edge sample is requested. In the preprocessing procedure we construct a data structure that will later be used to sample heavy vertices. In the sampling procedure, we repeatedly try to sample a vertex, each time either a light or a heavy with equal probability, until a vertex is returned. To sample light vertices, we invoke the above simple procedure with τ instead of $d_{\max}$. Namely, sample a uniform random vertex v, if $d(v) \leq \tau$, return it with probability $\frac{d(v)}{\tau}$. To sample heavy vertices, we use the data structure constructed by the preprocessing procedure as will be detailed shortly.

In the preprocessing procedure, we sample a set S of $O\left(\frac{n}{\tau} \cdot \frac{\log n}{\epsilon^2}\right)$ vertices uniformly at random. We then construct a data structure that allows to sample edges incident[12] to S uniformly at random. It holds that with high probability for every heavy vertex v, its number of neighbors in S, denoted $d_S(v)$, is close to its expected value, $d(v) \cdot \frac{|S|}{n}$. Also, it holds that with high probability the sum of degrees of the vertices in S, denoted $d(S)$, is close to its expected value, $2m \cdot \frac{|S|}{n}$. Hence, to sample heavy vertices, we first sample an edge (u, v) incident to S uniformly at random (without loss of generality $u \in S$) and then we check if the second endpoint v is heavy. If so, we return v, and otherwise we fail. By the previous discussion on the properties of S, it holds that every heavy vertex is sampled with probability approximately $\frac{d_S(v)}{d(S)} \approx \frac{d(v)}{2m}$.

1.3 Comparison to Previous Work

For the sake of this discussion assume that ϵ is some small constant. Most closely related to our work, is the algorithm of [13]. Their algorithm also works by partitioning the graph's vertices to *light* and *heavy* vertices according to their some degree threshold θ. Their method of sampling light edges is identical to ours: one simply samples a vertex uniformly at random, and keeps it with probability $d(v)/\theta$. In our algorithm, τ is the degree threshold for light and heavy vertices, so that τ and θ plays the same role. The difference between our works is in the sampling of heavy vertices. To sample heavy vertices, the algorithm of [13] tries to reach heavy vertices by sampling light vertices, and then querying one of their neighbors uniformly at random. For this approach to output heavy vertices with almost equal probability to light vertices, θ must be set to $\Omega(\sqrt{m})$. Our approach for sampling heavy vertices is different, and relies on the preprocessing phase, which later allows us to reach heavy vertices with $O(1)$ queries. This allows us, in a sense, to decouple the dependence of the threshold τ and the success probability of sampling light vertices. Hence, we can allow to set the degree threshold τ to smaller values, which results in a more efficient per-sample complexity (at a cost of a preprocessing step).

The algorithm of [7] also outputs a uniformly distributed single edge, however in graphs with bounded arboricity α. Here too the algorithm first defines light vertices, setting the threshold to $\Theta(\alpha)$. Sampling heavy edge is then performed by starting at light vertices as before, but taking longer random walks of length ℓ, for ℓ chosen uniformly in $[\log n]$. This method was later used by Tětek [26] to exponentially improve the dependence in ϵ of sampling a single edge in the general setting. It is an interesting open question whether

[12] We say that an edge (u, v) is incident to S if either u or v are in S.

there exists an algorithm for sampling multiple edges in bounded arboricity graphs which has better complexity than the algorithm of this work.

1.4 Further Related Work

We note that some of the related works were already mentioned, but we list them again for the sake of completeness.

1.4.0.1 Sampling edges in the adjacency list model.

As discussed previously, the most related work to ours is that of [13] for sampling a single edge from an almost uniform distribution in general graphs in $O^*(n/\sqrt{m})$ expected time. This was later refined by Eden, Rosenbaum and Ron [7] to an $O^*(n\alpha/m)$ expected time algorithm in bounded arboricity graphs, where a bound α on the arboricity of the graph at question is also given as input to the algorithm.[13] Recently, Tětek and Thorup [26] proved that the dependency in ϵ in the algorithm of [13] could be improved from $1/\sqrt{\epsilon}$ to $\log(1/\epsilon)$. They further proved (subsequent to our work) that given additional access to what they refer to as hash-based neighbor queries, there exists an algorithm for sampling multiple edges (with and without replacement) from the exactly uniform distribution in $O^*(\sqrt{q} \cdot (n/\sqrt{m}))$ time.

1.4.0.2 The augmented edge samples model.

In [3], Aliakbarpour et al. suggested a query model which allows access to uniform edge samples and degree queries. In this model they presented an algorithm for approximately counting the number of s-stars in expected time $O^*(m/\#H^{1/s})$, where $\#H$ denotes the number of s-stars in the graph. In [4], Assadi, Kaparalov and Khanna considered the combined power of neighbor, degree, pair and uniform vertex and edge samples. In this model, they presented an algorithm that approximates the number of occurrences of any arbitrary subgraph H in a graph G in expected time $O^*(m^{\rho(H)}/\#H)$, where $\rho(H)$ is the fractional edge cover[14] of H, and $\#H$ is the number of occurrences of H in G. In the same model, Fichtenberger, Gao, and Peng [15] simplified the above algorithm and proved the same complexity for the additional task of sampling a uniformly distributed copy of H. Recently, Biswas, Eden and Rubinfeld [5], paramerterized the complexity of counting and sampling arbitrary subgraph by what they refer to as the decomposition cost of H, improving the above results for a large family of subgraphs H. In [28], Tětek considers this model in the context of approximately counting triangles in the super-linear regime.

1.4.0.3 Sampling from networks.

Sampling from networks is a very basic primitive that is used in a host of works for studying networks' parameters (e.g., [20, 22, 31, 6, 27]). Most approaches for efficiently sampling edges from networks are random walk based approaches, whose complexity is proportional to the mixing time of the network, e.g., [22, 16, 25, 24]. We note that our approach cannot be directly compared with that of the random walk based ones, as the query models are different: The adjacency list query model assumes access to uniform vertex queries and one

[13] Note that since for all graphs $\alpha \leq \sqrt{m}$, this results is always at least as good as the previous one.
[14] The fractional edge cover of a graph is minimum weight assignment of weights to the graph's edges, so that the sum of weights over the edges incident to each vertex is at least 1.

can only query one neighbor at a time, while random walk based approaches usually only assume access to arbitrary seed vertices and querying a node reveals its set of neighbors. Furthermore, while in theory the mixing time of a graph can be of order $O(n)$, in practice, social networks tend to have smaller mixing times [24], making random walk based approaches very efficient. Still, denoting the mixing time of the network by t_{mix}, such approaches require one to perform $\Omega(t_{mix})$ queries in order to obtain *each* new sample, thus leaving the question of a more efficient amortized sampling procedure open.

2 Preliminaries

Let $G = (V, E)$ be an undirected simple graph over n vertices. We consider the adjacency list query model, which assumes the following set of queries:

- **Uniform vertex queries:** which return a uniformly distributed vertex in V.
- **Degree queries:** $deg(v)$, which return the degree of the queried vertex.
- **Neighbor queries** $nbr(v, i)$ which return the i^{th} neighbor of v, if one exists and $\perp$ otherwise.

We sometimes say that we perform a "uniform neighbor query" from some vertex v. This can be simply implemented by choosing an index $i \in [d(v)]$ uniformly at random, and querying $nbr(v, i)$.

Throughout the paper we consider each edge from both endpoints. That is, each edge $\{u, v\}$ is considered as two oriented edges (u, v) and (v, u). Abusing notation, let E denote the set of all oriented edges, so that $m = |E| = \sum_{v \in V} d(v)$ and $d_{\mathsf{avg}} = m/n$. Unless stated explicitly otherwise, when we say an "edge", we refer to oriented edges.

For a vertex $v \in V$ we denote by $\Gamma(v)$ the set of v's neighbors. For a set $S \subseteq V$ we denote by $E(S)$ the subset of edges (u, v) such that $u \in S$, and by $m(S)$ the sum of degrees of all vertices in S, i.e. $m(S) = |E(S)| = \sum_{v \in S} d(v)$. For every vertex $v \in V$ and set $S \subseteq V$, we denote by $d_S(v)$ the degree of v in S, $d_S(v) = |\Gamma(v) \cap S|$.

We consider the following definition of ϵ-pointwise close distributions:

▶ **Definition 1** (Definition 1.1 in [13]). *Let Q be a fixed probability distribution on a finite set Ω. We say that a probability distribution P is* pointwise ϵ-close to Q *if for all $x \in \Omega$,*

$$|P(x) - Q(x)| \leq \epsilon Q(x)\,, \quad \textit{or equivalently} \quad P(X) \in (1 \pm \epsilon)Q(X)\,.$$

If $Q = U$, the uniform distribution on Ω, then we say that P is pointwise ϵ-close to uniform.

3 Multiple Edge Sampling

As discussed in the introduction, our algorithm consists of a preprocessing procedure that creates a data structure that enables one to sample heavy vertices, and a sampling procedure that samples an almost uniformly distributed edge. Also recall that our procedures are parameterized by a value x which allows for a trade-off between the preprocessing complexity and the per-sample complexity. Namely, allowing per-sample complexity of $O(x/\epsilon)$, our preprocessing procedure will run in time $O^*(n/(d_{\mathsf{avg}} \cdot x))$. If one knows the number of queries, q, then setting $x = \frac{n/\sqrt{m}}{\sqrt{q}}$ yields the optimal trade-off between the preprocessing and the sampling.

3.1 Preprocessing

In this section we present our preprocessing procedure that will later allow us to sample heavy vertices. The procedure and its analysis are similar to the procedure Sample-degrees-typical of Eden, Ron, and Seshadhri [11].

The input parameters to the procedure are n, the number of vertices in the graph, x, the trade-off parameter, δ, a failure probability parameter, and ϵ, the approximation parameter. The output is a data structure that, with probability at least $1-\delta$, allows one to sample heavy vertices with probability (roughly) proportional to their degree.

We note that we set $\overline{x} = \min\{x, \sqrt{n/\overline{d}_{\mathsf{avg}}}\}$ since for values $x = \Omega(\sqrt{n/\overline{d}_{\mathsf{avg}}})$ it is better to simply use the $O^*(\sqrt{n/d_{\mathsf{avg}}})$ per-sample algorithm of [13]. We shall make use of the following theorems.

▶ **Theorem 3.1** (Theorem 1.1 of [17], restated.)**.** *There exists an algorithm that, given query access to a graph G over n vertices and m edges, an approximation parameter $\epsilon \in (0, \frac{1}{2})$, and a failure parameter $\delta \in (0,1)$, returns a value $\overline{m}$ such that with probability at least $1-\delta$, $\overline{m} \in [(1-\epsilon)m, m]$. The expected query complexity and running time of the algorithm are $O(\frac{n}{\sqrt{m}} \cdot \frac{\log^2 n}{\epsilon^{2.5}})$.*

▶ **Theorem 3.2** (Section 4.2 and Lemma 17 in [14], restated.)**.** *For a set S of size at least $\frac{n}{\sqrt{m}} \cdot \frac{34}{\epsilon}$, it holds that with probability at least $5/6$, $m(S)/s > \frac{1}{2} \cdot (1-\epsilon) \cdot d_{\mathsf{avg}}$.*

▶ **Theorem 3.3** (A data structure for a discrete distribution (e.g., [29, 30, 23]).)**.** *There exists an algorithm that receives as input a discrete probability distribution P over ℓ elements, and constructs a data structure that allows one to sample from P in linear time $O(\ell)$.*

Preprocessing (n, ϵ, δ, x)

1. Invoke the algorithm of [17][a] to get an estimate $\overline{d}_{\mathsf{avg}}$ of the average degree d_{avg}.
2. Let $\overline{x} = \min\left\{x, \sqrt{n/\overline{d}_{\mathsf{avg}}}\right\}$
3. Let $t = \lceil \log_3(\frac{3}{\delta}) \rceil$, and let $\tau = \frac{\overline{x} \cdot \overline{d}_{\mathsf{avg}}}{\epsilon}$.
4. For $i = 1$ to t do:
   a. Let S_i be a multiset of $s = \frac{n}{\tau} \cdot \frac{35 \log(6nt/\delta)}{\epsilon^2}$ vertices chosen uniformly at random.
   b. Query the degrees of all the vertices in S_i and compute $m(S_i) = \sum_{v \in S_i} d(v)$.
5. Let S be the first set S_i such that $\frac{m(S_i)}{s} \in \left[\frac{1}{4} \cdot \overline{d}_{\mathsf{avg}}, 12 \cdot \overline{d}_{\mathsf{avg}}\right]$.
   a. If no such set exists, then **return** *fail*.
   b. Else, set up a data structure[b] $D(S)$ that supports sampling each vertex $v \in S$ with probability $\frac{d(v)}{m(S)}$.
6. Let $\overline{\gamma} = \frac{m(S)}{\overline{d}_{\mathsf{avg}} \cdot |S|}$.
7. **Return** $(\overline{\gamma}, \tau, \overline{x}, D(S))$.

[a] See Theorem 3.1
[b] See Theorem 3.3

The following definitions will be useful in order to prove the lemma regarding the performance of the **Preprocessing** procedure.

▶ **Definition 2.** *We say that a sampled set $S \subseteq V$ is ϵ-good if the following two conditions hold:*

- *For every heavy vertex* $v \in V_{>\tau}$, $d_S(v) \in (1 \pm \epsilon)|S| \cdot \frac{d(v)}{n}$.
- $\frac{m(S)}{s} \in \left[\frac{1}{4} \cdot d_{\mathsf{avg}}, 12 \cdot d_{\mathsf{avg}}\right]$.

▶ **Definition 3.** *We say that* $\overline{d}_{\mathsf{avg}}$ *is an* ϵ*-good estimate of* d_{avg} *if* $\overline{d}_{\mathsf{avg}} \in [(1-\epsilon)d_{\mathsf{avg}}, d_{\mathsf{avg}}]$.

▶ **Lemma 4.** *Assume query access to a graph* G *over* n *vertices,* $\epsilon \in (0, \frac{1}{2})$, $\delta \in (0,1)$, *and* $x \geq 1$. *The procedure* **Preprocessing**(n, ϵ, δ, x), *with probability at least* $1-\delta$, *returns a tuple* $(\overline{\gamma}, \tau, \overline{x}, D(S))$ *such that the following holds.*

- $D(S)$ *is a data structure that supports sampling a uniform edge in* $E(S)$, *for an* ϵ*-good set* S, *as defined in Definition 2.*
- $\overline{x} \in [1, \sqrt{n/\overline{d}_{\mathsf{avg}}}]$, $\tau = \frac{\overline{x} \cdot \overline{d}_{\mathsf{avg}}}{\epsilon}$, *and* $\overline{\gamma} = \frac{m(S)}{\overline{d}_{\mathsf{avg}} \cdot |S|}$, *where* $\overline{d}_{\mathsf{avg}}$ *is an* ϵ*-good estimate of* d_{avg}, *as defined in Definition 3.*

The expected query complexity and running time of the procedure are $O\left(\max\left\{\frac{n}{d_{\mathsf{avg}} \cdot x}, \sqrt{\frac{n}{d_{\mathsf{avg}}}}\right\} \cdot \frac{\log^2(n \log(1/\delta)/\delta)}{\epsilon}\right)$.

Proof. We start with proving that with probability at least $1-\delta$ the set S chosen in Step 5 is a good set. Namely, that (1) $\frac{m(S)}{|S|} \in \left[\frac{1}{4} \cdot \overline{d}_{\mathsf{avg}}, 12 \cdot \overline{d}_{\mathsf{avg}}\right]$, and that (2) for all heavy vertices $v \in V_{>\tau}$, $d_S(v) \in (1 \pm \epsilon)s \cdot \frac{d(v)}{n}$.

By Theorem 1.1 of [17] (see Theorem 3.1), with probability at least $1 - \frac{\delta}{3}$, $\overline{d}_{\mathsf{avg}}$ is an ϵ-good estimate of d_{avg}, that is

$$(1-\epsilon)d_{\mathsf{avg}} \leq \overline{d}_{\mathsf{avg}} \leq d_{\mathsf{avg}}. \tag{1}$$

We henceforth condition on this event, and continue to prove the latter property. Fix an iteration $i \in [t]$. Observe that $\mathbb{E}\left[\frac{m(S_i)}{s}\right] = d_{\mathsf{avg}}$. By Markov's inequality,[15] equation (1), and the assumption that $\epsilon \in (0, \frac{1}{2})$,

$$\Pr\left[\frac{m(S_i)}{s} > 12 \cdot \overline{d}_{\mathsf{avg}}\right] \leq \frac{d_{\mathsf{avg}}}{12 \cdot \overline{d}_{\mathsf{avg}}} \leq \frac{1}{12(1-\epsilon)} \leq \frac{1}{6}.$$

Recall that $s = \frac{n}{\tau} \cdot \frac{35 \log(6nt/\delta)}{\epsilon^2}$, $\tau = \frac{\overline{x} \cdot \overline{d}_{\mathsf{avg}}}{\epsilon}$, and $\overline{x} \leq \sqrt{n/\overline{d}_{\mathsf{avg}}}$ and that we condition on $\overline{d}_{\mathsf{avg}} \geq (1-\epsilon)d_{\mathsf{avg}}$. Thus, $\tau \leq \frac{\sqrt{m}}{\epsilon}$, and $s \geq \frac{34}{\epsilon}\frac{n}{\sqrt{m}}$. Therefore, by Lemma 17 in [14] (see Theorem 3.2), for every i, it holds that

$$\Pr\left[\frac{m(S_i)}{s} \leq \frac{1}{2} \cdot (1-\epsilon)\, d_{\mathsf{avg}}\right] \leq \frac{1}{6}. \tag{2}$$

By equations (1), (2), and the assumption that $\epsilon \in (0, \frac{1}{2})$,

$$\Pr\left[\frac{m(S_i)}{s} < \frac{1}{4} \cdot \overline{d}_{\mathsf{avg}}\right] \leq \Pr\left[\frac{m(S_i)}{s} \leq \frac{1}{2} \cdot (1-\epsilon)\, d_{\mathsf{avg}}\right] \leq \frac{1}{6}$$

By the union bound, for every specific i,

$$\Pr\left[\frac{m(S_i)}{s} < \frac{1}{4} \cdot \overline{d}_{\mathsf{avg}} \quad \text{or} \quad \frac{m(S_i)}{s} > 12 \cdot \overline{d}_{\mathsf{avg}}\right] \leq \frac{1}{3}.$$

Hence, the probability that for all the selected multisets $\{S_i\}_{i \in [t]}$, either $\frac{m(S_i)}{s} < \frac{1}{4} \cdot \overline{d}_{\mathsf{avg}}$ or $\frac{m(S_i)}{s} > 12 \cdot \overline{d}_{\mathsf{avg}}$ is bounded by $\frac{1}{3^t} = \frac{\delta}{3}$ (recall $t = \lceil \log_3(\frac{3}{\delta}) \rceil$). Therefore, with probability at

[15] Markov's inequality: if X is a non-negative random variable and $a > 0$, $P(X \geq a) \leq \frac{E(X)}{a}$.

least $1-\frac{2\delta}{3}$, it holds that $\frac{m(S)}{s} \in \left[\frac{1}{4}\cdot\overline{d}_{\mathsf{avg}}, 12\cdot\overline{d}_{\mathsf{avg}}\right]$, and the procedure does not return *fail* in Step 5a.

Next, we prove that there exists a high-degree vertex $v \in V_{>\tau}$ such that $d_S(v) \notin (1\pm\epsilon)s\cdot\frac{d(v)}{n}$ with probability at most $\frac{\delta}{3}$. Fix an iteration $i\in[t]$, and let $S_i=\{u_1,\ldots,u_s\}$ be the sampled set. For any fixed high-degree vertex $v\in V_{>\tau}$ and for some vertex $u\in V$, let

$$\chi^v(u)=\begin{cases}1 & u \text{ is a neighbor of } v\\ 0 & \text{otherwise}\end{cases}.$$

Observe that $\mathbb{E}_{u\in V}\left[\chi^v(u)\right]=\frac{d(v)}{n}$, and that $d_{S_i}(v)=\sum_{j\in[s]}\chi^v(u_j)$. Thus, $\mathbb{E}\left[d_{S_i}(v)\right]=s\cdot\frac{d(v)}{n}$. Since the $\chi^v(u)$ variables are independent $\{0,1\}$ random variables, by the multiplicative Chernoff bound,[16]

$$\Pr\left[\left|d_{S_i}(v)-\frac{s\cdot d(v)}{n}\right|\geq\epsilon\cdot\frac{s\cdot d(v)}{n}\right]\leq 2\exp\left(-\frac{\epsilon^2\cdot s\cdot d(v)}{3n}\right)\leq\frac{\delta}{3nt}, \tag{3}$$

where the last inequality is by the assumption that $\epsilon\in(0,\frac{1}{2})$, the setting of $s=\frac{n}{\tau}\cdot\frac{35\log(6nt/\delta)}{\epsilon^2}$, and since we fixed a heavy vertex v so that $d(v)\geq\tau$. By taking a union bound over all high-degree vertices, it holds that there exists $v\in V_{>\tau}$ such that $d_{S_i}(v)\notin(1\pm\epsilon)\frac{s\cdot d(v)}{n}$ with probability at most $\frac{\delta}{3t}$.

Hence, with probability at least $1-\delta$, $D(S)$ is a data structure of a good set S. Moreover, by steps 2, 6, and 3 in the procedure **Preprocessing**(n,ϵ,δ,x) it holds that $\overline{x}\in\left[1,\sqrt{n/\overline{d}_{\mathsf{avg}}}\right]$, $\overline{\gamma}=\frac{m(S)}{\overline{d}_{\mathsf{avg}}\cdot|S|}$, and $\tau=\frac{\overline{x}\cdot\overline{d}_{\mathsf{avg}}}{\epsilon}$ respectively. By equation (1), $\overline{d}_{\mathsf{avg}}$ is an ϵ-good estimate for d_{avg}.

We now turn to analyze the complexity. By [17] (see Theorem 3.1), the query complexity and running time of step 1 is $O\left(\frac{n}{\sqrt{m}}\cdot\frac{\log^2(n)}{\epsilon^{2.5}}\right)$. The expected query complexity and running time of the for loop are $O(t\cdot s)=O(\frac{n}{\overline{d}_{\mathsf{avg}}\cdot\overline{x}}\cdot\frac{\log^2(n\log(1/\delta)/\delta)}{\epsilon})$, where the equality holds by the setting of s,t and since the expected value of $\overline{d}_{\mathsf{avg}}$ is d_{avg}. Step 5 takes $O(t)$ time. By [29, 30, 23] (see Theorem 3.3), the running time of step 5b is $O(s)$. All other steps takes $O(1)$ time. Hence, the expected query complexity and running time are dominated by the for loop. By the setting of $\overline{x}=\min\{x,\sqrt{n/\overline{d}_{\mathsf{avg}}}\}$ we have $O(s\cdot t)=O\left(\frac{n}{\overline{d}_{\mathsf{avg}}\cdot\overline{x}}\cdot\frac{\log^2(n\log(1/\delta)/\delta)}{\epsilon}\right)=O\left(\max\left\{\frac{n}{d_{\mathsf{avg}}\cdot x},\sqrt{\frac{n}{\overline{d}_{\mathsf{avg}}}}\right\}\cdot\frac{\log^2(n\log(1/\delta)/\delta)}{\epsilon}\right)$ which proves the claim. ◀

3.2 Sampling an edge

In this section we present our sampling procedures. The following definition and claim will be useful in our analysis.

▶ **Definition 5.** *Let τ be a degree threshold. Let $V_{\leq\tau}=\{v\in V\mid d(v)\leq\tau\}$, and let $V_{>\tau}=V\setminus V_{\leq\tau}$. We refer to $V_{\leq\tau}$ and $V_{>\tau}$ as the sets of* light vertices *and* heavy vertices*, respectively. Let $E_{\leq\tau}=\{(u,v)\mid u\in V_{\leq\tau}\}$ and $E_{>\tau}=\{(u,v)\mid u\in V_{>\tau}\}$.*

▶ **Definition 6.** *If the procedure* **Preprocessing**(n,ϵ,δ,x) *returns a tuple $(\overline{\gamma},\tau,\overline{x},D(S))$ such that the following items of Lemma 4 hold, then we say that this invocation is* successful*.*

[16] Multiplicative Chernoff bound: if $X_1,\ldots,X_n$ are independent random variables taking values in $\{0,1\}$, then for any $0\leq\delta\leq1$, $\Pr\left[\left|\sum_{i\in[n]}X_i-\mu\right|\geq\delta\mu\right]\leq 2e^{-\frac{\delta^2\mu}{3}}$ where $\mu=\mathbb{E}\left[\sum_{i\in[n]}X_i\right]$.

- $D(S)$ *is a data structure that supports sampling a uniform edge in* $E(S)$*, for an* ϵ*-good set* S*, as defined in Definition 2.*
- $\overline{x} \in [1, \sqrt{n/\overline{d}_{\mathsf{avg}}}]$*,* $\tau = \frac{\overline{x} \cdot \overline{d}_{\mathsf{avg}}}{\epsilon}$*, and* $\overline{\gamma} = \frac{m(S)}{\overline{d}_{\mathsf{avg}} \cdot |S|}$*, where* $\overline{d}_{\mathsf{avg}}$ *is an* ϵ*-good estimate of* d_{avg}*, as defined in Definition 3.*

▷ Claim 7. Let $\gamma = \frac{m(S)}{d_{\mathsf{avg}} \cdot |S|}$ and $\overline{\gamma} = \frac{m(S)}{\overline{d}_{\mathsf{avg}} \cdot |S|}$. If S is an ϵ-good set, as in Definition 2, and $\overline{d}_{\mathsf{avg}}$ is an ϵ-good estimate of d_{avg}, as in Definition 3, then it holds that $\overline{\gamma} \in [1/4, 12]$ and that $\gamma \in [(1-\epsilon)\overline{\gamma}, \overline{\gamma}]$.

Proof. By the assumption that S is an ϵ-good set, it holds that $\frac{m(S)}{|S|} \in [\frac{1}{4} \cdot \overline{d}_{\mathsf{avg}}, 12 \cdot \overline{d}_{\mathsf{avg}}]$. Therefore, $\overline{\gamma} \in [\frac{1}{4}, 12]$. By the assumption that $\overline{d}_{\mathsf{avg}}$ is an ϵ-good estimate of d_{avg}, namely $\overline{d}_{\mathsf{avg}} \in [(1-\epsilon)d_{\mathsf{avg}}, d_{\mathsf{avg}}]$, it holds that $\gamma \in [(1-\epsilon)\overline{\gamma}, \overline{\gamma}]$. ◀

3.2.1 The sampling procedures

We now present the two procedures for sampling light edges and heavy edges.

Sample-Uniform-Edge $(\overline{\gamma}, \tau, \overline{x}, D(S), \epsilon)$

1. While **True** do:
   a. Sample uniformly at random a bit $b \leftarrow \{0,1\}$.
   b. If $b = 0$ invoke **Sample-Light**$(\overline{\gamma}, \tau)$.
   c. Otherwise, invoke **Sample-Heavy**$(\tau, D(S), \overline{x}, \epsilon)$.
   d. If an edge (v,u) was returned, then **return** (v,u).

Sample-Light $(\overline{\gamma}, \tau)$

1. Sample a vertex $v \in V$ uniformly at random and query for its degree.
2. If $d(v) > \tau$ **return** *fail*.
3. Query a uniform neighbor of v. Let u be the returned vertex.
4. **Return** (v,u) with probability $\frac{d(v)}{\tau} \cdot \frac{1}{4\overline{\gamma}}$, otherwise **return** *fail*.

Sample-Heavy $(\tau, D(S), \overline{x}, \epsilon)$

1. Sample from the data structure $D(S)$ a vertex $v \in S$ with probability $\frac{d(v)}{m(S)}$.
2. Sample uniform neighbor of v. Let u be the returned vertex.
3. If $d(u) \leq \tau$ **return** *fail*.
4. Sample uniform neighbor of u. Let w be the returned vertex.
5. **Return** (u,w) with probability $\epsilon/4\overline{x}$, otherwise **return** *fail*.

Our procedure for sampling an edge **Sample-Uniform-Edge** gets as input a tuple $(\overline{\gamma}, \tau, \overline{x}, D(S))$ which is the output of the procedure **Preprocessing**. Our guarantees on the resulting distribution of edge samples rely on the preprocessing being successful (see Definition 6), which happens with probability at least $1 - \delta$.

▶ **Lemma 8.** *Assume that* **Preprocessing** *has been invoked successfully, as defined in Definition 6. The procedure* **Sample-Light**$(\overline{\gamma}, \tau)$ *returns an edge in* $E_{\leq \tau}$ *such that each edge is returned with probability* $\frac{\epsilon |S|}{4n \cdot \overline{x} \cdot m(S)}$*. The query complexity and running time of the procedure are* $O(1)$*.*

Proof. Let (v,u) be a fixed edge in $E_{\leq\tau}$.

$$\begin{aligned}\Pr[(v,u) \text{ returned}] = \Pr[\ &(v \text{ is sampled in Step 1}) \text{ and } (u \text{ sampled in Step 3}) \\ &\text{and } ((v,u) \text{ returned in Step 4})] \\ &= \frac{1}{n} \cdot \frac{1}{d(v)} \cdot \frac{d(v)}{\tau \cdot 4\overline{\gamma}} \ .\end{aligned}$$

Note that by Claim 7, $1/4\overline{\gamma} \leq 1$ and therefore, Step 4 is valid and the above holds. Hence, by the setting of $\tau = \frac{\overline{x}\cdot\overline{d}_{\mathsf{avg}}}{\epsilon}$ and $\overline{\gamma} = \frac{m(S)}{\overline{d}_{\mathsf{avg}}\cdot|S|}$,

$$\Pr[(v,u) \text{ is returned}] = \frac{1}{n \cdot \tau \cdot 4\overline{\gamma}} = \frac{\epsilon \cdot |S|}{4n \cdot \overline{x} \cdot m(S)}.$$

The procedure performs at most one degree query and one uniform neighbor query. All other operations take constant time. Therefore, the query complexity and running time of the procedure are constant. ◀

▶ **Lemma 9.** *Assume that* **Preprocessing** *has been invoked successfully, as defined in Definition 6. The procedure* **Sample-Heavy**$(\tau, D(S), \overline{x}, \epsilon)$ *returns an edge in $E_{>\tau}$ such that each edge is returned with probability $\frac{(1\pm\epsilon)\epsilon|S|}{4n\cdot\overline{x}\cdot m(S)}$. The query complexity and running time of the procedure are $O(1)$.*

Proof. Let (u,w) be an edge in $E_{>\tau}$. We first compute the probability that u is sampled in Step 2. Recall, the data structure $D(S)$ supports sampling a vertex v in S with probability $\frac{d(v)}{m(S)}$. The probability that u is sampled in Step 2 is equal to the probability that a vertex $v \in S$ which is a neighbor of u is sampled in step 1, and u is the selected neighbor of v in Step 2. Namely,

$$\Pr[u \text{ is sampled in Step 2}] = \sum_{v \in S \cap \Gamma(u)} \frac{d(v)}{m(S)} \cdot \frac{1}{d(v)} = \sum_{v \in S \cap \Gamma(u)} \frac{1}{m(S)} = \frac{d_S(u)}{m(S)} \ .$$

By the assumption that **Preprocessing** has been invoked successfully, so that S is ϵ-good, and because $u \in V_{>\tau}$,

$$d_S(u) \in (1 \pm \epsilon) \cdot |S| \cdot \frac{d(u)}{n}.$$

Hence, the probability that (u,w) is returned by the procedure is

$$\begin{aligned}\Pr[(u,w) \text{ is returned}] = \Pr[\ &(u \text{ sampled in Step 2}) \text{ and } (w \text{ sampled in Step 5}) \\ &\text{and } ((u,w) \text{ returned in Step 5})] \\ &= \frac{d_S(u)}{m(S)} \cdot \frac{1}{d(u)} \cdot \frac{\epsilon}{4\overline{x}} \in \frac{(1\pm\epsilon)|S| \cdot \frac{d(u)}{n} \cdot \epsilon}{m(S) \cdot d(u) \cdot 4\overline{x}} = \frac{(1\pm\epsilon)\epsilon|S|}{4n \cdot \overline{x} \cdot m(S)} \ .\end{aligned}$$

The procedure performs one degree query and two neighbor queries, and the rest of the operations take constant time. Hence the query complexity and running time are constant. ◀

We are now ready to prove the formal version of Theorem 1.1.

▶ **Theorem 3.4.** *There exists an algorithm that gets as input query access to a graph G, n, the number of vertices in the graph, $\epsilon \in (0, \frac{1}{2})$, an approximation parameter, $\delta \in (0,1)$, a failure parameter, and $x > 1$, a trade-off parameter. The algorithm has a preprocessing procedure and a sampling procedure.*

The preprocessing procedure has expected query complexity $O\left(\max\left\{\frac{n}{d_{\text{avg}}\cdot x}, \sqrt{\frac{n}{d_{\text{avg}}}}\right\}\cdot\frac{\log^2(n\log(1/\delta)/\delta)}{\epsilon}\right)$*, and it succeeds with probability at least* $1-\delta$*. If the preprocessing procedure succeeds, then each time the sampling procedure is invoked it returns an edge such that the distribution on returned edges is* 2ϵ*-point-wise close to uniform, as defined in Definition 1. Each invocation of the sampling procedure has expected* $O(\overline{x}/\epsilon)$ *query and time complexity.*

Proof. By 9, the procedure **Preprocessing** procedure succeeds with probability at least $1-\delta$. Furthermore, it has expected running time and query complexity as stated.

Condition on the event that the invocation of **Preprocessing** was successful. Let P denote the distribution over the returned edges by the procedure **Sample-Uniform-Edge**. By Lemma 2.3 in [13], in order to prove that P is pointwise 2ϵ-close to uniform, it suffices to prove that for every two edges e, e' in the graph, $\frac{P(e)}{P(e')} \in (1\pm 2\epsilon)$. By Lemma 8, every light edge e is returned with probability $\frac{\epsilon\cdot|S|}{4n\cdot\overline{x}\cdot m(S)}$. By Lemma 9, every heavy edge e' is returned with probability $\frac{(1\pm\epsilon)\epsilon|S|}{4n\cdot\overline{x}\cdot m(S)}$. Therefore, for every two edges e, e' in the graph, $\frac{P(e)}{P(e')} \in (1\pm 2\epsilon)$.

Next, we prove a lower bound on the success probability of a single invocation of the while loop in Step 1 in **Sample-Uniform-Edge**.

$$\begin{aligned}\Pr[\text{an edge is returned}] &= \frac{1}{2}\Pr[\textbf{Sample-Light} \text{ returns an edge}] \\ &\quad + \frac{1}{2}\Pr[\textbf{Sample-Heavy} \text{ returns an edge}] \\ &\geq \frac{1}{2}|E_{\leq\tau}|\cdot\frac{\epsilon\cdot|S|}{4n\cdot\overline{x}\cdot m(S)} + \frac{1}{2}\cdot|E_{>\tau}|\cdot\frac{(1-\epsilon)\epsilon\cdot|S|}{4n\cdot\overline{x}\cdot m(S)} \\ &\geq \frac{1}{2}\cdot\frac{(1-\epsilon)\cdot\epsilon|S|\cdot m}{4n\cdot\overline{x}\cdot m(S)} = \frac{(1-\epsilon)\epsilon}{8\gamma\overline{x}} \geq \frac{\epsilon}{192x}\,,\end{aligned}$$

where the second inequality is due to Claim 7, i.e. $\gamma \leq 12$. Hence, the expected number of invocations until an edge is returned is $O(\overline{x}/\epsilon)$.

◀

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.
2. Nesreen K Ahmed, Jennifer Neville, and Ramana Kompella. Network sampling: From static to streaming graphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(2):1–56, 2013.
3. 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.
4. Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A simple sublinear-time algorithm for counting arbitrary subgraphs via edge sampling. In *Innovations in Theoretical Computer Science Conference ITCS*, volume 124 of *LIPIcs*, pages 6:1–6:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.
5. Amartya Shankha Biswas, Talya Eden, and Ronitt Rubinfeld. Towards a decomposition-optimal algorithm for counting and sampling arbitrary motifs in sublinear time. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2021, to appear*, 2021.
6. Colin Cooper, Tomasz Radzik, and Yiannis Siantos. Estimating network parameters using random walks. *Social Network Analysis and Mining*, 4(1):168, 2014.

455 **7** Talya Eden, Dana Ron, and Will Rosenbaum. The arboricity captures the complexity of
456 sampling edges. In *46th International Colloquium on Automata, Languages, and Programming,*
457 *ICALP 2019, July 9-12, 2019, Patras, Greece.*, pages 52:1–52:14, 2019. doi:10.4230/LIPIcs.
458 ICALP.2019.52.
459 **8** Talya Eden, Dana Ron, and Will Rosenbaum. Almost optimal bounds for sublinear-time
460 sampling of k-cliques: Sampling cliques is harder than counting, 2020. arXiv:2012.04090.
461 **9** Talya Eden, Dana Ron, and C Seshadhri. Sublinear time estimation of degree distribution
462 moments: The arboricity connection. *SIAM Journal on Discrete Mathematics*, 33(4):2267–2285,
463 2019.
464 **10** Talya Eden, Dana Ron, and C Seshadhri. Faster sublinear approximation of the number
465 of k-cliques in low-arboricity graphs. In *Proceedings of the Fourteenth Annual ACM-SIAM*
466 *Symposium on Discrete Algorithms*, pages 1467–1478. SIAM, 2020.
467 **11** Talya Eden, Dana Ron, and C Seshadhri. On approximating the number of k-cliques in
468 sublinear time. *SIAM Journal on Computing*, 49(4):747–771, 2020.
469 **12** Talya Eden and Will Rosenbaum. Lower bounds for approximating graph parameters via
470 communication complexity. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer,
471 editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and*
472 *Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, volume
473 116 of *LIPIcs*, pages 11:1–11:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
474 doi:10.4230/LIPIcs.APPROX-RANDOM.2018.11.
475 **13** Talya Eden and Will Rosenbaum. On sampling edges almost uniformly. In Raimund Seidel,
476 editor, *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018, New*
477 *Orleans, LA, USA*, volume 61 of *OASICS*, pages 7:1–7:9. Schloss Dagstuhl - Leibniz-Zentrum
478 für Informatik, 2018. doi:10.4230/OASIcs.SOSA.2018.7.
479 **14** Uriel Feige. On sums of independent random variables with unbounded variance and estimating
480 the average degree in a graph. *SIAM Journal on Computing*, 35(4):964–984, 2006.
481 **15** Hendrik Fichtenberger, Mingze Gao, and Pan Peng. Sampling arbitrary subgraphs exactly
482 uniformly in sublinear time. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors,
483 *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020,*
484 *July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages
485 45:1–45:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.
486 ICALP.2020.45.
487 **16** Minas Gjoka, Maciej Kurant, Carter T. Butts, and Athina Markopoulou. Walking in facebook:
488 A case study of unbiased sampling of osns. In *INFOCOM 2010. 29th IEEE International*
489 *Conference on Computer Communications, Joint Conference of the IEEE Computer and*
490 *Communications Societies, 15-19 March 2010, San Diego, CA, USA*, pages 2498–2506. IEEE,
491 2010. doi:10.1109/INFCOM.2010.5462078.
492 **17** Oded Goldreich and Dana Ron. Approximating average parameters of graphs. *Random*
493 *Structures & Algorithms*, 32(4):473–493, 2008. doi:10.1002/rsa.20203.
494 **18** Mira Gonen, Dana Ron, and Yuval Shavitt. Counting stars and other small subgraphs in
495 sublinear-time. *SIAM Journal on Discrete Mathematics*, 25(3):1365–1411, 2011.
496 **19** Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for lp samplers, finding
497 duplicates in streams, and related problems. In *Proceedings of the thirtieth ACM SIGMOD-*
498 *SIGACT-SIGART symposium on Principles of database systems*, pages 49–58, 2011.
499 **20** Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. Efficient sampling algorithm for
500 estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–
501 1758, 2004.
502 **21** Tali Kaufman, Michael Krivelevich, and Dana Ron. Tight bounds for testing bipartiteness
503 in general graphs. *SIAM Journal on Computing*, 33(6):1441–1483, 2004. doi:10.1137/
504 S0097539703436424.
505 **22** Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th*
506 *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD

'06, pages 631–636, New York, NY, USA, 2006. ACM. URL: http://doi.acm.org/10.1145/1150402.1150479, doi:10.1145/1150402.1150479.

23 George Marsaglia, Wai Wan Tsang, Jingbo Wang, et al. Fast generation of discrete random variables. *Journal of Statistical Software*, 11(3):1–11, 2004.

24 Abedelaziz Mohaisen, Aaram Yun, and Yongdae Kim. Measuring the mixing time of social graphs. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 383–389, 2010.

25 Bruno Ribeiro and Don Towsley. Estimating and sampling graphs with multidimensional random walks. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 390–403, 2010.

26 Jakub Tětek and Mikkel Thorup. Sampling and counting edges via vertex accesses. *arXiv preprint arXiv:2107.03821*, 2021.

27 Duru Türkoglu and Ata Turk. Edge-based wedge sampling to estimate triangle counts in very large graphs. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 455–464. IEEE, 2017.

28 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.

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

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

31 Tianyi Wang, Yang Chen, Zengbin Zhang, Tianyin Xu, Long Jin, Pan Hui, Beixing Deng, and Xing Li. Understanding graph sampling algorithms for social network analysis. In *2011 31st international conference on distributed computing systems workshops*, pages 123–128. IEEE, 2011.