Non-Adaptive Edge Counting and Sampling via Bipartite Independent Set Queries

Raghavendra Addanki
Adobe Research, San Jose, CA, USA

Andrew McGregor
Manning College of Information and Computer Sciences,
University of Massachusetts Amherst, MA, USA

Cameron Musco
Manning College of Information and Computer Sciences,
University of Massachusetts Amherst, MA, USA

Abstract

We study the problem of estimating the number of edges in an n-vertex graph, accessed via the *Bipartite Independent Set* query model introduced by Beame et al. (TALG '20). In this model, each query returns a Boolean, indicating the existence of at least one edge between two specified sets of nodes. We present a *non-adaptive* algorithm that returns a $(1 \pm \epsilon)$ relative error approximation to the number of edges, with query complexity $\widetilde{O}(\epsilon^{-5} \log^5 n)$, where $\widetilde{O}(\cdot)$ hides $\mathrm{poly}(\log \log n)$ dependencies. This is the first non-adaptive algorithm in this setting achieving $\mathrm{poly}(1/\epsilon, \log n)$ query complexity. Prior work requires $\Omega(\log^2 n)$ rounds of adaptivity. We avoid this by taking a fundamentally different approach, inspired by work on single-pass streaming algorithms. Moreover, for constant ϵ, our query complexity significantly improves on the best known adaptive algorithm due to Bhattacharya et al. (STACS '22), which requires $O(\epsilon^{-2} \log^{11} n)$ queries. Building on our edge estimation result, we give the first non-adaptive algorithm for outputting a nearly uniformly sampled edge with query complexity $\widetilde{O}(\epsilon^{-6} \log^6 n)$, improving on the works of Dell et al. (SODA '20) and Bhattacharya et al. (STACS '22), which require $\Omega(\log^3 n)$ rounds of adaptivity. Finally, as a consequence of our edge sampling algorithm, we obtain a $\widetilde{O}(n \log^8 n)$ query algorithm for connectivity, using two rounds of adaptivity. This improves on a three-round algorithm of Assadi et al. (ESA '21) and is tight; there is no non-adaptive algorithm for connectivity making $o(n^2)$ queries.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases sublinear graph algorithms, bipartite independent set queries, edge sampling and counting, graph connectivity, query adaptivity

Digital Object Identifier 10.4230/LIPIcs.ESA.2022.2

Related Version *Full Version*: http://arxiv.org/abs/2207.02817

Funding This work was supported by a Dissertation Writing Fellowship awarded by the Manning College of Information and Computer Sciences, UMass Amherst to R. Addanki. In addition, this work was supported by NSF grants CCF-1934846, CCF-1908849, and CCF-1637536, awarded to A. McGregor; and NSF grants CCF-2046235, IIS-1763618, as well as Adobe and Google Research Grants, awarded to C. Musco.

Acknowledgements Most of this work was done while R. Addanki was a student at UMass Amherst. Part of this work was done while R. Addanki was a visiting student at the Simons Institute for the Theory of Computing. We thank the anonymous reviewers for their helpful suggestions.

1 Introduction

In this work, we study sublinear query algorithms for estimating the number of edges in a simple, unweighted graph $G = (V, E)$, and for sampling uniformly random edges. Access to G is via a *Bipartite Independent Set (BIS)* oracle [10]. A query to this oracle takes as input two disjoint subsets $L, R \subseteq V$ and returns

$$\mathcal{BIS}(L,R) = \begin{cases} \text{`1'} & \text{if there is no edge between } L \text{ and } R \\ \text{`0'} & \text{otherwise.} \end{cases}$$

Local Query Models. Prior work on sublinear query graph algorithms has largely focused on *local* queries, in particular, (i) vertex degree queries (ii) neighbor queries (output the i^{th} neighbor of a vertex) and (iii) edge existence queries [31, 33, 48]. In the literature, the first two types of queries form the *adjacency list* query model, while all three types of queries form the *adjacency matrix* query model. Under these models, a variety of graph estimation problems have been well studied, including edge counting and sampling [30, 33, 48, 51], subgraph counting [5, 17, 29], vertex cover [11, 43], and beyond [45].

For a graph with n nodes and m edges, given access only to degree queries, Feige [31] presented an algorithm for estimating m up to $(2 \pm \epsilon)$ relative error with query complexity $O(\sqrt{n} \cdot \text{poly}(1/\epsilon, \log n))$. This work also showed that any $(2 - o(1))$-approximation algorithm requires $\Omega(n)$ queries. In the adjacency list query model, Goldreich and Ron [33] gave a $(1 \pm \epsilon)$-approximation algorithm, with query complexity $O(n/\sqrt{m} \cdot \text{poly}(1/\epsilon, \log n))$. Recently, Eden and Rosenbaum [30] gave algorithms for near-uniform edge sampling with the same query complexity, and showed that this complexity is nearly tight.

Global Query Models. Motivated by the desire to obtain more query efficient algorithms, Beame et al. [10] studied edge estimation using *global* queries that can make use of information across the graph, including the BIS queries that we will focus on, and the related Independent Set (IS) queries, which were initially introduced in the literature on query efficient graph recovery [1, 6]. An IS query answers whether or not there exist any edges in the induced subgraph on a subset of nodes $S \subseteq V$. We refer the reader to the exposition in [10], which discusses applications of these global query models in group testing [22, 27], computational geometry [7, 18, 32], fine-grained complexity [25, 26], and decision versus counting complexity [26, 46, 49, 50].

In the IS query model, [10, 23] give a $O(\min\{\sqrt{m}, n/\sqrt{m}\} \cdot \text{poly}(\log n, 1/\epsilon))$ query algorithm for $(1 \pm \epsilon)$ approximate edge counting. In the BIS model, numerous authors [10, 26, 14] achieve $(1 \pm \epsilon)$-approximation for edge counting and near-uniform edge sampling using just $\text{poly}(1/\epsilon, \log n)$ queries. This can be exponentially smaller than the query complexities in the IS and local queries models.

Extending the BIS query model to hypergraphs, Dell et al. [26] introduce the *coloured independence oracle* which detects the presence of a size k hyperedge. They give algorithms for hyperedge estimation and sampling using this generalized oracle. Many other variants of global queries have been studied including Linear, OR and Cut queries [8, 20, 47]. These queries have been applied to solving maximum matching [38, 42], minimum cut [47], triangle estimation [12, 13, 26], connectivity [8], hitting sets [15], weighted edge estimation [16], problems related to linear algebra [44], quantum algorithms [40], and full graph recovery [1, 6].

The Role of Adaptivity. Notably, for both local and global queries, most sublinear time graph algorithms are *adaptive*, i.e., a query may depend on the answers to previous queries. In many cases, it is desirable for queries to be *non-adaptive*. This allows them to be completed

independently, and might allow for the resulting algorithm to be easily implemented in massively parallel computation frameworks [37]. Non-adaptive algorithms also lead naturally to single-pass, rather than multi-pass, streaming algorithms. In fact, the BIS query model can be seen as a very restricted subset of the more general LINEAR query model, in which each query outputs the inner product of the edge indicator vector with a query vector. This model has long been studied in the graph-streaming literature [4, 39], in part due to its usefulness in giving single-pass algorithms. However, it has remained open whether non-adaptive algorithms can be given in more restricted global query models.

For these reasons, Assadi et al. [8] and Chakrabarti and Stoeckl [20] have recently sought to reduce query adaptivity under a variety of global query models, including LINEAR, OR, CUT and BIS queries. These works study the *single element recovery* problem, which is a weaker variant of uniform edge sampling, requiring that the algorithm return a single edge in G. Assadi et al. also study the problem of checking connectivity, presenting a BIS query algorithm making $\widetilde{O}(n)$ queries and using three rounds of adaptivity. They give a two-round algorithm in the stronger OR query model, and show that even in this model, there is no non-adaptive algorithm for connectivity making $o(n^2)$ queries.

We note that reducing query adaptivity is also a well-studied direction in the closely related literature on group testing [28, 34]. IS and BIS oracles can be thought of as tests if there is a single element in a group of edges, where that group is required to be all edges incident on one node set (IS) or between two disjoint sets (BIS). Attempts to minimize query adaptivity have also been made for sparse recovery [35, 36, 41], submodular function maximization [9, 21], property testing [19] and multi-armed bandit learning [3].

1.1 Our Contributions

Our main result is the first *non-adaptive* algorithm for edge estimation up to $(1 \pm \epsilon)$ relative error, using $\text{poly}(1/\epsilon, \log n)$ BIS queries. Formally, we show:

▶ **Theorem 1** (Theorem 6 restated). *Given a graph G with n nodes and m edges, there is an algorithm that makes $O(\epsilon^{-5} \log^5 n \log^5(\log n) \log(\epsilon^{-1} \log n))$ non-adaptive BIS queries to G and returns an estimate $\widehat{m}$ satisfying: $m(1-\epsilon) \leq \widehat{m} \leq m(1+\epsilon)$,* with probability at least $3/5$.[1]

Prior methods for $(1 \pm \epsilon)$ error edge estimation using BIS queries are based on a binary search style approach [10, 26, 14], which is inherently adaptive, and leads to algorithms requiring $\Omega(\log^2 n)$ rounds of adaptivity. Beame et al. [10] present a non-adaptive algorithm giving a $O(\log^2 n)$ approximation factor for bipartite graphs, using $O(\log^3 n)$ queries. This algorithm can be extended to general graphs, via a simple reduction, described in Section 3.3. However, no non-adaptive results for general graphs achieving $1 \pm \epsilon$ relative error for arbitrary $\epsilon > 0$ were previously known. Even with adaptivity, the best known algorithm due to [14] has a query complexity of $O(\epsilon^{-2} \log^{11} n)$. The non-adaptive result of Theorem 1 improves upon this prior work significantly, whenever ϵ is constant with respect to n. Our second result builds on our edge estimation approach, giving the first non-adaptive BIS query algorithm that returns a near-uniformly sampled edge. Formally:

▶ **Theorem 2** (Theorem 7 restated). *Given a graph G with n nodes, m edges, and edge set E, there is an algorithm that makes $O(\epsilon^{-4} \log^6 n \log(\epsilon^{-1} \log n) + \epsilon^{-6} \log^5 n \log^6(\log n) \log(\epsilon^{-1} \log n))$ non-adaptive BIS queries which, with probability at least $1-\epsilon$, outputs an edge from a probability distribution P satisfying $(1-\epsilon)/m \leq P(e) \leq (1+\epsilon)/m$ for every $e \in E$.*

[1] Note that the success probability can be boosted in the standard way, by running multiple independent instantiations of the algorithm and taking their median estimate.

Prior results for near-uniform edge sampling required $\Omega(\log^3 n)$ rounds of adaptivity [14, 26]. Additionally, even ignoring adaptivity, our result improves on the best known query complexity of $O(\epsilon^{-2}\log^{14} n)$, due to [14], for large enough ϵ, including when ϵ is a constant.

By combining Theorem 2 with prior work on sublinear query graph connectivity, via edge sampling, we obtain a connectivity algorithm using two rounds for adaptivity:

▶ **Theorem 3** (Theorem 8 restated)**.** *Given a graph G with n nodes, there is a 2-round adaptive algorithm that determines if G is connected with probability at least $1 - 1/n$ using $O(n\log^8 n \log(\log n))$ BIS queries.*

Theorem 3 improves on a three-round algorithm of Assadi et al. [8] and is tight in terms of adaptivity: even in the stronger OR query model (which allows checking the presence of an edge within an arbitrary subset of node pairs) no non-adaptive algorithm can make $o(n^2)$ queries. Assadi et al. gave a two-round algorithm in this stronger OR query model. Thus, Theorem 3 closes the gap between BIS queries and OR queries for the connectivity problem. We note that there is a separation from the even stronger LINEAR query model, where non-adaptive algorithms for connectivity and cut approximation are well-known [4]. Understanding if there remain natural separations between the BIS and OR query models in terms of adaptivity would be very interesting.

2 Technical Overview

In this section, we present an overview of three main results: our non-adaptive BIS query algorithm for edge estimation (Theorem 1) and near-uniform edge sampling (Theorem 2), along with our 2-round algorithm for connectivity (Theorem 3).

2.1 Edge Estimation

A simple idea to estimate the number of edges in a graph via BIS queries is to sample small random subsets of nodes and run BIS queries to check the presence of edges between them. The fraction of these queries that return '1' (i.e., indicating the presence of no edge) can then be used to estimate the total number of edges in the graph. In particular, for a graph containing m edges, if the random subsets of nodes have $O(n/\sqrt{m})$ nodes in them, then we expect a '1' answer with constant probability. Beame et al. [10] describe a non-adaptive algorithm along these lines, which gives a $O(\log^2 n)$ approximation using $O(\log^3 n)$ queries. Unfortunately, going beyond this coarse approximation is difficult: many dependencies due to common neighbors arise and this increases the variance of the estimators. Beame et al. handle the issue by using the coarse estimates to subdivide the graph into smaller sub-graphs, until these divided graphs only contain $\operatorname{poly}\log n$ edges, at which point all their edges can be discovered with few queries. This strategy yields a $(1 \pm \epsilon)$ approximation, however, it is inherently adaptive.

Our non-adaptive algorithm takes a different approach. Let $d(v)$ denote the degree of a node v. Suppose we could sample each node with probability $p_v \approx d(v) \cdot \epsilon^{-2}/m$ and compute the degree of the sampled nodes. Letting $\widehat{m} = \sum_v \mathbb{I}[v \text{ sampled}] \cdot d(v)/p_v$, be the appropriately weighted average of the sampled degrees, it is straightforward to show that $\mathbf{E}[\widehat{m}] = \sum_v d(v) = 2m$. Further, via a standard Bernstein bound, $\widehat{m}/2$ gives a $(1 \pm \epsilon)$ relative error approximation to m with high probability. The challenge is showing that this type of approach can be approximated in the BIS query model.

Subsampling Nodes. The first idea, drawn from work on streaming algorithms, is to subsample the nodes of G at different rates of the form $1/\gamma^j$ where $\gamma > 1$ is constant and $j \in \{0, 1, \cdots, O(\log n)\}$. At each rate, we "recover" any sampled nodes (along with a corresponding degree estimate) whose degree is roughly $d(v) \approx \epsilon^2 m/\gamma^j$. In this way, each node will be recovered with probability roughly $1/\gamma^j \approx d(v) \cdot \epsilon^{-2}/m$, as desired. We describe this subsampling procedure in Section 3.3, as part of our main algorithm Edge-Estimator (Algorithm 3).

Recovering Heavy Nodes. The next challenge is to show that we can actually recover the appropriate nodes and degree estimates at each sampling rate. If we can approximate the degree of all nodes sampled at rate $1/\gamma^j$ up to additive error $O\left(\epsilon^3 \cdot m/\gamma^j\right)$, we will obtain a $(1 \pm \epsilon)$ relative error approximation to the degree of any node we hope to recover at that sampling rate, i.e., any node with degree roughly $\epsilon^2 m/\gamma^j$. Using these approximations, we can determine which nodes should be recovered at that rate, and form our edge estimate.

Degree Estimation via Neighborhood Size Estimation. To achieve such an additive error approximation, we use ideas from the sparse recovery and streaming literature. In particular, we implement an approach reminiscent of the Count-Min sketch algorithm [24]. The approach is described in detail in Section 3.2, where we present Algorithm Estimate-Degree (Algorithm 2). First observe that when sampling at rate $1/\gamma^j$, conditioned on any node v being included in the sample, the expected total degree of the sampled nodes other than v is $O(m/\gamma^j)$. If we further subdivide these nodes into $\widetilde{O}(1/\epsilon^3)$ random groups, the expected total degree of all nodes other than v in any group is $\widetilde{O}\left(\epsilon^3 \cdot m/\gamma^j\right)$.

Now, if v is placed in group S, we can approximately upper bound its degree by the total *neighborhood size of* S. This upper bound holds approximately as long as v does not have too many neighbors in S, which it won't with good probability. The neighborhood size of S is in turn upper bounded by the degree of v plus the total degree of other nodes in S, and thus by $d(v) + \widetilde{O}\left(\epsilon^3 \cdot m/\gamma^j\right)$ in expectation. So, in expectation, this approach gives an additive $\widetilde{O}\left(\epsilon^3 \cdot m/\gamma^j\right)$ error approximation to the degree of each sampled node v, with constant probability. Repeating this procedure $O(\log n)$ times, and, as in the Count-Min sketch, taking the minimum degree estimate for each node sampled at rate $1/\gamma^j$, gives us high probability approximation for such nodes.

Neighborhood Size Estimation. The final step is to implement an algorithm that can estimate the neighborhood size of the random subset of nodes S, to be used in our degree estimation procedure. We do this in Section 3.1, where we present Algorithm Neighborhood-Size (Algorithm 1). This algorithm takes as input two disjoint subsets L, R and returns a $(1 \pm \epsilon)$-approximation for the size of the neighborhood of L in R. We highlight that this may be very different than the *number of edges connecting* L *to* R – the neighborhood size is the number of nodes in R with at least one edge to L. This difference is critical in removing the correlations discussed previously due to common neighbors. Such correlations lead to the adaptive nature of prior algorithms [10, 26]. To estimate the size of the neighborhood of L in R, we sample the nodes in R at different rates and ask BIS queries on L and the sampled subset of R. Intuitively, when the sampling rate is the inverse of the size of the neighborhood, we will observe a '1' response with constant probability. We can detect this and thus estimate the neighborhood size.

Non-adaptivity. The approach described above is inherently non-adaptive. All random sampling of nodes and random subsets can be formed ahead of time, independently of any query responses. The only catch is that to determine which nodes should be recovered at each sampling rate, i.e., those nodes with degree $d(v) \approx \epsilon^{-2} \cdot m/\gamma^j$, we need a coarse estimate to the edge count m in the first place. Fortunately, we can bootstrap such an estimate starting with a coarse $O(\log^2 n)$-relative error approximate estimation, due to Beame et al. [10]. We then refine this estimate iteratively using Algorithm Refine-Estimate (Algorithm 4). Each refinement improves the approximation factor by ϵ, and after $O(\log_{1/\epsilon} \log n)$, refinements our estimate will result in a $(1 \pm \epsilon)$-approximation factor. The key observation here is that each refine step does not require any additional BIS queries. Thus, our algorithm remains non-adaptive.

2.2 Uniform Edge Sampling and Connectivity

In the full version [2], we prove Theorem 2 by designing and analyzing a non-adaptive algorithm for returning a near-uniform sample among the edges of the graph. Our approach builds heavily on our edge estimation algorithm. If we knew the degree $d(v)$ of all vertices, then to sample a uniform edge, we could sample a vertex $v \in V$ with probability $d(v)/\sum_{w \in V} d(w) = \frac{d(v)}{2m}$ and return a uniform neighbor among the neighbors of v. We can observe that the probability that an edge (v, u) is sampled is $\frac{d(v)}{2m} \cdot \frac{1}{d(v)} + \frac{d(u)}{2m} \cdot \frac{1}{d(u)} = \frac{1}{m}$. I.e., this approach yields a uniformly random edge sample.

Node Sampling. We implement the above approach approximately using BIS queries. First, observe that recovered vertices in our edge estimation algorithm are sampled with probabilities roughly proportional to their degrees. We argue that we can select a random vertex from this set, which overall is equal to any vertex v with probability approximately $\frac{d(v)}{2m}$. To do so, we leverage our degree estimates, and the fact that our edge count estimator, which is the sum of scaled degrees of recovered vertices, is well-concentrated.

Random Neighbor Sampling. It remains to show how to return a uniformly random neighbor of the sampled vertex. Similar to our edge estimation procedure, we design an algorithm which takes as input two disjoint subsets L, R and returns a uniform neighbor of L in R. To do so, we demonstrate an equivalence between the substantially more powerful OR queries and BIS queries in this specific setting, and argue that an existing algorithm for OR queries can be extended to return a uniform neighbor using BIS queries. An OR query takes as input a subset of pairs of vertices and returns '1' iff there is an edge in the subset queried. Building on this, in the full version [2], we present Algorithm Uniform-Neighbor that takes as input the subset of nodes sampled at any rate $1/\gamma^j$ as in our edge estimation algorithm, and approximately returns a uniform neighbor for every vertex v sampled in this set. Similar to Estimate-Degree (Algorithm 2), we construct $\widetilde{O}(1/\epsilon^4)$ random partitions of the sampled nodes. For every vertex v in a random subset S, we return a uniform neighbor (obtained using the idea just described) of the partition of S, as the neighbor of v. If v has large degree compared to the total degree of nodes in S, which it will if it is meant to be recovered at that sampling rate, this output will most likely be a neighbor of v, and will be close to a uniformly random one.

A Two-Round Algorithm for Connectivity. Our non-adaptive edge sampling algorithm (Theorem 2) directly yields a two-round algorithm for graph connectivity (Theorem 3), improving on a prior three-round algorithm of [8]. In particular, the algorithm of [8] selects

$O(\log^2 n)$ random neighbors per vertex, and contracts the connected components of this random graph into *supernodes*. This random sampling step can be performed using one round of $\widetilde{O}(n)$ BIS queries. They prove that in the contracted graph on the supernodes, there are at most $O(n \log n)$ edges. Using this fact, they then show how to identify whether all the supernodes are connected using $\widetilde{O}(n)$ BIS queries and two additional rounds of adaptivity.

We follow the same basic approach: using a first round of $\widetilde{O}(n)$ queries to randomly sample $O(\log^2 n)$ neighbors per vertex and contract the graph into supernodes. Once this is done, we observe that we have BIS query access to the contracted graph simply by always grouping together the set of nodes in each supernode. So, we can directly apply the non-adaptive sampling algorithm of Theorem 2 to sample edges from the contracted graph. By a coupon collecting argument, drawing $O(n \log^2 n)$ near-uniform edge samples (with replacement) from the contracted graph suffices to recover all $O(n \log n)$ edges in the graph, and thus determine connectivity of the contracted graph, and, in turn, the original graph.

2.3 Notation

Let $G(V, E)$ denote an undirected graph on vertex set V with edges $E \subseteq V \times V$. Let $|V| = n$ be the number of nodes and $|E| = m$ be the number of edges . For any set of nodes $S \subseteq V$, let $E[S] \subseteq E$ denote the edges in the induced subgraph on S. For any two disjoint sets of nodes $L, R \subseteq V$, let $E[L, R] = \{(u, v) \in E \mid u \in L, v \in R\}$ denote the edges between them. For any $v \in V$, let $\Gamma(v) = \{u \mid (u, v) \in E \text{ for some } v \in V\}$ be its set of neighbours. Let $d(v) = |\Gamma(v)|$ be its degree. For $S \subseteq V$, let $\Gamma(S) = \bigcup_{u \in S} \Gamma(u)$ and let $d(S) = \sum_{u \in S} d(u)$.

3 Non-adaptive algorithm for edge estimation

In this section, we present our non-adaptive algorithm for edge estimation using BIS queries. In Section 3.1, we describe an algorithm that takes as input two disjoint subsets L, R and returns an estimate of the size of the neighborhood $|\Gamma(L) \cap R|$. Next, in Section 3.2, we use this algorithm to give additive error approximations of degrees of all the vertices in a given subset. Finally, in Section 3.3, using the approximate degree estimates, we construct a $(1 \pm \epsilon)$-approximate estimator for m by sampling nodes with probabilities roughly proportional to their degrees. Missing details are included in the full version [2].

3.1 Estimating the size of neighborhood

Neighborhood-Size (Algorithm 1) takes as input two disjoint subsets $L, R \subseteq V$ and returns a $(1 \pm \epsilon)$-approximation of the size of neighborhood of L in R, i.e., $|\Gamma(L) \cap R|$ using $\text{poly}(1/\epsilon, \log n)$ BIS queries. We overview the analysis of this algorithm here.

The main idea is to sample subsets of vertices in R (denoted $\widehat{R}_1, \widehat{R}_2, \ldots$) with exponentially decreasing probability values $1/2, 1/4, 1/8, \ldots$. When the sampling rate $1/2^i$ falls below $1/|\Gamma(L) \cap R|$, we expect L to no longer have any neighbors in $\widehat{R}_i$ with good probability. In particular, we can return the inverse of the smallest probability $1/2^i$ for which $\mathcal{BIS}(L, \widehat{R}_i) =$ '1', as a coarse estimate for $|\Gamma(L) \cap R|$.

To boost the accuracy of this estimate, we repeat the process $T = O(\epsilon^{-2} \log(\delta^{-1} \cdot \log n))$ times, and at each sampling rate count the number of times the BIS query $\mathcal{BIS}(L, \widehat{R}_i)$ returns '1'. This count is denoted $\texttt{count}(i)$ in Algorithm 1, and its expectation can be written in closed form as $\mathbf{E}[\texttt{count}(i)] = T \cdot (1 - 1/2^i)^{|\Gamma(L) \cap R|}$. Suppose $2^{\widehat{i}} \leq |\Gamma(L) \cap R| < 2^{\widehat{i}+1}$, then, $\mathbf{E}[\texttt{count}(\widehat{i})] = \Theta(T)$. Via a standard Chernoff bound, it will be approximated to $(1 \pm \epsilon)$ error with high probability by $\texttt{count}(\widehat{i})$. Thus, we can compute an accurate estimate of the

neighborhood size by inverting our estimate of $\mathbf{E}[\texttt{count}(\widehat{i})]$, as $\log_{(1-1/2^{\widehat{i}})}(\texttt{count}(\widehat{i})/T)$. We identify the appropriate $\widehat{i}$ in line 12 of Algorithm 1, and compute the corresponding estimate in lines 13-14. There is one edge case handled in line 13: if $|\Gamma(L) \cap R| = 1$ we will have $\widehat{i} = 0$, and $\texttt{count}(\widehat{i}) = 0$. The final error bound for Algorithm 1 is stated below.

▶ **Lemma 4.** *Algorithm 1 uses $O(\epsilon^{-2}\log n \log(\delta^{-1} \cdot \log n))$ BIS queries and returns an estimate $\eta_{est}(L)$ of $|\Gamma(L) \cap R|$ such that with probability at least $1 - \delta$,*

$$(1-\epsilon) \cdot |\Gamma(L) \cap R| \leq \eta_{est}(L) \leq (1+\epsilon) \cdot |\Gamma(L) \cap R|.$$

Algorithm 1 NEIGHBORHOOD-SIZE: Estimating the neighborhood size of L in R.

Input: $L, R \subseteq V$, approximation error ϵ, failure probability δ.
Output: $\eta_{\text{est}}(L)$ as an estimate of $|\Gamma(L) \cap R|$.

1: Initialize $\eta_{\text{est}}(L) \leftarrow 0$.
2: **for** $i = 0, 1, \ldots \log_2 n$ **do**
3: $\quad \texttt{count}(i) \leftarrow 0$.
4: $\quad$ **for** $t = 1, 2, \ldots T = 2e^8 \ln(\log n/\delta) \cdot \epsilon^{-2}$ **do**
5: $\quad\quad \widehat{R}_i^t \leftarrow \{u \in R \mid u \text{ is included independently with probability } 1/2^i\}$.
6: $\quad\quad \texttt{count}(i) = \texttt{count}(i) + \mathcal{BIS}(L, \widehat{R}_i^t)$
7: $\quad$ **end for**
8: **end for**
9: **if** $\texttt{count}(0) = T$ **then**
10: $\quad$ **return** $\eta_{\text{est}}(L) = 0$.
11: **else**
12: $\quad$ Set $\widehat{i} \leftarrow \max\left\{i \mid \frac{\texttt{count}(i)}{T} < \frac{(1-\epsilon)}{2e^2}\right\}$.
13: $\quad$ **if** $\widehat{i} = 0$ **then return** $\eta_{\text{est}}(L) = 1$.
14: $\quad$ **else return** $\eta_{\text{est}}(L) = \log_{(1-1/2^{\widehat{i}})}(\texttt{count}(\widehat{i})/T)$.
15: $\quad$ **end if**
16: **end if**

3.2 Finding good approximation for degrees of vertices

We now describe how to use NEIGHBORHOOD-SIZE (Algorithm 1) to estimate the degrees of all vertices in a given subset $S \subseteq V$ up to additive error depending on the total degree of S. Our approach is inspired by the Count-Min sketch algorithm [24]. We randomly partition S into subsets $S^1, S^2, \ldots, S^\lambda$ where $\lambda = O(\epsilon^{-3}\log^2 n)$. The choice of the parameter λ is based on the analysis in Section 3.3. For each S^i, we estimate the size of the neighborhood of S^i in $V \setminus S^i$ using NEIGHBORHOOD-SIZE. We then return this neighborhood size estimate as the degree estimate for all vertices in S^i. For $v \in S^i$, $|\Gamma(S^i) \cap V \setminus S^i|$ is nearly an overestimate for $d(v)$, as long as v has few neighbors in S^i, which it will with high probability. Additionally, it is not too large an overestimate – we can observe that $|\Gamma(S^i) \cap V \setminus S^i| - d(v) \leq d(S^i \setminus v)$. I.e., the error in the overestimate is at most the total degree of the other nodes in S^i. In expectation, this error is at most $\frac{d(S)}{\lambda} = O\left(d(S) \cdot \frac{\epsilon^3}{\log^2 n}\right)$ due to our random choice of S^i.

As in the Count-Min sketch algorithm, to obtain high probability estimates, we repeat the process $T = O(\log n)$ times and assign the minimum among the neighborhood estimates as the degree estimate of $d(v)$. The full approach is given in Algorithm 2 (ESTIMATE-DEGREE) and the error bound in Lemma 5 below. We set the failure probability, $\delta = O(\epsilon^3/\log^4 n)$, for each of the calls to NEIGHBORHOOD-SIZE, to ensure that the total failure probability of Algorithm 2 is at most $O(1/\log n)$. As we make at most $T \cdot \lambda$ calls to the Algorithm NEIGHBORHOOD-SIZE, the total BIS queries used is $O(\log n \cdot \epsilon^{-3}\log^2 n \cdot \epsilon^{-2}\log n \log(\log^4 n \cdot \log n)) = O(\epsilon^{-5}\log^4 n \log(\log n))$. Formally, we have:

▶ **Lemma 5.** *For any $S \subseteq V$, Algorithm 2 uses $O(\epsilon^{-5} \log^4 n \log(\log n))$ BIS queries and with probability $1 - O(1/\log n)$, returns degree estimates $\widehat{d}(v)$ for every $v \in S$ satisfying:*

$$d(v)(1-\epsilon) \leq \widehat{d}(v) \leq d(v) + \frac{\epsilon^3}{\log^2 n} \cdot d(S).$$

Algorithm 2 ESTIMATE-DEGREE: Obtain additive approximate degree estimates.

Input: S is a subset of V, ϵ is approximation error.
Output: Degree estimates of vertices in S.

Initialize $\widehat{d}(v) \leftarrow n$ for every $v \in S$.
for t in $\{1, 2, \ldots, O(\log n)\}$ **do**
  Form a random partition of S into $S^{t,1}, S^{t,2}, \ldots S^{t,\lambda}$ where $\lambda = O(\epsilon^{-3} \log^2 n)$.
  for every set $S^{t,a}$ where $a \in [\lambda]$ **do**
    $\eta_{\text{est}}(S^{t,a}) \leftarrow$ NEIGHBORHOOD-SIZE$(S^{t,a}, V \setminus S^{t,a}, \epsilon/3, \delta)$, where $\delta = O(\epsilon^3/\log^4 n)$.
    For all $v \in S^{t,a}$, set $\widehat{d}(v) \leftarrow \min\{\widehat{d}(v), \eta_{\text{est}}(S^{t,a})\}$.
  end for
end for
return $\widehat{d}(v)$ for every $v \in S$.

3.3 Edge Estimation

In this section, we describe the algorithm EDGE-ESTIMATOR (Algorithm 3) that obtains a $(1 \pm \epsilon)$-approximation for the number of edges m. Missing details are presented in the full version [2].

Our Approach. A naive strategy to estimate the number of edges m is to sample $\widetilde{O}(\epsilon^{-2})$ nodes uniformly, and estimate m as $n/2$ times the average degree of the sampled nodes. However, the variance of such an estimator depends on the maximum degree, which could be as high as n. To fix this issue, we would like to sample vertices with probabilities proportional to their degrees. In particular, we sample vertices at different rates $1/\gamma^j$, where $\gamma > 1$ is a constant and $j \in \{0, 1, \cdots, \log n\}$. We use the term j^{th} level to refer to the sampling rate $1/\gamma^j$. Our estimator is given by: $\widehat{m} = \sum_v \mathbb{I}[v \text{ sampled}] \cdot d(v)/p_v$, be the appropriately weighted average of the sampled degrees. It is straightforward to show that $\mathbf{E}[\widehat{m}] = \sum_v d(v) = 2m$ and as argued in section 1, it is also concentrated around $2m$ with high probability. It is easy to observe that when a vertex v is sampled at rate $\widetilde{O}(\epsilon^{-2} d(v)/m)$, its contribution to $\widehat{m}$ is $\widetilde{O}(\epsilon^2 m)$. In other words, we need to detect the event that $d(v) \approx \epsilon^2 m/\gamma^j$, for some sampling level j. If we identify $\widetilde{O}(\epsilon^{-2})$ such vertices, $\widehat{m}$ will be an accurate estimate of the total edges, after appropriate scaling. However, there are three main challenges in implementing this approach which we detail below.

Approximate degrees. Algorithm ESTIMATE-DEGREE returns degree estimates with an additive approximation error of $O(\epsilon^3 \log^{-2} n \cdot d(S_j))$ at sampling level j. $\widetilde{O}\left(\epsilon^3 m/\gamma^j\right)$ at sampling level j. It is easy to see that $\mathbf{E}[d(S_j)] = O(m/\gamma^j)$. From Markov's inequality and union bound, we have that: $d(S_j) = O(m \log n/\gamma^j)$ for all $j \in [L]$ with probability at least $3/4$. Therefore, the additive approximation error term is $\widetilde{O}(\epsilon^3 \cdot m/\gamma^j)$. To include the contribution of a vertex v in the estimator, we must ensure that this error term is small in a relative sense – i.e., at most $O(\epsilon \cdot d(v))$. This holds whenever $d(v) = \widetilde{\Omega}(\epsilon^2 m/\gamma^j)$. Observe that this corresponds to the threshold we mentioned earlier. Therefore, our goal is to identify all vertices at every level j that pass the threshold of $\widetilde{\Omega}(\epsilon^2 m/\gamma^j)$. When that happens, we say that the vertex v has been recovered at level j and can be safely included in our estimator.

Knowledge of m. As we do not know the value of m, we start with an $O(\log^2 n)$-relative error approximate estimate, obtained using the Algorithm CoarseEstimator in Beame et al. [10], as follows: Given a partition $L, R \subseteq V$, Algorithm CoarseEstimator returns an approximate estimate for the number of edges between L and R, given by: $\frac{m(L,R)}{8 \log n} \leq \bar{m}(L,R) \leq 8 \log n \cdot m(L,R)$, where $m(L,R) = |E[L,R]|$. However, using a simple reduction, we can convert this estimate into an $O(\log^2 n)$-relative error approximation for total edges in the graph G. First, we partition the graph uniformly into two sets of vertices L and R. We set our estimate for edges as: $\bar{m} = O(\log n) \cdot \bar{m}(L,R)$. From Lemma 3.2 in [10], $m(L,R) = \Theta(m)$ when $m \geq 2$, with constant probability. So, our initial estimate for refinement, $\bar{m}$, satisfies: $m \leq \bar{m} \leq O(\log^2 n) \cdot m$.

We repeatedly refine the approximate estimate using Algorithm Refine-Estimate, until we get a $(1 \pm \epsilon)$-relative error approximation of m. Each *refinement* improves the approximation factor from the previous stage by a multiplicative factor of ϵ. We note that each refinement does not require any additional BIS queries and uses the available approximate degree estimates.

Boundary Vertices. We partition the space of possible degrees, i.e., $[0, n)$ into geometrically decreasing partitions, called *levels*. Each level is represented by an integer in $[0, L]$. We recover a vertex at a particular level $j \in [0, L]$, if it is sampled with probability corresponding to the level, i.e., γ^{-j} and it passes the threshold mentioned earlier. It is possible that some vertices have degrees close to the threshold values at each sampling level. We denote the set of such vertices as $V_{\text{boundary}} \subseteq V$. For such boundary vertices, as we use approximate degree estimates, they might be recovered at a level different from their true levels (defined with respect to exact degrees). Such a scenario could potentially affect the contribution of the recovered vertex in our estimator by an additional multiplicative factor dependent on γ and the difference between recovered level and true level. As a result, our estimator might not be a $(1 \pm \epsilon)$-relative error approximation anymore. We get around this limitation by dividing the region between any two consecutive levels into $B = O(1/\epsilon)$ buckets, where ϵ denotes the approximation parameter, and shifting the boundaries of all the levels by a random shift selected uniformly from the first B buckets. We account for this by changing the sampling rates to $\gamma^{-\mu(j)}$ where $\mu(j)$ encodes the random shift. We set the parameter corresponding to the sampling probability, $\gamma = 1/(1-\epsilon)$.

With the random shift of the level boundaries, we ensure that every vertex will lie close to the boundary with probability at most ϵ. Moreover, we argue that every boundary vertex is recovered at its true level or level adjacent to its true level. Therefore, the total contribution of V_{boundary} to our edge estimator is $O(\epsilon m)$.

Random Boundary Shift. The region between two consecutive levels is divided into B buckets with the boundaries of buckets proportional to the values given by: $\{[1/\gamma^B, 1/\gamma^{B-1}), \cdots, [1/\gamma^2, 1/\gamma), [1/\gamma, 1)\}$. We select a random integer offset for shifting our levels, denoted by s, which is selected uniformly at random from $[0, B)$. Now, the level boundaries are located at values proportional to $\gamma^{-\mu(j)}$ where $\mu(j) = j \cdot B - s$ and $0 \leq j \leq L$. Observe that the number of sampling levels is given by $L = \frac{2}{B} \cdot \log_\gamma n + 1 \leq \log n + 1$. Combining everything, the exact level boundaries are dependent on the estimate $\bar{m}$ and given by $O\left(\frac{\bar{m}}{\gamma^{\mu(j)}} \cdot \frac{\epsilon^2}{\log n}\right)$, for every $j \in [0, L]$.

3.3.1 Overview of Algorithm Edge-Estimator

In Algorithm Edge-Estimator, we construct sets $V = S_0 \supseteq S_1 \supseteq \cdots \supseteq S_L$ where a set S_j (for all $j \geq 2$) is obtained by sampling vertices in S_{j-1} with probability $1/\gamma^B$. The set S_1 is obtained by sampling vertices in V with probability $1/\gamma^{-s+B}$. Our sampling scheme results in each vertex being included in a set S_j with probability $1/\gamma^{\mu(j)}$. As described previously, with constant probability, $d(S_j) = O(m \log n/\gamma^{\mu(j)})$, for all j. Using Algorithm 2, we obtain approximate degree estimates of vertices in S_j for every sampling level $j \leq L$ with an approximation error of $O\left(\epsilon^3/\log^2 n \cdot d(S_j)\right) = O\left(m\epsilon^3/\gamma^{\mu(j)} \log n\right)$. In order that this error is small, we need to recover vertices $v \in S_j$ with high degree, such that $d(v) = \widetilde{\Omega}(\frac{m}{\gamma^{\mu(j)}} \cdot \frac{\epsilon^2}{\log n})$. As we do not know m, we bootstrap it with a $O(\log^2 n)$-relative error approximate estimate due to [10], denoted by $\bar{m}_0$. We repeatedly refine the estimate $T = 2\log_{1/\epsilon} \log n$ times, using Refine-Estimate (Algorithm 4), where the estimate $\bar{m}_{t-1}$ is used to construct an improved estimate $\bar{m}_t$. We return the estimate $\bar{m}_T$ as our final estimate for m. The constants used c_1, c_2 satisfy $c_1 \leq c_2/10$ and $c_2 \geq 50$.

Algorithm 3 Edge-Estimator: Non-adaptive algorithm for estimating edges.

Input: V set of n vertices and $\epsilon > 0$ error parameter.
Output: Estimate $\widehat{m}$ of number of edges in G.

1: Scale $\epsilon \leftarrow \frac{\epsilon}{600 \log_{1/\epsilon} \log n}$ and initialize $\gamma \leftarrow 1/(1-\epsilon)$ and $B \leftarrow 2/\epsilon$.
2: Let s be an integer selected uniformly at random from the interval $[0, B)$.
3: Let $\mu(j) \leftarrow -s + j \cdot B$ for every integer j in the interval $\left[0, \frac{2}{B} \cdot \log_\gamma n + 1\right]$.
4: Initialize $S_0 \leftarrow V$ and construct S_1 by sampling vertices in S_0 with probability $1/\gamma^{\mu(1)}$.
5: Construct $S_2 \supseteq \ldots \supseteq S_L$ for $L = \frac{2}{B} \cdot \log_\gamma n$ where each S_j is obtained by sampling vertices in S_{j-1} $\forall j \geq 2$, independently with probability $1/\gamma^B$.
6: **for** $j = 0, 1, \ldots L$ **do**
7: Run Estimate-Degree (S_j) to obtain the estimates $\widehat{d}_j(v)$ for all $v \in S_j$ satisfying:

$$(1-\epsilon)d(v) \leq \widehat{d}_j(v) \leq d(v) + \frac{c_1\epsilon^3 \cdot m}{\log n \cdot \gamma^{\mu(j)}}.$$

8: **end for**
9: Construct a random partition L, R of V. Let $\bar{m}_0$ be the $O(\log n)$-approximate estimate from the Algorithm CoarseEstimator in Beame et al. [10] on the partition L, R.
10: Set $\bar{m}_0 \leftarrow \max\{2, O(\log n) \cdot \bar{m}_0\}$, so that we have $m \leq \bar{m}_0 \leq O(\log^2 n) \cdot m$.
11: **for** $t = 1, 2, \cdots, T = 2\log_{1/\epsilon} \log n$ **do**
12: $\bar{m}_t$ is assigned the output of Refine-Estimate that takes as input approximate degree values $\widehat{d}_j(v)$ $\forall v \in S_j$ $\forall j \in [L]$, the previous estimate $\bar{m}_{t-1}$ and the iteration t.
13: **end for**
14: **return** $\widehat{m} \leftarrow \bar{m}_T$.

3.3.2 Overview of Algorithm Refine-Estimate

Suppose we are given an initial estimate $\bar{m}$ satisfying $m \leq \bar{m} \leq (1+\alpha)m$ for some unknown approximation factor α satisfying $\epsilon \leq \alpha \leq O(\log^2 n)$. We set the threshold value for recovering a vertex at a level j as $\frac{\bar{m}}{\gamma^{\mu(j)}} \cdot \frac{c_2\epsilon^2}{\log n}$ where c_2 is a constant. So, when a vertex v, that hasn't been recovered at a smaller level yet, with degree estimate $\widehat{d}_j(v)$ (obtained from Algorithm 3) satisfies $\widehat{d}_j(v) \geq \frac{\bar{m}}{\gamma^{\mu(j)}} \cdot \frac{c_2\epsilon^2}{\log n}$, we set the level of recovery $\widehat{\ell}(v) = j$ and recovered flag $r(v) = 1$.

From construction, we can observe that once a vertex is recovered at a particular level it is not available to be recovered at higher level later. Our estimator is the summation of terms $\gamma^{\mu(\widehat{\ell}(v))} \cdot \widehat{d}(v)$ for every v satisfying $r(v) = 1$. We normalize $\widehat{m}$ by adding an additional term of $(\epsilon \log\log n)^t \bar{m}_0$ in iteration t, to ensure that after every call to Refine-Estimate (Algorithm 4), the estimate returned $\widehat{m}$, satisfies: $m \leq \widehat{m} \leq m(1+\epsilon \cdot \log\log n \cdot \alpha)$ (see the full version for additional details [2]). After $T = 2\log_{1/\epsilon} \log n$ iterations of Refine-Estimate, we return the estimate $\bar{m}_T$ as our final estimate in Edge-Estimator (Algorithm 3).

Algorithm 4 Refine-Estimate: Refines the current estimate of number of edges.

Input: $\bar{m}$ satisfying $m \leq \bar{m} \leq m(1+\alpha)$, approximate degree values $\widehat{d}_j(v)\ \forall v \in S_j\ \forall j \in [L]$ obtained using Algorithm 3, $\bar{m}_0$, and iteration t.
Output: Estimate $\widehat{m}$ satisfying $m \leq \widehat{m} \leq m(1+\epsilon \cdot \alpha \cdot \log\log n)$ of number of edges in G.

Initialize $\widehat{m} \leftarrow 0$.
Initialize $r(v) \leftarrow 0$ for all v (indicator if v has been recovered yet).
for $j = 0, 1, \ldots L$ **do**
  for $v \in S_j$ **do**
    if $r(v) = 0$ and $\widehat{d}_j(v) \geq \frac{\bar{m}}{\gamma^{\mu(j)}} \cdot \frac{c_2 \epsilon^2}{\log n}$ **then**
      $\widehat{m} \leftarrow \widehat{m} + \gamma^{\mu(j)} \cdot \widehat{d}_j(v)$.
      $\widehat{\ell}(v) \leftarrow j$ and $r(v) \leftarrow 1$. ▷ Used in the analysis.
    end if
  end for
end for
if $t < T = 2\log_{1/\epsilon} \log n$ **then**
  $\widehat{m} = \widehat{m}/2 + (\epsilon \log\log n)^t \bar{m}_0$. ▷ We normalize $\widehat{m}$ so that we have $\widehat{m} \geq m$.
else
  $\widehat{m} = \widehat{m}/2$.
end if
return $\widehat{m}$.

3.3.3 Final Guarantees of Edge-Estimator

Using Bernstein's inequality, we argue that in iteration t, we can improve the approximation factor of the previous estimate $\bar{m}_{t-1}$ by a multiplicative factor of ϵ in the new estimate $\bar{m}_t$. After $T = O(\log_{1/\epsilon} \log n)$ iterations, the edge estimate will be a $(1 \pm \epsilon)$-relative error approximation satisfying:

▶ **Theorem 6.** *Given a graph G with n nodes and m edges, there is an algorithm that makes $O(\epsilon^{-5} \log^5 n \log^5(\log n) \log(\epsilon^{-1} \log n))$ non-adaptive BIS queries to G and returns an estimate $\widehat{m}$ satisfying: $m(1-\epsilon) \leq \widehat{m} \leq m(1+\epsilon)$,* with probability at least 3/5.

4 Uniform Edge Sampling

In this section, we give brief overview of an algorithm that returns a near-uniformly sampled edge from the graph using $\text{poly}(\log n, 1/\epsilon)$ BIS queries. We present the complete details in the full version [2]. Our algorithm extends Edge-Estimator (Alg. 3) and is based on the following idea. Suppose we know the degrees of all the vertices. In order to sample a uniform edge, we can sample a vertex v with probability $d(v)/\sum_{w \in V} d(w) = d(v)/2m$ and return a uniform neighbor among the neighbors of v. The probability that an edge $e = (v, u)$ is sampled is $d(v)/2m \cdot 1/d(v) + d(u)/2m \cdot 1/d(u) = 1/m$.

In order to return a uniform edge, using the above approach, to our setting, there are two challenges. First, we do not know the degrees (or approximate degrees) of *all the vertices*. This is because the set of recovered vertices in Refine-Estimate (Alg. 4) at a particular level j, i.e., $r(v) = 1$, is a subset of the sampled vertices S_j. Only the recovered vertices at any particular level have accurate degree estimates, i.e., $\widehat{d_j}(v) \approx (1 \pm \epsilon)d(v)$. Secondly, vertices are recovered at different levels and are therefore sampled with different probabilities. In order to return a uniform edge using the previously discussed idea, we must return a single vertex among the set of recovered vertices with probability dependent on the sampling probability at which the vertex was recovered, and it's approximate degree.

We address these two challenges, by suitably modifying Edge-Estimator (Alg. 3) and returning a vertex v, among the recovered vertices, with probability proportional to $\gamma^{\widehat{\ell}(v)} \cdot \widehat{d}_{\widehat{\ell}(v)}(v)$ where $\widehat{\ell}(v)$ is the level at which it is recovered, and $\widehat{d}_{\widehat{\ell}(v)}(v)$ is the degree estimate at the level of recovery. From Section 3.3, we know that our estimator: $\sum_{v \text{ is recovered}} \gamma^{\widehat{\ell}(v)} \cdot \widehat{d}(v)$, is concentrated around $2m$ (Theorem 6). Combining all the above, we obtain the following result about sampling an edge:

▶ **Theorem 7.** *Given a graph G with n nodes, m edges, and edge set E, there is an algorithm that makes $O(\epsilon^{-4}\log^6 n \log(\epsilon^{-1}\log n) + \epsilon^{-6}\log^5 n \log^6(\log n)\log(\epsilon^{-1}\log n))$ non-adaptive BIS queries which, with probability at least $1-\epsilon$, outputs an edge from a probability distribution P satisfying $(1-\epsilon)/m \le P(e) \le (1+\epsilon)/m$ for every $e \in E$.*

Graph Connectivity. Using the non-adaptive uniform sampling algorithm, we obtain a 2-round adaptive algorithm for determining graph connectivity, by building upon the work of [8]. See Section 2.2 for an outline of this result. Details are deferred to the full version [2].

▶ **Theorem 8.** *Given a graph G with n nodes, there is a 2-round adaptive algorithm that determines if G is connected with probability at least $1 - 1/n$ using $O(n\log^8 n \log\log n)$ BIS queries.*

5 Conclusion and Open Questions

In this paper, we presented the first $(1 \pm \epsilon)$ relative error non-adaptive algorithms for edge estimation and sampling using BIS queries. It would be interesting to investigate if better dependencies on ϵ than given by our algorithms can be obtained. Further, using Independent Set (IS) queries, *adaptive* algorithms for edge estimation with optimal query complexity $O(\min\{\sqrt{m}, n/\sqrt{m}\} \cdot \text{poly}(\log n, 1/\epsilon))$ were obtained only recently [10, 23]. It would be interesting to see if we can extend our techniques to study *non-adaptive* algorithms for edge estimation using IS queries or in the standard adjacency list query model, studied in the sublinear time graph algorithms literature.

References

1 Hasan Abasi and Bshouty Nader. On learning graphs with edge-detecting queries. In *Algorithmic Learning Theory (ALT)*, pages 3–30, 2019.

2 Raghavendra Addanki, Andrew McGregor, and Cameron Musco. Non-adaptive edge counting and sampling via bipartite independent set queries. *arXiv*, 2022. `arXiv:2207.02817`.

3 Arpit Agarwal, Shivani Agarwal, Sepehr Assadi, and Sanjeev Khanna. Learning with limited rounds of adaptivity: Coin tossing, multi-armed bandits, and ranking from pairwise comparisons. In *Proceedings of the 30th Annual Conference on Computational Learning Theory (COLT)*, pages 39–75, 2017.

4 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 459–467, 2012.

5 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.

6 Dana Angluin and Jiang Chen. Learning a hidden graph using $o(\log n)$ queries per edge. *Journal of Computer and System Sciences*, 74(4):546–556, 2008.

7 Boris Aronov and Sariel Har-Peled. On approximating the depth and related problems. *SIAM Journal on Computing*, 38(3):899–921, 2008.

8 Sepehr Assadi, Deeparnab Chakrabarty, and Sanjeev Khanna. Graph connectivity and single element recovery via linear and OR queries. In *29th Annual European Symposium on Algorithms, (ESA)*, pages 7:1–7:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

9 Eric Balkanski and Yaron Singer. The adaptive complexity of maximizing a submodular function. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1138–1151, 2018.

10 Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge estimation with independent set oracles. *ACM Transactions on Algorithms (TALG)*, 16(4):1–27, 2020.

11 Soheil Behnezhad. Time-optimal sublinear algorithms for matching and vertex cover. In *Proceedings of the 63rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 873–884, 2022.

12 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Hyperedge estimation using polylogarithmic subset queries. *arXiv*, 2019. `arXiv:1908.04196`.

13 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. On triangle estimation using tripartite independent set queries. *Theory of Computing Systems*, pages 1–28, 2021.

14 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Faster counting and sampling algorithms using colorful decision oracle. In *Proceedings of the 39th International Symposium on Theoretical Aspects of Computer Science (STACS)*, 2022.

15 Arijit Bishnu, Arijit Ghosh, Sudeshna Kolay, Gopinath Mishra, and Saket Saurabh. Parameterized query complexity of hitting set using stability of sunflowers. In *29th International Symposium on Algorithms and Computation*, 2018.

16 Arijit Bishnu, Arijit Ghosh, Gopinath Mishra, and Manaswi Paraashar. Efficiently sampling and estimating from substructures using linear algebraic queries. *arXiv*, 2019. `arXiv:1906.07398`.

17 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.

18 Sergio Cabello and Miha Jejčič. Shortest paths in intersection graphs of unit disks. *Computational Geometry*, 48(4):360–367, 2015.

19 Clément L Canonne and Tom Gur. An adaptivity hierarchy theorem for property testing. *Computational Complexity*, 27(4):671–716, 2018.

20 Amit Chakrabarti and Manuel Stoeckl. The element extraction problem and the cost of determinism and limited adaptivity in linear queries. *arXiv*, 2021. `arXiv:2107.05810`.

21 Chandra Chekuri and Kent Quanrud. Parallelizing greedy for submodular set function maximization in matroids and beyond. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, pages 78–89, 2019.

22 Chao L Chen and William H Swallow. Using group testing to estimate a proportion, and to test the binomial model. *Biometrics*, pages 1035–1046, 1990.

23 Xi Chen, Amit Levi, and Erik Waingarten. Nearly optimal edge estimation with independent set queries. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2916–2935, 2020.

24 Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

25 Holger Dell and John Lapinskas. Fine-grained reductions from approximate counting to decision. *ACM Transactions on Computation Theory (TOCT)*, 13(2):1–24, 2021.

26 Holger Dell, John Lapinskas, and Kitty Meeks. Approximately counting and sampling small witnesses using a colourful decision oracle. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2201–2211, 2020.

27 Robert Dorfman. The detection of defective members of large populations. *The Annals of Mathematical Statistics*, 14(4):436–440, 1943.

28 Dingzhu Du, Frank K Hwang, and Frank Hwang. *Combinatorial group testing and its applications*, volume 12. World Scientific, 2000.

29 Talya Eden, Dana Ron, and C Seshadhri. On approximating the number of k-cliques in sublinear time. *SIAM Journal on Computing*, 49(4):747–771, 2020.

30 Talya Eden and Will Rosenbaum. On sampling edges almost uniformly. In *1st Symposium on Simplicity in Algorithms (SOSA)*, 2018.

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

32 Aleksei V Fishkin. Disk graphs: A short survey. In *International Workshop on Approximation and Online Algorithms*, pages 260–264. Springer, 2003.

33 Oded Goldreich and Dana Ron. Approximating average parameters of graphs. *Random Structures & Algorithms*, 32(4):473–493, 2008.

34 Piotr Indyk, Hung Q Ngo, and Atri Rudra. Efficiently decodable non-adaptive group testing. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1126–1142, 2010.

35 Piotr Indyk, Eric Price, and David P Woodruff. On the power of adaptivity in sparse recovery. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 285–294, 2011.

36 Akshay Kamath and Eric Price. Adaptive sparse recovery with limited adaptivity. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2729–2744, 2019.

37 Howard Karloff, Siddharth Suri, and Sergei Vassilvitskii. A model of computation for mapreduce. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 938–948, 2010.

38 Lidiya Khalidah binti Khalil and Christian Konrad. Constructing large matchings via query access to a maximal matching oracle. In *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2020.

39 Andrew McGregor. Graph stream algorithms: a survey. *ACM SIGMOD Record*, 43(1):9–20, 2014.

40 Ashley Montanaro and Changpeng Shao. Quantum algorithms for learning a hidden graph. In *17th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

41 Vasileios Nakos, Xiaofei Shi, David P Woodruff, and Hongyang Zhang. Improved algorithms for adaptive compressed sensing. In *Proceedings of the 45th International Colloquium on Automata, Languages and Programming (ICALP)*, 2018.

42 Noam Nisan. The demand query model for bipartite matching. In *Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 592–599, 2021.

43 Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1123–1131, 2012.

44 Cyrus Rashtchian, David P Woodruff, and Hanlin Zhu. Vector-matrix-vector queries for solving linear algebra, statistics, and graph problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, 2020.

45 Dana Ron. Sublinear-time algorithms for approximating graph parameters. In *Computing and Software Science*, pages 105–122. Springer, 2019.

46 Dana Ron and Gilad Tsur. The power of an example: Hidden set size approximation using group queries and conditional sampling. *ACM Transactions on Computation Theory (TOCT)*, 8(4):1–19, 2016.

47 Aviad Rubinstein, Tselil Schramm, and S. Matthew Weinberg. Computing Exact Minimum Cuts Without Knowing the Graph. In *Proceedings of the 9th Conference on Innovations in Theoretical Computer Science (ITCS)*, 2018.

48 C Seshadhri. A simpler sublinear algorithm for approximating the triangle count. *arXiv*, 2015. `arXiv:1505.01927`.

49 Larry Stockmeyer. The complexity of approximate counting. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing (STOC)*, pages 118–126, 1983.

50 Larry Stockmeyer. On approximation algorithms for # p. *SIAM Journal on Computing*, 14(4):849–861, 1985.

51 Jakub Tětek and Mikkel Thorup. Edge sampling and graph parameter estimation via vertex neighborhood accesses. In *Proceedings of the 54th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1116–1129, 2022.