On the Complexity of Triangle Counting Using Emptiness Queries

Arijit Bishnu
Indian Statistical Institute, Kolkata, India

Arijit Ghosh
Indian Statistical Institute, Kolkata, India

Gopinath Mishra
University of Warwick, Coventry, UK

Abstract

Beame et al. [ITCS'18 & TALG'20] introduced and used the BIPARTITE INDEPENDENT SET (BIS) and INDEPENDENT SET (IS) oracle access to an unknown, simple, unweighted and undirected graph and solved the edge estimation problem. The introduction of this oracle set forth a series of works in a short time that either solved open questions mentioned by Beame et al. or were generalizations of their work as in Dell and Lapinskas [STOC'18 and TOCT'21], Dell, Lapinskas, and Meeks [SODA'20 and SICOMP'22], Bhattacharya et al. [ISAAC'19 & TOCS'21], and Chen et al. [SODA'20]. Edge estimation using BIS can be done using polylogarithmic queries, while IS queries need sub-linear but more than polylogarithmic queries. Chen et al. improved Beame et al.'s upper bound result for edge estimation using IS and also showed an almost matching lower bound. Beame et al. in their introductory work asked a few open questions out of which one was on estimating structures of higher order than edges, like triangles and cliques, using BIS queries.

In this work, we almost resolve the query complexity of estimating triangles using BIS oracle. While doing so, we prove a lower bound for an even stronger query oracle called Edge Emptiness (EE) oracle, recently introduced by Assadi, Chakrabarty, and Khanna [ESA'21] to test graph connectivity.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Lower bounds and information complexity

Keywords and phrases Triangle Counting, Emptiness Queries, Bipartite Independent Set Query

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2023.48

Category RANDOM

Related Version *Full Version*: `https://arxiv.org/abs/2110.03836`

Funding *Gopinath Mishra*: Research supported in part by the Centre for Discrete Mathematics and its Applications (DIMAP), by EPSRC award EP/V01305X/13.

1 Introduction

Motivated by connections to group testing, to emptiness versus counting questions in computational geometry, and to the complexity of decision versus counting problems, Beame et al. introduced the BIPARTITE INDEPENDENT SET (shortened as BIS) and INDEPENDENT SET (shortened as IS) oracles as a counterpoint to the local queries [18, 16, 19]. The BIS query oracle can be seen in the lineage of the query oracles [24, 25, 23] that go beyond the local queries. The local queries for a graph $G = (V(G), E(G))$ are: (i) DEGREE query: given $u \in V(G)$, the oracle reports the degree of u in $V(G)$; (ii) NEIGHBOUR query: given $(u \in V(G))$, the oracle reports the i-th neighbor of u, if it exists; otherwise, the oracle reports

NULL; (iii) ADJACENCY query: given u, $v \in V(G)$, the oracle reports whether $\{u, v\} \in E(G)$. There is another related query oracle RANDOMEDGE query: the oracle reports an edge uniformly at random when we query.[1]

Let us start by looking into the formal definitions of BIS and IS.

▶ **Definition 1.1** (BIPARTITE INDEPENDENT SET)**.** Given disjoint subsets $U, U' \subseteq V(G)$, a BIS query answers whether there exists an edge between U and U' in G.

▶ **Definition 1.2** (INDEPENDENT SET)**.** Given a subset $U \subseteq V(G)$, a IS query answers whether there exists an edge between vertices of U in G.

The introduction of this new type of oracle access to a graph spawned a series of works that either solved open questions [12, 13] mentioned in Beame et al. or were generalizations [13, 8, 6]. Beame et al. used BIS and IS queries to estimate the number of edges in a graph [4, 5]. One of their striking observations was that BIS queries were more effective than IS queries for estimating edges. This observation also fits in with the fact that IS queries can be simulated in a randomized fashion using polylogarithmic BIS queries.[2] Edge estimation using BIS was also solved in [12] albeit in a higher query complexity than [4]. There were later generalizations of the BIS oracle to estimate higher order structures like triangles and hyperedges [13, 7, 6]. On the IS front, Beame et al.'s result for edge estimation using IS oracle was improved in [11] with an almost matching lower bound and thereby resolving the query complexity of estimating edges using IS oracle. One can observe the interest generated in these (bipartite) independent set based oracles in a short span of time. The results are summarized in Table 1: a cursory glance would tell us that commensurate higher order queries were needed for estimating higher order structures (TRIPARTITE INDEPENDENT SET (shortened as TIS) for counting triangles, COLORFUL INDEPENDENCE ORACLE (shortened as CID) for counting hyperedges) if polylogarithmic number of queries is the benchmark. We provide the definitions of TIS and CID below.

▶ **Definition 1.3** (TRIPARTITE INDEPENDENT SET (TIS) [7])**.** Given three disjoint subsets A, B, C of the vertex set V of a graph $G(V, E)$, the TIS oracle reports whether there exists a triangle having endpoints in A, B and C.

▶ **Definition 1.4** (COLORFUL INDEPENDENCE ORACLE (CID) [9, 13])**.** Given d pairwise disjoint subsets of vertices $A_1, \ldots, A_d \subseteq U(\mathcal{H})$ of a hypergraph $\mathcal{H}$ ($U(\mathcal{H})$ is the vertex set of the hypergraph $\mathcal{H}$) as input, CID query oracle answers whether $m(A_1, \ldots, A_d) \neq 0$, where $m(A_1, \ldots, A_d)$ denotes the number of hyperedges in $\mathcal{H}$ having exactly one vertex in each A_i, $\forall i \in \{1, 2, \ldots, d\}$.

Notice the use of number of disjoint subsets in the definition of TIS and CID. That is why, we call TIS and CID as higher order query oracles than BIS and IS.

1.1 The open questions suggested by Beame et al. [4, 5]

For a work that has spawned many interesting results in such a short span of time, let us focus on the open problems and future research directions mentioned in [4, 5].

[1] Note that that, in RANDOMEDGE query, the probability space is the set of all edges. So, it is not actually a local query.

[2] Let us consider an IS query with input U. Let us partition U into two parts X and Y by putting each vertex in U to X or Y independently and uniformly at random. Then we make a BIS query with inputs X and Y, and report U is an independent set if and only if BIS reports that there is no edge with one endpoint in each of X and Y. Observe that we will be correct with at least probability $1/2$. We can boost up the probability by repeating the above procedure suitable number of times.

Table 1 The whole gamut of results involving LOCAL queries [17], BIS, IS and its generalizations. † Δ is the maximum number of triangles on an edge. ‡ Both these results estimate the number of hyperedges in a d-uniform hypergraph, where d is treated as a constant. Here n, m and T denote the number of vertices, edges and triangles in a graph G, respectively. $\widetilde{\mathcal{O}}(\cdot)$ and $\widetilde{\Omega}(\cdot)$ hide a multiplicative factor of $poly(\log n, 1/\varepsilon)$ and $1/poly(\log n, 1/\varepsilon)$, respectively.

Work	Oracle used	Structure estimated	Upper bound / Lower bound	Any other problem solved?
[19]	LOCAL	edge	$\widetilde{\mathcal{O}}\left(\frac{n}{\sqrt{m}}\right)$ / $\Omega\left(\frac{n}{\sqrt{m}}\right)$	Approximating average distance.
[11]	IS	edge	$\widetilde{\mathcal{O}}\left(\min\left\{\sqrt{m}, n/\sqrt{m}\right\}\right)$ / $\widetilde{\Omega}\left(\min\left\{\sqrt{m}, n/\sqrt{m}\right\}\right)$	– / –
[4]	BIS	edge	$poly(\log n, 1/\varepsilon) = \widetilde{\mathcal{O}}(1)$ / –	Edge estimation using IS queries.
[7]	TIS	triangle	$poly(\log n, \Delta, 1/\varepsilon)^\dagger$ / –	– / –
[13], [6] ‡	CID	hyperedge	$poly(\log n, 1/\varepsilon) = \widetilde{\mathcal{O}}(1)$ / –	[13] resolved Q2 in positive.
[14]	LOCAL	triangle	$\widetilde{\mathcal{O}}\left(\frac{n}{T^{1/3}} + \min\left\{m, \frac{m^{3/2}}{T}\right\}\right)$ / $\Omega\left(\frac{n}{T^{1/3}} + \min\left\{m, \frac{m^{3/2}}{T}\right\}\right)$	– / –
[2]	LOCAL+ RANDOM EDGE	triangle	$\widetilde{\mathcal{O}}\left(\min\left\{m, \frac{m^{3/2}}{T}\right\}\right)$ / $\Omega\left(\min\left\{m, \frac{m^{3/2}}{T}\right\}\right)$	Estimated number of arbitrary subgraphs.
This work	BIS	triangle	$\widetilde{\mathcal{O}}\left(\min\left\{\frac{m}{\sqrt{T}}, \frac{m^{3/2}}{T}\right\}\right)$ / $\widetilde{\Omega}\left(\min\left\{\sqrt{T}, \frac{m^{3/2}}{T}\right\}\right)$	– / –

Q1 Can we estimate the number of cliques using polylogarithmic number of BIS queries?

Q2 Can polylogarithmic number of BIS queries sample an edge uniformly at random?

Q3 Can BIS or IS queries possibly be used in combination with local queries for graph parameter estimation problems?

Q4 What other oracles, besides subset queries, allow estimating graph parameters with a polylogarithmic number of queries?

Answers to the above questions and a discussion

Only Q2 has been resolved till now in the positive [13] as can be observed from Table 1. At its core, Q1 asks if a query oracle can step up, that is, if it can estimate a structure that is of a higher order than what the oracle was designed for. The framing of Q1 seems that Beame et al. expected a polylogarithmic query complexity for estimation of the number of cliques using BIS. Pertinent to these questions, we also want to bring to focus a work [22] where the authors mention that it seems to them that estimation of higher order structures will require higher order queries (see the discussion after Proposition 23 of [22]). They showed that $\Omega(n^2/\log n)$ BIS queries are required to separate *triangle free* graph instances from graph instances having at least one triangle. This lower bound follows directly from the communication complexity of triangle freeness testing [3]. However, the full complexity of triangle estimation using emptiness queries like BIS remains elusive. It seems to us that the observations in [4, 5] and [22] about the power of BIS in estimating higher order structures stand in contrast. In this backdrop, we have almost resolved the lower bound for estimating triangles using BIS queries, and our upper bound result show that even if BIS can not estimate triangles using polylogarithmic queries but it is still more powerful than LOCAL + RANDOM EDGE queries on graph for estimating triangles (see Table 1). BIS has an inherent asymmetry in its structure in the following sense – when BIS says that there exists no edge between two disjoint sets, then BIS stands as a witness to the existence of two sets of vertices having no interdependence, while a yes answer implies that there can be any number of edges, varying from one to the product of the cardinality of the two sets, going across the two sets. We feel that this property of BIS gives it its power, but on the other hand, also makes it difficult to prove lower bounds. That is probably the reason why works related to upper bound for BIS and its generalizations exist, whereas works on lower bound were not forthcoming. Though not on BIS, the work of Chen et al. using IS queries gave an interesting lower bound construction for IS oracle. Our work goes one step further in being able to prove a lower bound for the BIS oracle which is much stronger than IS oracle. We have almost resolved the open question Q1 by proving almost matching lower and upper bounds involving BIS for estimating triangles.

1.2 A stronger oracle than BIS, our main result and its consequences

Now we define EDGE EMPTINESS (shortened as EE) query oracle which is stronger than both BIS and IS. The EDGE EMPTINESS query is a form of a *subset query* [24, 25, 23] where a subset query with a subset $P \subseteq U$ asks whether $P \cap T$ is empty or not, where T is also a subset of the universe U. The EDGE EMPTINESS query operates with U being the set of all vertex pairs in G, T being the set of edges E in G, and P being a subset of pairs of vertices of V.

▶ **Definition 1.5** (EDGE EMPTINESS). Given a subset $P \subseteq \binom{V(G)}{2}$, a EE query answers whether there exists an $\{u, v\} \in P$ such that $\{u, v\}$ is an edge in G.

EE query is recently introduce by Assadi et al. [1].[3] Note that BIS and IS queries can be simulated by an EE query. [4] We prove our lower bound in terms of the *stronger*[5] EE queries that will directly imply the lower bound in terms of BIS. But we prove matching upper bound in terms of BIS. Our main results are stated below in an informal setting. The formal statements are given in Theorems 3.1 and A.1.

▶ **Theorem 1.6** (Main lower bound (informal statement)). *Let m, n, $t \in \mathbb{N}$. Any (randomized) algorithm that has* EE *query access to a graph $G(V, E)$ with n vertices and $\Theta(m)$ edges, requires $\widetilde{\Omega}\left(\min\left\{\sqrt{t}, \frac{m^{3/2}}{t}\right\}\right)$* EE *queries to decide whether the number of triangles in G is at most t or at least $2t$.*

▶ **Theorem 1.7** (An upper bound (informal statement)). *There exists an algorithm, that has* BIS *query access to a graph $G(V, E)$, finds a $(1 \pm \varepsilon)$-approximation to the number of triangles in G with high probabilility, and makes $\widetilde{\mathcal{O}}\left(\min\left\{\frac{m}{\sqrt{T}}, \frac{m^{3/2}}{T}\right\}\right)$* BIS *queries in expectaton. Here m, n, T denote the number of vertices, edges and triangles in G.*

[3] Assadi et al. [1] named EE query as OR query in their paper.

[4] Let us consider a BIS query with inpus A and B. Let P be the set of vertex pairs with one vertex from each of A and B. We call EE oracle with input P, and report there is an edge having one vertex in each of A and B if and only if the EE oracle reports that there exists an $\{u, v\} \in P$ that forms an edge in G. Similarly, we can simulate an IS query with input U by using an EE query with input $P = \binom{U}{2}$.

[5] For an example to show how EE query can be powerful than that of BIS or IS, $\mathcal{O}(\log \log n)$ EE queries [24] are enough to estimate the number of edges in a graph, as opposed to high query complexity (compared to $\mathcal{O}(\log \log n)$) when we have BIS or IS queries (See Table 1).

Note that Edge Emptiness query is the *strongest* subset query on edges of the graph. Informally speaking, our lower bound states that no subset query on edges can estimate the number of triangles in a graph by using polylogarithmic queries. However, the results of Bhattacharya et al. [6] and Dell et al. [13] imply that polylogarithmic TIS queries are enough to estimate the number of triangles in the graph. Note that TIS query is also a subset query on triangles in the graph. To complement our lower bound result, we also give an algorithm (see Theorem 1.7) for estimating the number of triangles in a graph with BIS queries that matches our lower bound. Here we would also like to mention that the number of BIS queries our algorithm uses is less than that of the number of local queries [17] needed to estimate the number of triangles in a graph. This implies that we are resolving Q3 in positive in the sense that BIS queries are efficient queries for triangle estimation vis-a-vis local queries [14] coupled with even random edge queries [2] (see Table 1).

1.3 Notations

Throughout the paper, the graphs are undirected and simple. For a graph $G(V,E)$, $V(G)$ and $E(G)$ denote the set of vertices and edges, respectively; $|V(G)|=n$, $|E(G)|=m$ and the number of triangles is T, unless otherwise specified. We use $\binom{V(G)}{2}$ to denote the set of vertex pairs in G. Note that $E(G) \subseteq \binom{V(G)}{2}$. For $P \subseteq \binom{V(G)}{2}$, $V(P)$ represents the set of vertices that belong to at least one pair in P. The neighborhood of a vertex $v \in V(G)$ is denoted by $N_G(v)$, and $|N_G(v)|$ is called the degree of vertex v in G. $\Gamma(\{x,y\})$ denotes the set $N_G(x) \cap N_G(y)$, that is, the set of common neighbors of x and y in G. If $e=\{x,y\} \in E(G)$, $\Gamma(e)$ denotes the set of vertices that forms triangles with e as one of their edges. The induced degree of a vertex v in $Z \subseteq V(G) \setminus \{v\}$ is the cardinality of $N_G(v) \cap Z$. For $X \subseteq V(G)$, the subgraph of G induced by X is denoted by $G[X]$. Note that $E(G[X]) = \{\{x,y\} : x,y \in X\}$. For two disjoint sets A, $B \subset V(G)$, the bipartite subgraph of G induced by A and B is denoted by $G[A,B]$. Note that $E(G[A,B])$ is the set of edges having one vertex in A and the other vertex in B.

Throughout the paper, $\varepsilon \in (0,1)$ is the approximation parameter. When we say a is a $(1\pm\varepsilon)$-approximation of b, then $(1-\varepsilon)b \leq a \leq (1+\varepsilon)b$. Polylogarithmic means *poly* $(\log n, 1/\varepsilon)$. $\widetilde{\mathcal{O}}(\cdot)$ and $\widetilde{\Omega}(\cdot)$ hide a multiplicative factor of $poly(\log n, 1/\varepsilon)$ and $1/poly(\log n, 1/\varepsilon)$, respectively. We have avoided floor and ceiling for simplicity of presentation. The constants in this paper are not taken optimally. We have taken them to let the calculation work with clarity. However, those can be changed to other suitable and appropriate constants.

1.4 Paper organization

We start with the technical overview of our lower and upper bounds in Section 2.1 and Section 2.2, respectively. The detailed lower and upper bound proofs are in Section 3 and Appendix A, respectively. The missing proofs of Section 3 are presented in the full version of the paper [10].

2 Technical overview

In this section, we discuss about the techniques and proof overview, The overview of the lower bound is discussed in Section 2.1 and that of the upper bound is discussed in Section 2.2.

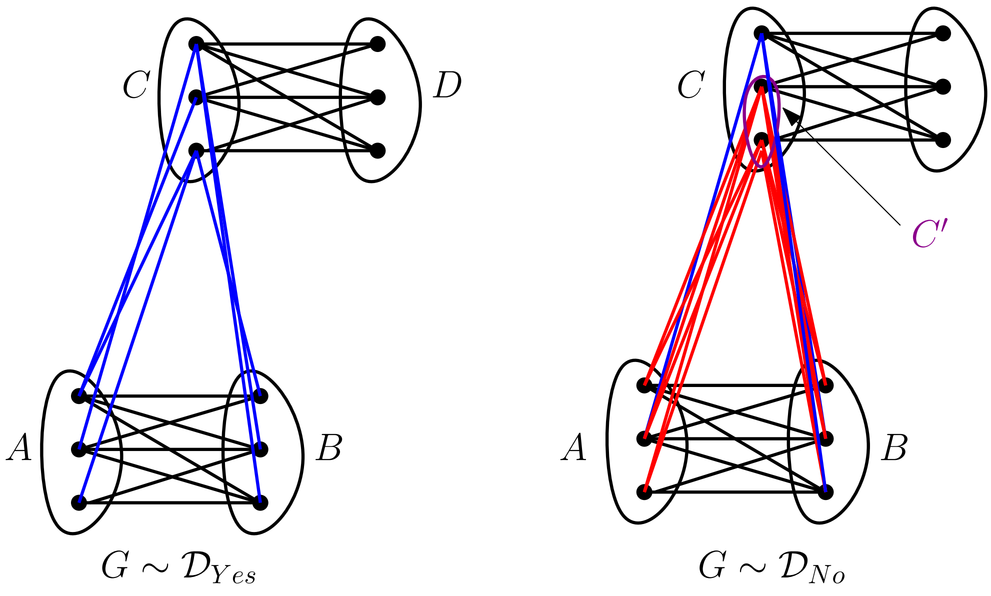

Figure 1 Illustration of $G \sim \mathcal{D}_{\mathbf{Yes}}$ and $G \sim \mathcal{D}_{\mathbf{No}}$ when $t = \Omega(m \log n)$.

2.1 Overview for the proof of our lower bound (Theorem 1.6)

Let us consider m, n, t as in Theorem 1.6. We prove the desired bound for BIS (stated in Theorem 1.6) by proving the lower bound is $\widetilde{\Omega}\left(\frac{m^{3/2}}{t}\right)$ when $t \geq km \log n$ and $\widetilde{\Omega}\left(\sqrt{t}\right)$ when $t < km \log n$ for EE query access, where k is a suitably chosen constant. In this Section, we discuss the overview of the proof when $t \geq km \log n$. The desired lower bound when $t < km \log n$ can be proved by using a reduction from the case when $t \geq km \log n$. The main intuition behind the lower bound is to "hide" a suitably generated vertex set such that a large number of queries is necessary to detect such a vertex.

The idea for the lower bound of $\widetilde{\Omega}\left(\frac{m^{3/2}}{t}\right)$ when $t \geq km \log n$

We prove by using Yao's method [21]. There are two distributions $\mathcal{D}_{\mathbf{Yes}}$ and $\mathcal{D}_{\mathbf{No}}$ (as described below) from which G is sampled satisfying $\mathbb{P}\left(G \sim \mathcal{D}_{\mathbf{Yes}}\right) = \mathbb{P}\left(G \sim \mathcal{D}_{\mathbf{No}}\right) = 1/2$. Note that, for each $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$, $|V(G)| = n = \Theta(\sqrt{m})$,[6] and $|E(G)| = \Theta(m)$ with a probability of at least $1 - o(1)$. But the number of triangles in each $G \sim \mathcal{D}_{\mathbf{No}}$ is at least two factor more than that of the number of triangles in any $G \sim \mathcal{D}_{\mathbf{Yes}}$, with a probability of at least $1 - o(1)$.

$\mathcal{D}_{\mathbf{Yes}}$: The vertex set $V(G)$ (with $|V(G)| = \Theta(\sqrt{m})$) is partitioned into four parts A, B, C, D uniformly at random. Vertex set A forms a biclique with vertex set B and vertex set C forms a biclique with vertex set D. Then every vertex pair $\{x, y\}$, with $x \in A \cup B$ and $y \in C$, is added as an edge to graph G with probability $\Theta\left(\sqrt{\frac{t}{m^{3/2}}}\right)$;

$\mathcal{D}_{\mathbf{No}}$: The vertex set $V(G)$ (with $|V(G)| = \Theta(\sqrt{m})$) is partitioned into four parts A, B, C, D uniformly at random. Vertex set A forms a biclique with vertex set B and vertex set C forms a biclique with vertex set D. Then every vertex pair $\{x, y\}$, with $x \in A \cup B$ and

[6] Without loss of generality, we assume that $\sqrt{m}$ is an integer. The proof can be extended to any graph having $n \geq \sqrt{m}$ vertices and m edges by adding $n - \sqrt{m}$ isolated vertices.

$y \in C$, is added as an edge to graph G with probability $\Theta\left(\sqrt{\frac{t}{m^{3/2}}}\right)$. Then each vertex of C is sampled with probability $\Theta\left(\frac{t}{m^{3/2}}\right)$. Let C' be the sampled set. Each vertex of C' is connected to every vertex of $A \cup B$ with an edge;

See Figure 1 for an illustration of the above construction. The constants, including k, in the order notations above are suitably set to have the following:

When $G \sim \mathcal{D}_{\text{Yes}}$: The number of triangles in each graph is at most t, with a probability of at least $1 - o(1)$;

When $G \sim \mathcal{D}_{\text{No}}$: $|C'| = \Theta\left(\frac{t}{m}\right)$ with a probability of at least $1 - o(1)$. Hence, the number of triangles in each $G \sim \mathcal{D}_{\mathbf{No}}$ is at least $2t$, with a probability of at least $1 - o(1)$.

Now, consider a particular EE query with input $P \subseteq \binom{V(G)}{2}$. Here, we divide the discussion into two parts, based on $|P| \leq \tau$ and $|P| > \tau$, where $\tau = \Theta(\log^2 n)$ is a threshold. If we query with the number of vertex pairs more than the threshold, chances are more we will not be able to distinguish between $G \sim \mathcal{D}_{\mathbf{Yes}}$ and $G \sim \mathcal{D}_{\mathbf{No}}$. When $|P| > \tau$, we can show that there exists a vertex pair $\{x, y\} \in P$ such that $\{x, y\}$ is an edge in G, with a probability of at least $1 - o(1)$, irrespective of whether $G \sim \mathcal{D}_{\mathbf{Yes}}$ or $G \sim \mathcal{D}_{\mathbf{No}}$. Intuitively, this is because the number of vertices and edges in G are $\Theta(\sqrt{m})$ and $\Theta(m)$, respectively. So, EE queries with input $P \subseteq \binom{V(G)}{2}$ such that $|P| > \tau$ will not be useful to distinguish whether $G \sim \mathcal{D}_{\mathbf{Yes}}$ or $G \sim \mathcal{D}_{\mathbf{No}}$.

We prove the desired lower bound by proving $\widetilde{\Omega}\left(\frac{m^{3/2}}{t}\right)$ EE queries are necessary to decide whether $G \sim \mathcal{D}_{\mathbf{Yes}}$ or $G \sim \mathcal{D}_{\mathbf{No}}$ with a probability of at least $2/3$. Note that $C' = \emptyset$ when $G \sim \mathcal{D}_{\mathbf{Yes}}$. So, the number of EE queries needed to decide whether $G \sim \mathcal{D}_{\mathbf{Yes}}$ or $G \sim \mathcal{D}_{\mathbf{No}}$, is at least the number of EE queries needed to *touch* at least one vertex of C' when $G \sim \mathcal{D}_{\mathbf{No}}$. Here, by touching at least a vertex of C', we mean $V(P) \cap C' \neq \emptyset$. As we have argued that only EE query with input $P \subseteq \binom{V(G)}{2}$ with $|P| \leq \tau$ can be useful, the probability that we touch a vertex in C' with such a query is at most $p = \mathcal{O}\left(\frac{C'}{\sqrt{m}} \cdot \tau\right) = \mathcal{O}\left(\frac{t \log^2 n}{m^{3/2}}\right)$. Hence the number of EE queries to touch at least a vertex of C', is at least $1/p$, that is, $\widetilde{\Omega}\left(\frac{m^{3/2}}{t}\right)$.

To let the the above discussion work, when $G \sim \mathcal{D}_{\mathbf{No}}$, $|C'| = \Theta\left(\frac{t}{m}\right)$ must be at least $\Omega(\log n)$. But $|C'| = \Theta\left(\frac{t}{m}\right)$ with a probability of at least $1 - o(1)$. Because of this, we take $t \geq km \log n$ in the above discussion. The formal statement of the lower bound, when $t \geq km \log n$, is given in Lemma 3.2 in Section 3. What we have discussed here is just an overview, the formal proof of Lemma 3.2 is much more invloved and delicate, which is presented in Section 3.1.

As we have already mentioned, the desired lower bound of $\widetilde{\Omega}\left(\sqrt{t}\right)$ when $t < km \log n$ can be proved by a reduction from the case when $t \geq km \log n$. The formal statement of the lower bound, when $t < km \log n$, is given in Lemma 3.3 in Section 3, and the proof is presented in Section 3.2.

2.2 Overview for our upper bound (Theorem 1.7)

We establish the upper bound claimed in Theorem 1.7 by giving two algorithms that report a $(1 \pm \varepsilon)$-approximation to the number of triangles in the graph:

(i) TRIANGLE-EST-HIGH(G, ε) that makes $\widetilde{\mathcal{O}}\left(\frac{m^{3/2}}{T}\right)$ BIS queries;

(ii) TRIANGLE-EST-LOW(G, ε) that makes $\widetilde{\mathcal{O}}\left(\frac{m+T}{\sqrt{T}}\right)$ BIS queries.

Informally speaking, our final algorithm TRIANGLE-EST calls TRIANGLE-EST-HIGH(G, ε) and TRIANGLE-EST-LOW(G, ε) when $T = \Omega(m)$ and $T = \mathcal{O}(m)$, respectively. Observe that, if TRIANGLE-EST knows T within a constant factor, then it can decide which one to use among TRIANGLE-EST-HIGH(G, ε) and TRIANGLE-EST-LOW(G, ε). If TRIANGLE-EST does not know T within a constant factor, then it starts from a guess $L = \binom{n}{3}/2$ and makes a geometric search on L until the output of TRIANGLE-EST is consistent with L. Depending on whether $L = \Omega(m)$ or $L = \mathcal{O}(m)$, TRIANGLE-EST decides which one among TRIANGLE-EST-HIGH(G, ε) and TRIANGLE-EST-LOW(G, ε) to call. This guessing technique is very standard by now in property testing literature [19, 14, 15, 2]. Another point to note is that we do not know m. However, we can estimate m by using $poly(\log n)$ BIS queries (see Table 1). An estimate of m will perfectly work for us in this case.

Algorithm TRIANGLE-EST-HIGH(G, ε)

Algorithm TRIANGLE-EST-HIGH is inspired by the triangle estimation algorithm of Assadi et al. [2], where we have ADJACENCY, DEGREE, RANDOM NEIGHBOR and RANDOM EDGE queries. Please see Appendix A.2 for formal definitions of these queries. Note that the algorithm by Assadi et al. can be *suitably* modified even if we have approximate versions of DEGREE, RANDOM NEIGHBOR and RANDOM EDGE queries. Also refer Appendix A.2 for formal definitions of approximate version of the above queries. By Corollary A.8, $\widetilde{\mathcal{O}}(1)$ BIS queries are enough to simulate the approximate versions of DEGREE and RANDOM NEIGHBOR, with a probability of at least $1 - o(1)$. By Proposition A.7, approximate version of RANDOM EDGE queries can also be simulated by $\widetilde{\mathcal{O}}(1)$ BIS queries, with a probability of at least $1 - o(1)$. Putting everything together, we get TRIANGLE-EST-HIGH(G, ε) for triangle estimation that makes $\widetilde{\mathcal{O}}(1)$ BIS queries. The formal statement of the corresponding triangle estimation result is given in Lemma A.2, and algorithm TRIANGLE-EST-HIGH(G, ε) is described in Appendix A.2.

Algorithm TRIANGLE-EST-LOW(G, ε)

This algorithm is inspired by the two pass streaming algorithm for triangle estimation by McGregor et al. [20]. Basically, we show that the steps of McGregor et al.'s algorithm can be executed by using $\widetilde{\mathcal{O}}\left(\frac{m+T}{\sqrt{T}}\right)$ BIS queries. To do so, we have used the fact that, given any $X \subseteq V(G)$, all the edges of the subgraph induced by X can be enumerated by using $\widetilde{\mathcal{O}}\left(|E(G[X])|\right)$ BIS queries (see Proposition A.4 for the formal statement). The formal statement of the corresponding triangle estimation result is given in Lemma A.3, and algorithm TRIANGLE-EST-LOW(G, ε) is described in Appendix A.3 along with its correctness proof and query complexity analysis.

3 Lower bound for estimating triangles using EDGE EMPTINESS queries

In this Section, we prove the main lower bound result as sketched in Theorem 1.6; the formal theorem statement is stated below. As mentioned earlier, the lower bound proofs will be for the stronger query oracle EE. This will imply the lower bound for BIS.

▶ **Theorem 3.1** (Main lower bound result). *Let $m, n, t \in \mathbb{N}$ be such that $1 \leq t \leq \frac{m^{3/2}}{2}$. Any (randomized) algorithm that has* EE *oracle access to a graph $G(V, E)$ must make $\widetilde{\Omega}\left(\min\left\{\sqrt{t}, \frac{m^{3/2}}{t}\right\}\right)$* EE *queries to decide whether the number of triangles in G is at most t or at least $2t$ with a probability of at least $2/3$, where G has $n \geq 4\sqrt{m}$ vertices and $\Theta(m)$ edges.*

We prove the above theorem by proving Lemmas 3.2 and 3.3, as stated below. Note that Lemmas 3.2 and 3.3 talk about the desired lower bound when the number of triangles in the graph is *large* ($\Omega(m \log n)$) and *small* ($\mathcal{O}(m \log n)$), respectively.

▶ **Lemma 3.2** (Lower bound when there are *large* number of triangles)**.** *Let $m, n, t \in \mathbb{N}$ be such that $t \geq \frac{m \log n}{8}$. Any (randomized) algorithm that has* EE *oracle access to a graph $G(V,E)$ must make $\widetilde{\Omega}\left(\frac{m^{3/2}}{t}\right)$* EE *queries to decide whether the number of triangles in G is at most t or at least $2t$ with a probability of at least $2/3$, where G has $n \geq 4\sqrt{m}$ vertices, $\Theta(m)$ edges.*

▶ **Lemma 3.3** (Lower bound when there are *small* number of triangles)**.** *Let $m, n, t \in \mathbb{N}$ be such that $t < \frac{m \log n}{8}$. Any (randomized) algorithm that has* EE *oracle access to a graph $G(V,E)$ must make $\widetilde{\Omega}\left(\sqrt{t}\right)$* EE *queries to decide whether the number of triangles in G is at most t or at least $2t$ with a probability of at least $2/3$, where G has $n \geq 4\sqrt{m}$ vertices and $\Theta(m)$ edges.*

We first show Lemma 3.2 in Section 3.1, and then Lemma 3.3 in Section 3.2. Note that the proof of Lemma 3.3 will use Lemma 3.2.

3.1 Proof of Lemma 3.2

Without loss of generality, assume that $\sqrt{m}$ is an integer. We prove for the case when $n = 4\sqrt{m}$. But, we can make the proof work for any $n \geq 4\sqrt{m}$ by adding $n - 4\sqrt{m}$ isolated vertices. Note that $t \geq \frac{m \log n}{8}$ here. We further assume that $t \leq \frac{m^{3/2}}{128}$, and $m = \Omega(\log^2 n)$. Otherwise, the stated lower bound of $\widetilde{\Omega}\left(\frac{m^{3/2}}{t}\right)$ trivially follows as $\widetilde{\Omega}(\cdot)$ hides a multiplicative factor of $\frac{1}{poly(\log n)}$.

We use Yao's min-max principle to prove the lower bound. To do so, we consider two distributions $\mathcal{D}_{\mathbf{Yes}}$ and $\mathcal{D}_{\mathbf{No}}$ on graphs where

- Any graph $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$ has $4\sqrt{m}$ vertices;
- Any graph $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$ has $\Theta(m)$ edges with a probability of at least $1 - o(1)$;
- The number of triangles in any graph $G \sim \mathcal{D}_{\mathbf{Yes}}$ is at most t with a probability of at least $1 - o(1)$, and any graph $G \sim \mathcal{D}_{\mathbf{No}}$ has at least $2t$ triangles with a probability of at least $1 - o(1)$.

Note that, if we can show that any deterministic algorithm that distinguishes graphs from $\mathcal{D}_{\mathbf{Yes}}$ and $\mathcal{D}_{\mathbf{No}}$, with a probability of at least $2/3$, must make $\widetilde{\Omega}\left(\frac{m^{3/2}}{t}\right)$ EE queries, then we are done with the proof of Lemma 3.2.

3.1.1 The (hard) distribution for the input, its properties, and the proof set up

$\mathcal{D}_{\mathbf{Yes}}$: A graph $G \sim \mathcal{D}_{\mathbf{Yes}}$ is sampled as follows:

- Partition the vertex set $V(G)$ into 4 parts A, B, C, D, by initializing A, B, C, D as empty sets, and then putting each vertex in $V(G)$ into one of the parts uniformly at random and independent of other vertices;
- Connect each vertex of A with every vertex of B with an edge to form a biclique. Also, connect each vertex of C with every vertex of D with an edge to form another biclique;
- For every $\{x, y\}$ where $x \in A \cup B$ and $y \in C$, add edge $\{x, y\}$ to G with probability $\sqrt{\frac{t}{16m^{3/2}}}$.

$\mathcal{D}_{\mathbf{No}}$: A graph $G \sim \mathcal{D}_{\mathbf{No}}$ is sampled as follows:

- Partition the vertex set $V(G)$ into 4 parts A, B, C, D, by initializing A, B, C, D as empty sets, and then putting each vertex in $V(G)$ into one of the partitions uniformly at random and independent of other vertices;
- Connect each vertex of A with every vertex of B with an edge to form a biclique. Also, connect each vertex of C with every vertex of D with an edge to form another biclique;

- For every $\{x,y\}$ where $x \in A \cup B$ and $y \in C$, add edge $\{x,y\}$ to G with probability $\sqrt{\frac{t}{16m^{3/2}}}$.
- Select $C' \subseteq C$ by putting each $x \in C$ into C' with a probability of at least $\frac{32t}{m^{3/2}}$, independently, and then, add each edge in $\{x, y : x \in A \cup B, y \in C'\}$ to G.

The following observation establishes the number of vertices, edges, and the number of triangles in the graphs that can be sampled from $\mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$. The proof uses *large deviation inequalities* and is presented in the full version of the paper.

▶ **Observation 3.4** (Properties of the graph $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$).
(i) For $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$, the number of vertices in G is $4\sqrt{m}$. Also, $\frac{\sqrt{m}}{2} \leq |A|, |B|, |C|, |D| \leq 2\sqrt{m}$ holds with a probability of at least $1 - o(1)$, and the number of edges in G is $\Theta(m)$ with a probability of at least $1 - o(1)$;
(ii) If $G \sim \mathcal{D}_{\mathbf{Yes}}$, then there are at most t triangles in G with a probability of at least $1 - o(1)$,
(iii) If $G \sim \mathcal{D}_{\mathbf{No}}$, $\frac{8t}{m} \leq |C'| \leq \frac{64t}{m}$ with a probability of at least $1 - o(1)$, and there are at least $2t$ triangles in G with a probability of at least $1 - o(1)$.

The following remark is regarding the connection between graphs in $\mathcal{D}_{\mathbf{Yes}}$ and that in $\mathcal{D}_{\mathbf{No}}$. This will be used later in our proof, particularly in the proof of Claim 3.12.

▶ **Remark 1** (A graph $G' \sim \mathcal{D}_{\mathbf{No}}$ can be generated from a graph $G \sim \mathcal{D}_{\mathbf{Yes}}$). Let us first generate a graph $G \sim \mathcal{D}_{\mathbf{Yes}}$. Select $C' \subseteq C$ by putting each $x \in C$ into C' with a probability of at least $\frac{32t}{m^{3/2}}$, and then, add each edge in $\{x, y : x \in A \cup B, y \in C'\}$ to G to generate G', then (the resulting graph) $G' \sim \mathcal{D}_{\mathbf{No}}$.

The following observation says that a $\{x,y\} \in \binom{V(G)}{2}$ (with some condition) forms an edge with a probability of at least a constant. It will be used while we prove Claim 3.11.

▶ **Observation 3.5** (Any vertex pair $\{x,y\}$ is an edge in G with constant probability). Let $\{x,y\} \in \binom{V(G)}{2}$, and we are in the process of generating $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$. Let at most one of x and y has been put into one of the parts out of A, B, C and D. Then $\{x,y\}$ is an edge in G with probability at least $\frac{1}{4}$.

The above observation follows from the description of $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$ – each vertex in $V(G)$ is put into one of the parts out of A, B, C, D uniformly at random, each vertex of A is connected with every vertex in B, and each vertex of C is connected with every vertex in D.

In order to prove Lemma 3.2, by contradiction, assume that there is a randomized algorithm that makes $q = o\left(\frac{m^{3/2}}{t}\frac{1}{\log^2 n}\right)$ EE queries and decides whether the number of triangles in the input graph is at most t or at least $2t$, with a probability of at least $2/3$. Then there exists a deterministic algorithm ALG that makes q EE queries and decides the following (when the input graph $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$ be such that both $G \sim \mathcal{D}_{\mathbf{Yes}}$ and $G \sim \mathcal{D}_{\mathbf{No}}$ holds with probability $1/2$) –

$$\mathbb{P}_{G \sim \mathcal{D}_{\mathbf{No}}}(\text{ALG(G) reports NO}) - \mathbb{P}_{G \sim \mathcal{D}_{\mathbf{Yes}}}(\text{ALG(G) reports NO}) \geq \frac{1}{3} - o(1).$$

(Here $\mathbb{P}_{G \sim \mathcal{D}_{\mathbf{No}}}(\mathcal{E})$ and $\mathbb{P}_{G \sim \mathcal{D}_{\mathbf{Yes}}}(\mathcal{E})$ denote the probability of the event $\mathcal{E}$ under the conditional space $G \sim \mathcal{D}_{\mathbf{No}}$ and $G \sim \mathcal{D}_{\mathbf{Yes}}$, respectively.) Hence, we will be done with the proof of Lemma 3.2 by showing the following lemma.

▶ **Lemma 3.6** (Lower bound on the number of EE queries when $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$)**.** *Let the unknown graph G be such that $G \sim \mathcal{D}_{\mathbf{Yes}}$ and $G \sim \mathcal{D}_{\mathbf{No}}$ hold with equal probabilities. Consider any deterministic algorithm* ALG *that has* EE *access to G, and makes $q = o\left(\frac{m^{3/2}}{t}\frac{1}{\log^2 n}\right)$* EE *queries to G. Then*

$$\mathbb{P}_{G\sim\mathcal{D}_{\mathbf{No}}}(\text{ALG}(G) \textit{ reports } \text{NO}) - \mathbb{P}_{G\sim\mathcal{D}_{\mathbf{Yes}}}(\text{ALG}(G) \textit{ reports } \text{NO}) \leq o(1).$$

Next, we define an augmented EE oracle (EE* oracle). EE* is tailor-made for the graphs coming from $\mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$. Moreover, it is *stronger* than EE, that is, any EE query can be simulated by a EE* query. We will prove the claimed lower bound in Lemma 3.6 when we have access to EE* oracle. Note that this will imply Lemma 3.6.

Before getting into the formal description of EE* oracle, note that the algorithm (with EE* access) maintains a four tuple data structure initialized with $\emptyset$. With each query to EE* oracle, the oracle updates the data structure and returns the updated data structure to the algorithm. Note that the updated data structure is a function of all previously made EE* queries, and it is enough to answer corresponding EE queries.

3.1.2 Augmented Edge Emptiness oracle (EE*)

Before describing the EE* query oracle and its interplay with the algorithm, first we present the data structure $(E_Q, V(E_Q), e, \ell_v)$ that the algorithm maintains with the help of EE* oracle. The data structure keeps track of the following information.

Information maintained by $(E_Q, V(E_Q), e, \ell_v)$

- E_Q is a subset of $\binom{V(G)}{2}$ that have been seen by the algorithm till now, $V(E_Q)$ is the set of vertices present in any vertex pair in E_Q.
- $e : \binom{V(E_Q)}{2} \to \{0,1\}$ such that $e(\{x,y\}) = 1$ means the algorithm knows that $\{x,y\}$ is an edge in G, $e(x,y) = 0$ means that $\{x,y\}$ is not an edge in G.
- $\ell_v : V(E_Q) \to \{A, B, C, C', D\}$, where

$$\ell_v(x) = \begin{cases} A, & x \in A \\ B, & x \in B \\ C, & x \in C \setminus C' \\ C', & x \in C' \\ D, & x \in D \end{cases}$$

Intuitively speaking, unless the algorithm knows about the presence of some vertex in C', it cannot distinguish whether the unknown graph $G \sim \mathcal{D}_{\mathbf{Yes}}$ or $G \sim \mathcal{D}_{\mathbf{No}}$. So, we define the notion of good and bad vertices, along with good and bad data structures. This notion will be used later in our proof.

▶ **Definition 3.7** (Bad vertex)**.** A vertex $x \in V(E_Q)$ is said to be a *bad* vertex if $\ell_v(x) = C'$. $(E_Q, V(E_Q), e, \ell_v)$ is said to be *good* if there does not exist any bad vertex in $V(E_Q)$.

EE* oracle and its interplay with the algorithm

The algorithm initializes the data structure $(E_Q, V(E_Q), e, \ell_v)$ with $E_Q = \emptyset$, $V(E_Q) = \emptyset$. So, e and ℓ_v are initialized with trivial functions with domain $\emptyset$. At the beginning of each round, the algorithm queries the EE* oracle with a subset $P \subseteq \binom{V(G)}{2}$ deterministically. Note that the choice of P depends on the current status of the data structure. Now, we explain how EE* oracle responds to the query and how the data structure is updated.

(1) If $|P| \le \tau = 25 \log^2 n$, the oracle sets $E_Q \leftarrow E_Q \cup P$, and changes $V(E_Q)$ accordingly. The oracle also sets the function e and ℓ_v as per their definitions, and then it sends the updated data structure to the algorithm.

(2) Otherwise (if $|P| > \tau$), the oracle finds a random subset $P' \subseteq P$ such that $|P'| = \tau$. The oracle checks if there is a pair $\{u, v\} \in P'$ such that $\{u, v\}$ is an edge. If yes, then the oracle responds as in (1) with P being replaced by P'. If no, the oracle sends the data structure corresponding to the entire graph along with a FAILURE signal.[7]

Owing to the way EE* oracle updates the data structure after each EE* query, we can make some assumptions on the inputs to the EE* oracle, as described in Remark 2. It will actually be useful when we prove Claim 3.11.

▶ **Remark 2** (Some assumptions on the EE* query). Let $(E_Q, V(E_Q), e, \ell_v)$ be the data structure just before the algorithm makes EE query with input P, and let $(E'_Q, V(E'_Q), e', \ell'_v)$ be the data structure updated by EE* oracle after the algorithm makes EE* query with input P. Without loss of generality, we assume that

(i) P is disjoint from $\binom{V(E_Q)}{2}$. It is because EE* maintains whether $\{x, y\}$ is an edge or not for each $\{x, y\} \in \binom{V(E_Q)}{2}$;

(ii) When $x, z \in V(E_Q)$, there does not exist $\{x, y\}$ and $\{y, z\}$ in P. It is because the oracle updates the data structure in the same way in each of the following three cases when $x, z \in V(E_Q)$ – (i) $\{x, y\}$ and $\{y, z\}$ are in P, (ii) $\{x, y\} \in P$ and $\{y, z\} \notin P$, and (iii) $\{x, y\} \notin P$ and $\{y, z\} \in P$. By the description of EE* oracle and its interplay with the algorithm, in all the three cases, the updated data structure $(E'_Q, V(E'_Q), e', \ell'_v)$ contains labels of all the three vertices x, y, z along with the information whether $\{x, y\}$ and $\{y, z\}$ form edges in G or not. So, instead of having both $\{x, y\}$ and $\{y, z\}$ in P with $x, z \in V(E_Q)$, it is *equivalent* to have exactly one among $\{x, y\}$ and $\{y, z\}$ in P.

In the following observation, we formally show that EE* oracle is stronger than that of EE. Then we prove Lemma 3.9 that says that $\Omega\left(\frac{m^{3/2}}{t}\frac{1}{\log^2 n}\right)$ EE* queries are necessary to distinguish between $G \sim \mathcal{D}_{\mathbf{Yes}}$ and $G \sim \mathcal{D}_{\mathbf{No}}$. Note that Lemma 3.9 will imply Lemma 3.6.

▶ **Observation 3.8** (EE* is stronger than EE). Let $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$. Each EE query to G can be simulated by using an EE* query to G.

Proof. Let us consider an EE query with input $P \subseteq \binom{V(G)}{2}$. We make a EE* query with the same input P, and answer the EE query as follows depending on whether $|P| \le \tau$ or $|P| > \tau$.

$|P| \le \tau$: The EE* oracle updates the data structure and let $(E'_Q, V(E'_Q), e', \ell'_v)$ be the updated data structure. It contains the the information about each $\{x, y\} \in P$ whether it forms an edge in G or not. So, from $(E'_Q, V(E'_Q), e', \ell'_v)$, the EE query with input P can be answered as follows: there exists an edge $\{x, y\} \in P$ with $\{x, y\} \in E(G)$ if and only if $e'(\{x, y\}) = 1$.

$|P| > \tau$: In this case, the EE* oracle finds a random subset $P' \subseteq P$ such that $|P'| = \tau$. It checks if there is an $\{x, y\} \in P'$ such that $\{x, y\}$ is an edge. If yes, the updated data structure contains the the information about each $\{x, y\} \in P'$ whether it forms an edge. In this case, we can report that there exists an $\{x, y\} \in P$ such that $\{x, y\}$ is an edge in G. If there is no $\{x, y\} \in P'$ such that $\{x, y\}$ is an edge, then (by the description of

[7] We later argue that FAILURE signal is sent with a very low probability.

EE oracle and its interplay with the algorithm) the EE* oracle sends the data structure corresponding to the entire graph. Obviously, we can report whether there exists an $\{x, y\} \in P$ such that $\{x, y\} \in E(G)$ or not.

Hence, in any case, we can report the answer to EE query with input P. ◀

We are left with proving the following technical lemma. As noted earlier, this will imply Lemma 3.6.

▶ **Lemma 3.9** (Lower bound on the number of EE* queries when $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$). *Let the unknown graph G be such that $G \sim \mathcal{D}_{\mathbf{Yes}}$ and $G \sim \mathcal{D}_{\mathbf{No}}$ hold with equal probabilities. Consider any deterministic algorithm* ALG* *that has* EE* *access to G, and makes $q = o\left(\frac{m^{3/2}}{t}\frac{1}{\log^2 n}\right)$* EE* *queries to G. Then*

$$\mathbb{P}_{G\sim\mathcal{D}_{\mathbf{No}}}(\text{ALG}^*\ (G)\ \textit{reports}\ \text{NO}) - \mathbb{P}_{G\sim\mathcal{D}_{\mathbf{Yes}}}(\text{ALG}^*\ (G)\ \textit{reports}\ \text{NO}) \leq o(1).$$

3.1.3 Proof of Lemma 3.9

For clarity of explanation, we first describe ALG* as a decision tree. Then we will prove Lemma 3.9.

Decision tree view of ALG*

- Each internal node of $\mathcal{T}$ is labeled with a nonempty subset $\binom{V(G)}{2}$ and each leaf node is labeled with YES or NO;
- Each edge in the tree is labeled with a data structure $(E_Q, V(E_Q), e, \ell_v)$;
- The algorithm starts the execution from the root node r by setting r as the current node. Note that for the root node r, $E_Q = V(E_Q) = \emptyset$ and e and ℓ_v are the trivial functions. As the algorithm ALG* is deterministic, the first EE* query is same irrespective of the graph $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$ that we are querying. By making that query, we get an updated data structure from the oracle and let $\{r, u\}$ be the edge that is labeled with the updated data structure. Then ALG* sets u as the current node.
- If the current node u is not a leaf node in $\mathcal{T}$, ALG* makes a EE* query with a subset $P \subseteq \binom{V(G)}{2}$, where P is determined by the label of the node u. Note that P satisfies the condition described in Remark 2. The oracle updates the knowledge structure and ALG* moves to a child of u depending on the updated data structure;
- If the current node u is a leaf node in $\mathcal{T}$, report YES or NO according to the label of u.

Now, we define the notion of *good* and *bad* nodes in $\mathcal{T}$. The following definition is inspired from Definition 3.7.

▶ **Definition 3.10** (Bad node in the decision tree). Let u be a node of $\mathcal{T}$ and $(E_Q, V(E_Q), e, \ell_v)$ be the current data structure. u is said to be *good* if there does not exist x in $V(E_Q)$ such that $\ell_v(x) = C'$. Otherwise, u is a *bad* node.

If $G \sim \mathcal{D}_{\mathbf{Yes}}$, then ALG* will never encounter a bad node. In other words, when ALG* reaches a bad node of the tree $\mathcal{T}$, then it can (easily) decide $G \sim \mathcal{D}_{\mathbf{No}}$. However, the inverse in not true. From this fact, consider two claims (Claims 3.11 and 3.12) about the traversal of the decision tree $\mathcal{T}$ when the graph $G \sim \mathcal{D}_{\mathbf{Yes}} \cup \mathcal{D}_{\mathbf{No}}$. These claims will be useful to show Lemma 3.9. Intuitively, Claim 3.11 says that the probability of reaching a bad node is very low when $G \sim \mathcal{D}_{\mathbf{No}}$. Claim 3.12 says that the probability to reach any particular good node is more when $G \sim \mathcal{D}_{\mathbf{Yes}}$ as compared to that of when $G \sim \mathcal{D}_{\mathbf{No}}$.

▶ **Claim 3.11** (Probability of reaching a bad node is very low when $G \sim \mathcal{D}_{\mathbf{No}}$)**.** *Let $G \sim \mathcal{D}_{\mathbf{No}}$. Then the probability that* ALG* *reaches a bad node of the decision tree is $o(1)$. That is,*

$$\mathbb{P}_{G\sim\mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \textit{ reaches a bad node}) = o(1).$$

▶ **Claim 3.12** (A technical claim to prove Lemma 3.9)**.** *For any good node in the decision tree $\mathcal{T}$, the following holds:*

$$\mathbb{P}_{G\sim\mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \textit{ reaches } v) \le \mathbb{P}_{G\sim\mathcal{D}_{\mathbf{Yes}}}(\text{ALG}^* \textit{ reaches } v).$$

The proofs of Claims 3.11 and 3.12 are non trivial and are in the full version due to paucity of space.

Now, we will prove Lemma 3.9.

Proof of Lemma 3.9. Let $\mathcal{L}_{\mathbf{No}}$ denote the set of leaf nodes of the decision tree $\mathcal{T}$ that are labeled NO. Also, let $\mathcal{L}_g \subseteq \mathcal{L}_{\mathbf{No}}$ be the set of leaf nodes that are good and labeled as NO.

$$\begin{aligned}
&\mathbb{P}_{G\sim\mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \text{ (G) reports NO})\\
&\le \sum_{v\in\mathcal{L}_{\mathbf{No}}} \mathbb{P}_{G\sim\mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \ (G) \text{ reaches } u)\\
&= \sum_{u\in\mathcal{L}_g} \mathbb{P}_{G\sim\mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \ (G) \text{ reaches } u) + \sum_{u\in\mathcal{L}_{\mathbf{No}}\setminus\mathcal{L}_g} \mathbb{P}_{G\sim\mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \ (G) \text{ reaches } u)\\
&\le \sum_{u\in\mathcal{L}_g} \mathbb{P}_{G\sim\mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \ (G) \text{ reaches } u) + \mathbb{P}_{G\sim\mathcal{D}_{\mathbf{No}}}(\text{ALG}^* \ (G) \text{ reaches a bad node})\\
&\le \sum_{u\in\mathcal{L}_g} \mathbb{P}_{G\sim\mathcal{D}_{\mathbf{Yes}}}(\text{ALG}^* \ (G) \text{ reaches } u) + o(1) \qquad \text{(By Claims 3.11 and Claims 3.12)}\\
&\le \mathbb{P}_{G\sim\mathcal{D}_{\mathbf{Yes}}}(\text{ALG}^* \ (G) \text{ reports NO}) + o(1)
\end{aligned}$$

◀

3.2 Proof of Lemma 3.3

We assume that $t = \omega(\log^7 n)$. Otherwise, as $\widetilde{\Omega}(\cdot)$ hides a multiplicative term of $\frac{1}{poly(\log n)}$, the stated lower bound is trivial. Assume, for a contradiction, that there is an algorithm $\mathcal{A}$ for $t < \frac{m \log n}{8}$ such that

- it has EE oracle access to a graph $G_1(V_1, E_1)$ with $\Theta(\sqrt{m})$ vertices and $\Theta(m)$ edges;
- makes $o\left(\sqrt{t}\frac{1}{\log^{3.5} n}\right)$ EE queries;
- decides whether the number of triangles in G_1 is at most t or at least $2t$ with a probability of at least $2/3$.

Now we give an algorithm $\mathcal{A}'$ for

- it has EE oracle access to a graph $G_2(V_2, E_2)$ with $4\sqrt{t/\log n}$ vertices and $8t/\log n$ edges, that is, $t = \frac{|E_2| \log n}{8}$;
- makes $o\left(\frac{\sqrt{t}}{\log^{3.5} n}\right) = o\left(\frac{|E_2|^{3/2}}{t}\frac{1}{\log^2 n}\right)$ EE queries;
- decides whether the number of triangles in G_2 is at most t or at least $2t$ with a probability of at least $2/3$.

Description of $\mathcal{A}'$ using $\mathcal{A}$

- Let the unknown graph be $G_1 = G_2 \cup G'$ such that $V(G_1) = V(G_2) \sqcup V(G')$ and $E(G_1) = E(G_2) \sqcup E(G')$, where G' is a graph having $\Theta(\sqrt{m - t/\log n})$ vertices (disjoint from $V(G_2)$), $\Theta\left(\sqrt{(m - 8t/\log n)}\right)$ edges, and no triangles.[8] The number of vertices and edges in G_1 are $\Theta(\sqrt{m})$ and $\Theta(m)$, respectively. Also, the number of triangles in G_1 is same as in G_2;
- As an EE query to G_1 can be answered using one EE query to G_2, we can consider having EE query access to graph G_1;
- We run algorithm $\mathcal{A}$ assuming G_1 as the unknown graph;
- We report the output of $\mathcal{A}$ as the output of $\mathcal{A}$.

The correctness of $\mathcal{A}'$ follows from the correctness of $\mathcal{A}$. The number of queries made by the algorithm $\mathcal{A}'$ is $o\left(\sqrt{t}\frac{1}{\log^{3.5} n}\right)$. Recalling that $\mathcal{A}$ works on graph $G_2(V_2, E_2)$ satisfying $t = \frac{|E_2| \log n}{8}$ and by Lemma 3.2, algorithm $\mathcal{A}'$ does not exist as such an algorithm requires at least $\Omega\left(\frac{|E_2|^{3/2}}{t}\frac{1}{\log^2 n}\right)$ EE queries, which is $\Omega\left(\frac{\sqrt{t}}{\log^{3.5} n}\right)$.

Hence, we are done with the proof of Lemma 3.3.

4 Conclusion

We touched upon two open questions of Beame et al. [5] in this paper. We resolved the query complexity of triangle estimation when we have a Bipartite Independent Set oracle access to the unknown graph when $T = \Omega(m)$. But the query complexity of triangle counting remain illusive when $T = o(m)$ though we believe that our upper bound of $\widetilde{\mathcal{O}}(m/\sqrt{T})$ BIS queries is tight in this regard. It is also interesting if our upper bound can be improved when $T = o(m)$.

References

1 Sepehr Assadi, Deeparnab Chakrabarty, and Sanjeev Khanna. Graph Connectivity and Single Element Recovery via Linear and OR Queries. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume bisbisbisbis204 of *LIPIcs*, pages 7:1–7:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.ESA.2021.7.

2 Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A Simple Sublinear-Time Algorithm for Counting Arbitrary Subgraphs via Edge Sampling. In *Proceedings of the 10th Innovations in Theoretical Computer Science Conference, ITCS*, volume 124, pages 6:1–6:20, 2019.

3 Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Reductions in Streaming Algorithms, with an Application to Counting Triangles in Graphs. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 623–632, 2002.

4 Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge Estimation with Independent Set Oracles. In *Proceedings of the 9th Innovations in Theoretical Computer Science Conference, ITCS*, volume 94, pages 38:1–38:21, 2018.

5 Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge Estimation with Independent Set Oracles. *ACM Trans. Algorithms*, 16(4):52:1–52:27, 2020.

[8] The constants in $\Theta(\sqrt{m - 8t/\log n})$ and $\Theta(m - 8t/\log n)$ are chosen suitably such that the graph G_1 satisfies the requirement of $\mathcal{A}$.

6 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Hyperedge Estimation using Polylogarithmic Subset Queries. *CoRR*, abs/1908.04196, 2019.

7 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Triangle Estimation Using Tripartite Independent Set Queries. In *Proceedings of the 30th International Symposium on Algorithms and Computation, ISAAC*, volume 149, pages 19:1–19:17, 2019.

8 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. On Triangle Estimation Using Tripartite Independent Set Queries. *Theory Comput. Syst.*, 65(8):1165–1192, 2021.

9 Arijit Bishnu, Arijit Ghosh, Sudeshna Kolay, Gopinath Mishra, and Saket Saurabh. Parameterized Query Complexity of Hitting Set Using Stability of Sunflowers. In *Proceedings of the 29th International Symposium on Algorithms and Computation, ISAAC*, volume 123, pages 25:1–25:12, 2018.

10 Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. On the Complexity of Triangle Counting using Emptiness Queries. *CoRR*, abs/2110.03836, 2021. `arXiv:2110.03836`.

11 Xi Chen, Amit Levi, and Erik Waingarten. Nearly Optimal Edge Estimation with Independent Set Queries. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2916–2935, 2020.

12 Holger Dell and John Lapinskas. Fine-Grained Reductions from Approximate Counting to Decision. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 281–288, 2018.

13 Holger Dell, John Lapinskas, and Kitty Meeks. Approximately Counting and Sampling Small Witnesses Using a Colourful Decision Oracle. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2201–2211, 2020.

14 Talya Eden, Amit Levi, Dana Ron, and C. Seshadhri. Approximately Counting Triangles in Sublinear Time. *SIAM J. Comput.*, 46(5):1603–1646, 2017.

15 Talya Eden, Dana Ron, and C. Seshadhri. On Approximating the Number of k-Cliques in Sublinear Time. *SIAM J. Comput.*, 49(4):747–771, 2020.

16 Uriel Feige. On Sums of Independent Random Variables with Unbounded Variance and Estimating the Average Degree in a Graph. *SIAM J. Comput.*, 35(4):964–984, 2006.

17 Oded Goldreich. *Introduction to Property Testing.* Cambridge University Press, 2017.

18 Oded Goldreich and Dana Ron. Property Testing in Bounded Degree Graphs. *Algorithmica*, 32(2):302–343, 2002.

19 Oded Goldreich and Dana Ron. Approximating Average Parameters of Graphs. *Random Struct. Algorithms*, 32(4):473–493, 2008.

20 Andrew McGregor, Sofya Vorotnikova, and Hoa T. Vu. Better Algorithms for Counting Triangles in Data Streams. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS*, pages 401–411, 2016.

21 Rajeev Motwani and Prabhakar Raghavan. Randomized Algorithms. In *Algorithms and Theory of Computation Handbook*, Chapman & Hall/CRC Applied Algorithms and Data Structures series. CRC Press, 1999.

22 Cyrus Rashtchian, David P. Woodruff, and Hanlin Zhu. Vector-Matrix-Vector Queries for Solving Linear Algebra, Statistics, and Graph Problems. In *Proceedings of the 24th International Conference on Randomization and Computation, RANDOM*, volume 176, pages 26:1–26:20, 2020.

23 Dana Ron and Gilad Tsur. The Power of an Example: Hidden Set Size Approximation Using Group Queries and Conditional Sampling. *ACM Trans. Comput. Theory*, 8(4):15:1–15:19, 2016.

24 Larry J. Stockmeyer. The Complexity of Approximate Counting (Preliminary Version). In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, STOC*, pages 118–126, 1983.

25 Larry J. Stockmeyer. On Approximation Algorithms for #P. *SIAM J. Comput.*, 14(4):849–861, 1985.

A Upper bound for estimating triangles using BIS

In this Section, we prove Theorem 1.7, which is formally stated as follows:

▶ **Theorem A.1** (Upper bound matching the lower bound in Theorem 3.1)**.** *There exists an algorithm* TRIANGLE-EST(G, ε) *that has* BIS *query access to a graph* $G(V, E)$ *having* n *vertices, takes a parameter* $\varepsilon \in (0, 1)$ *as input, and reports a* $(1 \pm \varepsilon)$*-approximation to the number of triangles in* G *with a probability of at least* $1 - o(1)$*. Moreover, the expected number of* BIS *queries made by the algorithm is* $\widetilde{\mathcal{O}}\left(\min\{\frac{m}{\sqrt{T}}, \frac{m^{3/2}}{T}\}\right)$*, where* m *and* T *denote the number of edges and triangles in* G*, respectively.*

In order to prove the above theorem, we first prove Lemmas A.2 and A.3.

▶ **Lemma A.2** (Upper bound when the number of triangles is *large*)**.** *There exists an algorithm* TRIANGLE-EST-HIGH(G, ε) *that has* BIS *query access to a graph* $G(V, E)$ *having* n *vertices,* $\varepsilon \in (0, 1)$ *as input, and reports a* $(1 \pm \varepsilon)$*-approximation to the number of triangles in* G *with a probability of at least* $1 - o(1)$*. Moreover, the expected number of* BIS *queries made by the algorithm is* $\widetilde{\mathcal{O}}\left(\frac{m^{3/2}}{T}\right)$*, where* m *and* T *denote the number of edges and triangles in* G*, respectively.*

▶ **Lemma A.3** (Upper bound when the number of triangles is *small*)**.** *There exists an algorithm* TRIANGLE-EST-LOW(G, ε) *that has* BIS *query access to a graph* $G(V, E)$ *having* n *vertices,* $\varepsilon \in (0, 1)$ *as input, and reports a* $(1 \pm \varepsilon)$*-approximation to the number of triangles in* G *with a probability of at least* $1 - o(1)$*. Moreover, the expected number of* BIS *queries made by the algorithm is* $\widetilde{\mathcal{O}}\left(\frac{m+T}{\sqrt{T}}\right)$*, where* m *and* T *denote the number of edges and triangles in* G*, respectively.*

Our final algorithm TRIANGLE-EST(G, ε) (as stated in Theorem A.1) is a combination of TRIANGLE-EST-HIGH(G, ε) and TRIANGLE-EST-LOW(G, ε). Informally, TRIANGLE-EST (G, ε) calls TRIANGLE-EST-HIGH(G, ε) and TRIANGLE-EST-LOW (G, ε) when $T = \Omega(m)$ and $T = \mathcal{O}(m)$, respectively. Observe that, if TRIANGLE-EST knows T within a constant factor, then it can decide which one to use among TRIANGLE-EST-HIGH(G, ε) and TRIANGLE-EST-LOW(G, ε). If TRIANGLE-EST does not know T within a constant factor, then it starts from a guess $L = \binom{n}{3}/2$ and updates L by making a *geometric* search until the output of TRIANGLE-EST is consistent with L. Depending on whether $L = \Omega(m)$ or $L = \mathcal{O}(m)$, TRIANGLE-EST decides which one among TRIANGLE-EST-HIGH(G, ε) and TRIANGLE-EST-LOW(G, ε) to call. This guessing technique is standard in the property testing literature. It has been used several times in the literature (for example in [19, 14], generalized in [15], and used directly in [2]). So, we explain TRIANGLE-EST-HIGH(G, ε) and TRIANGLE-EST-LOW(G, ε) assuming a promised lower bound on L, and the respective query complexities will be in terms of L instead of T.

Another important thing to observe is that to execute the above discussed steps of algorithm TRIANGLE-EST(G, ε) must know m. But we note that an estimate of m will be good enough for our purpose, and that can be estimated by using $\widetilde{\mathcal{O}}(1)$ BIS queries (see Table 1).

In Appendix A.1, we discuss some properties of BIS and some tasks it can perform. These properties will be useful while describing our TRIANGLE-EST-HIGH(G, ε) and TRIANGLE-EST-LOW(G, ε), and proving Lemma A.2 and Lemma A.3, in Section A.2 and Section A.3, respectively.

A.1 Some preliminaries about BIS

Let $G(V, E)$ be the unknown graph to which we have BIS query access. One can compute the exact number of edges using $\widetilde{\mathcal{O}}(|E(G)|)$ queries [5] deterministically. Also, we can estimate the number of edges in graph G [5, 6, 13] and sample an edge from G *almost uniformly* [13], with a probability of at least $1 - o(1)$, and making $\widetilde{\mathcal{O}}(1)$ BIS queries. Here, we would like to note that, all three results we mentioned above hold for induced subgraphs as well as induced bipartite subgraph, as formally described below. Those will be used when we design our upper bounds in Appendix A.2 and A.3.

▶ **Proposition A.4** (Exact edge estimation using BIS [5]). *There exists a deterministic algorithm that has* BIS *query access to an unknown graph $G(V, E)$ with n vertices, takes $X \subseteq V(G)$ (alternatively, two disjoint subsets A, B) as input, makes $\widetilde{\mathcal{O}}(|E(G[X])|)$ (alternatively, $\widetilde{\mathcal{O}}(|E(G[A, B])|)$)* BIS *queries, and reports all the edges in $E(G)$ (alternatively, $E(G[A, B])$).*

▶ **Proposition A.5** (Approximate edge estimation using BIS [6, 13]). *There exists an algorithm that has* BIS *query access to an unknown graph $G(V, E)$ with n vertices, takes $X \subseteq V(G)$ (alternatively, two disjoint subsets A, B) and a parameter $\varepsilon \in (0, 1)$ as inputs, makes $\widetilde{\mathcal{O}}(1)$* BIS *queries, and reports a $(1 \pm \varepsilon)$-approximation to $|E(G[X])|$ (alternatively, $|E(G[A, B])|$), with a probability of at least $1 - o(1)$.*

To state the next proposition, we need the following definition.

▶ **Definition A.6** (Approximate uniform sample from a set). For a nonempty set X and $\varepsilon \in (0, 1)$, getting a $(1 \pm \varepsilon)$-approximate uniform sample from X means getting a sample from a distribution on X such that the probability of getting $x \in X$ lies in $[(1-\varepsilon)/|X|, (1+\varepsilon)/|X|]$.

▶ **Proposition A.7** (Approximate edge sampling using BIS [13]). *There exists an algorithm that has* BIS *query access to an unknown graph $G(V, E)$ with n vertices, takes $X \subseteq V(G)$ (alternatively, two disjoint subsets A, B) and a parameter $\varepsilon \in (0, 1)$ as inputs, makes $\widetilde{\mathcal{O}}(1)$* BIS *queries, and reports a $(1 \pm \varepsilon)$-approximate uniform sample from $E(G[X])$ (alternatively, $E(G[A, B])$, with a probability of at least $1 - o(1)$.*

Observe that the following corollary follows from Propositions A.4, A.5 and A.7, by taking $A = \{v\}$ and $B = Z$, where $v \in V(G)$ and $Z \subseteq V(G) \setminus \{v\}$.

▶ **Corollary A.8** (BIS query can extract useful information about the neighborhood of a given vertex).

(i) Entire neighbourhood of a vertex using BIS: *There exists a deterministic algorithm that has* BIS *query access to an unknown graph $G(V, E)$ with n vertices, takes $v \in V(G)$ and $Z \subseteq V(G) \setminus \{v\}$ as input, makes $\widetilde{\mathcal{O}}(|N_G(v) \cap Z|)$* BIS *queries, and reports all the neighbors of v in Z.*

(ii) Approximate degree using BIS: *There exists an algorithm that has* BIS *query access to an unknown graph $G(V, E)$ with n vertices, takes $v \in V(G)$, $Z \subseteq V(G) \setminus \{v\}$ and $\varepsilon \in (0, 1)$ as inputs, makes $\widetilde{\mathcal{O}}(1)$* BIS *queries, and reports a $(1 \pm \varepsilon)$-approximation to $|N_G(v) \cap Z|$, with a probability of at least $1 - o(1)$.*

(iii) Finding an approximate random neighbor of a vertex using BIS: *There exists an algorithm that has* BIS *query access to an unknown graph $G(V, E)$ with n vertices, takes $v \in V(G)$, $Z \subseteq V(G) \setminus \{v\}$ and $\varepsilon \in (0, 1)$ as inputs, makes $\widetilde{\mathcal{O}}(1)$* BIS *queries, and reports a $(1 \pm \varepsilon)$-approximate uniform sample from the set $N_G(v) \cap Z$, with a probability of at least $1 - o(1)$.*

A.2 Algorithm Triangle-Est-High and proof of Lemma A.2

Algorithm Triangle-Est-High is inspired by the triangle estimation algorithm of Assadi et al. [2] [9] when we have the following query access to the unknown graph.

Adjacency Query: Given vertices $u, v \in V(G)$ as input, the oracle reports whether (u, v) is an edge or not;

Degree Query: Given a vertex $u \in V(G)$ as input, the oracle reports the degree of vertex u in G;

Random Neighbor Query: Given a vertex $u \in V(G)$, the oracle reports a neighbor of u uniformly at random if the degree of u is nonzero. Otherwise, the oracle reports a special symbol $\perp$;

Random Edge Query: With this query, the oracle reports an edge from the graph G uniformly at random.

The number of queries to the oracle made by Assadi et al.'s algorithm [2] is $\widetilde{\mathcal{O}}\left(\frac{m^{3/2}}{L}\right)$, where m denotes the number of edges and L is a promised lower bound on the number of triangles in G. Also, note that, the triangle estimation algorithm by Assadi et al. [2] can be *suitably* modified even if we have approximate versions of Degree, Random Neighbor and Random Edge queries, as described below.

Apx Degree Query: Given a vertex $u \in V(G)$ and $\varepsilon \in (0, 1)$ as input, the oracle reports a $(1 \pm \varepsilon)$-approximation to the degree of vertex u in G;

Apx Random Neighbor Query: Given a vertex $u \in V(G)$ and $\varepsilon \in (0, 1)$ as input, the oracle reports a $(1 \pm \varepsilon)$-approximate uniform sample from $N_G(u)$ if the degree of u is nonzero. Otherwise, the oracle reports a special symbol $\perp$;

Apx Random Edge Query: Given $\varepsilon \in (0, 1)$, the oracle reports a $(1 \pm \varepsilon)$-approximate uniform sample from $E(G)$.

From Corollary A.8, $\widetilde{\mathcal{O}}(1)$ BIS queries are enough to simulate Apx Degree Query and Apx Random Neighbor Query, with a probability of at least $1 - o(1)$. Also, by Proposition A.7, Apx Random Edge Query can be simulated by $\widetilde{\mathcal{O}}(1)$ BIS queries, with a probability of at least $1 - o(1)$. Moreover, a BIS query can trivially simulate an Adjacency Query. Combining these facts with the fact that the triangle estimation algorithm by Assadi et al. [2] can be suitably modified even if we have approximate versions of Degree, Random Neighbor and Random Edge queries, we are done with the proof of Lemma A.2.

A.3 Algorithm Triangle-Est-Low and the proof of Lemma A.3

Algorithm Triangle-Est-Low is inspired by the streaming algorithm for triangle counting by McGregor et al. [20]. Algorithm Triangle-Est-Low extracts a subset of edges by making BIS queries in a specific way as explained below. Later, we discuss that those sets of edges will be enough to estimate the number of triangles in G.

[9] Actually, they have given an algorithm for estimating the number of copies of any given subgraph of fixed size.

Generating a random sample $S \subseteq V(G)$ and exploring its neighborhood

Algorithm Triangle-Est-Low adds each vertex in $V(G)$ to S with probability $\widetilde{\mathcal{O}}\left(\frac{1}{\sqrt{L}}\right)$. Recall Corollary A.8 (i). For each $v \in S$, we find all the neighbors of v (the set $N_G(v)$) by making $\widetilde{\mathcal{O}}(|N_G(v)|)$ BIS queries. Let E_S be the set of all the edges having at least one end point in S, that is, $E_S = \{(v,w) : v \in S \text{ and } w \in N_G(v)\}$. After finding E_S, we do the following. For each $v \in S$, we find all the edges in the subgraph induced by $N_G(v)$ by using $\widetilde{\mathcal{O}}\left(|E(G[N_G(v)])|\right)$ BIS queries. This is again possible by Corollary A.8 (i). Note that $|E(G[N_G(v)])|$ is the number of edges in the subgraph of G induced by $N_G(v)$. Let E'_S be the set of all edges present in the subgraph induced by $N_G(v)$ for some $v \in S$, that is, $E'_S = \bigcup\limits_{v \in S} E(G[N_G(v)])$. Later, we argue that the number of BIS queries that we make to generate E_S and E'_S is bounded in expectation.

Apart from S, E_S and E'_S, Triangle-Est-Low extracts some more required edges by making BIS queries, as explained below.

Generating F, a set of $(1 \pm \mathcal{O}(\varepsilon))$-approximate uniform sample from $E(G)$, and exploring the subgraphs induced by sets $N_G(v) \cap V(F)$ for each $v \in V(F)$

Algorithm Triangle-Est-Low calls the algorithm corresponding to Proposition A.7, for $\widetilde{\mathcal{O}}\left(\frac{m}{\sqrt{L}}\right)$ times. By this process, we get a set F of $(1 \pm \mathcal{O}(\varepsilon))$-approximate uniform sample from $E(G)$, with a probability of at least $1 - o(1)$. Note that $|F| = \widetilde{\mathcal{O}}\left(\frac{m}{\sqrt{L}}\right)$, and the number of BIS queries we make to generate F is $\widetilde{\mathcal{O}}\left(\frac{m}{\sqrt{L}}\right)$. Let $V(F)$ be the set of vertices present in at least one edge in F. For each vertex $v \in V(F)$, we find all the edges in the subgraph of G induced by $N_G(v) \cap V(F)$, by using $\widetilde{\mathcal{O}}\left(|E(G[N_G(v) \cap V(F)])|\right)$ BIS queries (see Corollary A.8 (i)). Note that $|E\left(G[N_G(v) \cap V(F)]\right)|$ is the number of edges in the subgraph of G induced by $N_G(v) \cap F$. Let E_F be the set of all the edges that are present in subgraphs induced by $N_G(v) \cap V(F)$ for some $v \in V(F)$, that is, $E_F = \bigcup\limits_{v \in V(F)} E(G[N_G(v) \cap V(F)])$. Later we show that the expected number of BIS queries needed to find F and E_F is bounded.

In algorithm Triangle-Est-Low, BIS queries are made only to generate S, E_S, E'_S, F and E_F. After these sets are generated, no more BIS queries are made by the algorithm. We formally prove the query complexity of Triangle-Est-Low. But, first, we show that S, E_S, E'_S, F and E_F can be carefully used to estimate T, the number of triangles in G.

Connection with streaming algorithm for triangle counting by McGregor et al. [20]

(Estimating the number of triangles from S, E_S, E'_S, F and E_F)

McGregor et al. [20] gave a two-pass algorithm that estimates the number of triangles in a graph G when the edges of G arrive in an arbitrary order. Moreover, the space complexity of their algorithm is $\widetilde{\mathcal{O}}\left(\frac{m}{\sqrt{L}}\right)$. Note that their algorithm assumes a lower bound L on the number of triangles in the graph. The high level sketch of their algorithm is as follows:

- Generate a subset X of $V(G)$ by sampling each vertex in $V(G)$ with probability $\widetilde{\mathcal{O}}\left(\frac{1}{\sqrt{L}}\right)$;
- In the first pass, the edges having at least one vertex in X is found, and let it be E_X. Also, in the first pass, a subset of edges Y is generated by sampling each edge with probability $\widetilde{\mathcal{O}}\left(\frac{1}{\sqrt{L}}\right)$;
- In the second pass, for each edge $e = \{x, y\}$ in the stream, their algorithm finds the vertices in X with which e forms a triangle. Also, for each edge, $e = \{x, y\}$ in the stream, their algorithm finds the pairs of edges in Y that forms a triangle with e. Let Z be the set of *useful* edges in the second pass, that is, the set of edges that either forms a triangle with a vertex in X or forms a triangle with two edges in Y.

Note that their algorithm does not talk about the set Z. We are introducing it for our analysis. Executing the two passes described above is straightforward. They have proved that performing two passes as described is good enough to estimate the number of triangles in the graph.

Now, we compare the information maintained by our algorithm with the information maintained by McGregor et al.'s algorithm. S and E_S in our algorithm TRIANGLE-EST-LOW follows the same probability distribution as that of X and E_X, respectively, in McGregor et al.'s algorithm. Recall that F is a set of $\widetilde{\mathcal{O}}\left(\frac{m}{\sqrt{L}}\right)$ $(1 \pm \varepsilon)$-approximate sample from $E(G)$ with a probability $1 - o(1)$. But Y in McGregor et al.'s algorithm is generated by sampling each edge with probability $\widetilde{\mathcal{O}}\left(\frac{1}{\sqrt{L}}\right)$. But observe that the *total variation* distance between the probability distributions of F and Y is $o(1)$.

Before discussing about E'_S and E_F in our algorithm TRIANGLE-EST-LOW, consider the following observation about the set Z. Note that we have defined Z while describing the second pass of McGregor et al.'s algorithm.

▶ **Observation A.9.** Consider X, E_X, and Y generated by the first pass of McGregor et al.'s algorithm. Let E'_X be the edges in the subgraph induced by $N_G(v)$ for some $v \in X$, and let E_Y be the set of edges in the subgraph induced by $N_G(v) \cap V(Y)$. Here $V(Y)$ denotes the set of vertices present in at least one vertex of Y. Then for each edge $e \notin E'_X \cup E_Y$, there is neither a vertex in X with which e forms a triangle in G nor there are two edges in Y with which e forms a triangle in G. Then the set of useful edges Z is $E'_X \cup E_Y$.

By the above observation, E'_S and E_F in our algorithm TRIANGLE-EST-LOW are essentially enough for maintaining the information and executing the same steps as that of the second pass of McGregor et al.'s algorithm.

Putting everything together, algorithm TRIANGLE-EST-LOW outputs a $(1 \pm \varepsilon)$-approximation to the number of triangles in the graph.

Query complexity analysis

The set S can be generated without making any BIS queries. The number of BIS queries we make to find the set E_S is at most $\sum\limits_{v \in S} \widetilde{\mathcal{O}}(|N_G(v)|)$, which in expectation is

$$\mathbb{E}\left[\sum_{v \in S} \widetilde{\mathcal{O}}(|N_G(v)|)\right] = \sum_{v \in V(G)} \Pr(v \in S) \cdot \widetilde{\mathcal{O}}(|N_G(v)|) = \widetilde{\mathcal{O}}\left(\frac{m}{\sqrt{L}}\right).$$

The number of BIS queries we make to find the set E'_S is at most $\sum\limits_{v \in S} \widetilde{\mathcal{O}}\left(|E(G[N_G(v)])|\right)$. Note that $|E(G[N_G(v)]|$ is T_v, that is, the number of triangles having v as one of the vertex. So, the expected number of BIS queries we make to find E'_S is at most

$$\mathbb{E}\left[\sum_{v \in S} \widetilde{\mathcal{O}}(T_v)\right] = \sum_{v \in V(G)} \Pr(v \in S) \cdot \widetilde{\mathcal{O}}(T_v) = \widetilde{\mathcal{O}}\left(\frac{T}{\sqrt{L}}\right).$$

The number of BIS queries we make to find the set F is $\widetilde{\mathcal{O}}(|F|) = \widetilde{\mathcal{O}}\left(\frac{m}{\sqrt{L}}\right)$

The number of BIS queries to generate E_F is at most $\sum\limits_{v \in V(F)} \widetilde{\mathcal{O}}\left(|E(G[N_G(v) \cap V(F)])|\right)$. Observe that an edge $\{x, y\}$ is present in E_F if there exists a $z \in V(G)$ such that $\{x, y, z\}$

forms a triangle in G and $\{x,z\}$ and $\{y,z\}$ are in F. So, the probability that an edge $\{x,y\}$ is in $E(G[N_G(v) \cap V(F)]$ is at most $|\Gamma(\{x,y\})| \cdot \widetilde{\mathcal{O}}\left(\frac{1}{L}\right)$, where $\Gamma(\{x,y\})$ denotes the set of common neighbors of x and y in G. So, the expected number of BIS queries to enumerate all the edges in E_F is at most

$$\sum_{\{x,y\} \in E(G)} \widetilde{\mathcal{O}}\left(\frac{|\Gamma(\{x,y\}|)}{\sqrt{L}}\right) = \widetilde{\mathcal{O}}\left(\frac{T}{\sqrt{L}}\right).$$

Hence, the expected number of BIS queries made by the algorithm is $\widetilde{\mathcal{O}}\left(\frac{m+T}{\sqrt{L}}\right)$.