Triangle Estimation Using Tripartite Independent Set Queries

Anup Bhattacharya
Indian Statistical Institute, Kolkata, India

Arijit Bishnu
Indian Statistical Institute, Kolkata, India

Arijit Ghosh
Indian Statistical Institute, Kolkata, India

Gopinath Mishra
Indian Statistical Institute, Kolkata, India

Abstract

Estimating the number of triangles in a graph is one of the most fundamental problems in sublinear algorithms. In this work, we provide an approximate triangle counting algorithm using only polylogarithmic queries when *the number of triangles on any edge in the graph is polylogarithmically bounded*. Our query oracle *Tripartite Independent Set* (TIS) takes three disjoint sets of vertices A, B and C as input, and answers whether there exists a triangle having one endpoint in each of these three sets. Our query model generally belongs to the class of *group queries* (Ron and Tsur, ACM ToCT, 2016; Dell and Lapinskas, STOC 2018) and in particular is inspired by the *Bipartite Independent Set* (BIS) query oracle of Beame et al. (ITCS 2018). We extend the algorithmic framework of Beame et al., with TIS replacing BIS, for triangle counting using ideas from color coding due to Alon et al. (J. ACM, 1995) and a concentration inequality for sums of random variables with bounded dependency (Janson, Rand. Struct. Alg., 2004).

2012 ACM Subject Classification Theory of computation → Models of computation; Theory of computation → Streaming, sublinear and near linear time algorithms; Mathematics of computing → Probabilistic algorithms

Keywords and phrases Triangle estimation, query complexity, sublinear algorithm

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2019.19

Related Version A full version of the paper is available at `https://arxiv.org/abs/1808.00691`.

1 Introduction

Counting the number of triangles in a graph is a fundamental algorithmic problem in the RAM model [4, 10, 22], streaming [1, 2, 5, 11, 13, 24, 25, 26, 27, 28, 33] and the query model [17, 21]. In this work, we provide the first approximate triangle counting algorithm using only polylogarithmic queries to a query oracle named *Tripartite Independent Set* (TIS).

Notations, the query model, the problem and the result

We denote the set $\{1, \ldots, n\}$ by $[n]$. Let $V(G)$, $E(G)$ and $T(G)$ denote the set of vertices, edges and triangles in the input graph G, respectively. Let $t(G) = |T(G)|$. The statement A, B, C are disjoint, means A, B, C are pairwise disjoint. For three non-empty disjoint sets A, B, $C \subseteq V(G)$, $G(A, B, C)$, termed as a *tripartite subgraph* of G, denotes the induced subgraph of $A \cup B \cup C$ in G minus the edges having both endpoints in A or B or C. $t(A, B, C)$ denotes the number of triangles in $G(A, B, C)$. We use the triplet (a, b, c) to denote the triangle having a, b, c as its vertices. Let Δ_u denote the number of triangles having u as one of its vertices. Let $\Delta_{(u,v)}$ be the number of triangles having (u, v) as one of its edges

30th International Symposium on Algorithms and Computation (ISAAC 2019).
Editors: Pinyan Lu and Guochuan Zhang; Article No. 19; pp. 19:1–19:17
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

and $\Delta_E = \max_{(u,v) \in E(G)} \Delta_{(u,v)}$. For a set $\mathcal{U}$, "$\mathcal{U}$ is COLORED with [n]", means that each member of $\mathcal{U}$ is assigned a color out of $[n]$ colors independently and uniformly at random. Let $\mathbb{E}[X]$ and $\mathbb{V}[X]$ denote the expectation and variance of a random variable X. For an event $\mathcal{E}$, $\mathcal{E}^c$ denotes the complement of $\mathcal{E}$. The statement "a is an $1 \pm \epsilon$ multiplicative approximation of b" means $|b - a| \leq \epsilon \cdot b$. Next, we describe the query oracle.

Tripartite independent set oracle (TIS). Given three non-empty disjoint subsets $V_1, V_2, V_3 \subseteq V(G)$ of a graph G, TIS query oracle answers "YES" if and only if $t(V_1,\, V_2,\, V_3) \neq 0$.

Notice that the query oracle looks at only those triangles that have vertices in all of these sets V_1, V_2, V_3. The TRIANGLE-ESTIMATION problem is to report an $1 \pm \epsilon$ multiplicative approximation of $t(G)$ where the input is $V(G)$, TIS oracle for graph G and $\epsilon \in (0, 1)$.

▶ **Theorem 1.** *Let G be a graph with $\Delta_E \leq d$, $|V(G)| = n \geq 64$. For any $\epsilon > 0$,* TRIANGLE-ESTIMATION *can be solved using $\mathcal{O}\left(\epsilon^{-12} d^{12} \log^{25} n\right)$* TIS *queries with probability $1 - \mathcal{O}(n^{-2})$.*

Note that the query complexity stated in Theorem 1 is poly$(\log n, \frac{1}{\epsilon})$, even if d is $\mathcal{O}(\log^c n)$, where c is a positive constant. We reiterate that the only bound we require is on the number of triangles on an edge; neither do we require any bound on the maximum degree of the graph, nor do we require any bound on the number of triangles incident on a vertex.

Query models and TIS

Query models for graphs are essentially of two types:

Local Queries: This query model was initiated by Feige [19] and Goldreich and Ron [20] and used even recently by [17, 18]. The queries on the graphs are (i) degree query: the oracle reports the degree of a vertex; (ii) neighbor query: the oracle reports the i^{th} neighbor of v, if it exists; and (iii) edge existence query: the oracle reports whether there exists an edge between a given pair of vertices.

Group Queries or Subset queries and Subset samples: These queries were implicitly initiated in the works of Stockmeyer [31, 32] and formalized by Ron and Tsur [29]. *Group queries* can be viewed as a generalization of membership queries in sets. The essential idea of the group queries is to estimate the size of an unknown set $S \subseteq U$ by using a YES/NO answer from the oracle to the existence of an intersection between sets S and $T \subseteq U$; and give a uniformly selected item of $S \cap T$, if $S \cap T \neq \emptyset$ in the *subset sample* query. Subset sample queries are at least as powerful as group queries. The *cut query* by Rubinstein et al. [30], though motivated by submodular function minimization problem, can also be seen in the light of group queries – we seek the number of edges that intersect both the vertex sets that form a cut. Choi and Kim [12] used a variation of group queries for *graph reconstruction.* Dell and Lapinskas [14] essentially used this class of queries for estimating the number of edges in a bipartite graph. *Bipartite independent set* (BIS) queries for a graph, initiated by Beame et al. [6], can also be seen in the light of group queries. It provides a YES/NO answer to the existence of an edge in $E(G)$ that intersects with both $V_1, V_2 \subset V(G)$ of G, where V_1 and V_2 are disjoint. A subset sample version of BIS oracle was used in [9].

In TIS, we seek a YES/NO answer about the existence of an intersection between the set of triangles, that we want to estimate, and three disjoint sets of vertices. Thus TIS belongs to the class of group queries, as does BIS. A bone of contention for any newly introduced query oracle is its worth[1]. Beame et al. [6] had given a subjective justification in favor of BIS to establish it as a query oracle. It is easy to verify that TIS, being in the same class of group queries, have the interesting connections to group testing and computational geometry as BIS. We provide justifications in favor of considering $\Delta_E \leq d$ in Appendix A. Intuitively, TIS is to triangle counting what BIS is to edge estimation.

Prior works

Eden et al. [17] used $\tilde{\mathcal{O}}\left(\frac{|V(G)|}{t(G)^{1/3}} + \min\left\{\frac{|E(G)|^{3/2}}{t(G)}, |E(G)|\right\}\right)$[2] local queries to estimate the number of triangles. Their algorithmic results aided by an almost matching lower bound have almost closed this line of study. Matching upper and lower bounds on k-clique counting in G using local query model have also been reported [18]. A precursor to triangle estimation in graphs is edge estimation. The number of edges in a graph can be estimated by using $\tilde{\mathcal{O}}\left(n/\sqrt{m}\right)$ many degree and neighbor queries, and $\Omega\left(n/\sqrt{m}\right)$ queries are necessary to estimate the number of edges even if we allow all the three local queries [20]. This result would almost have closed the edge estimation problem but for having a relook at the problem with stronger query models and hoping for polylogarithmic number of queries. Beame et al. [6] precisely did that by estimating the number of edges in a graph using $\mathcal{O}\left(\epsilon^{-4}\log^{14} n\right)$ *bipartite independent set* (BIS) queries. Motivated by this result, we explore whether triangle estimation can be solved using only polylogarithmic queries to TIS.

Organization of the paper

We give a broad overview of the algorithm in Section 2. Sections 3, 4 and 5 give the details of sparsification, exact estimation and *coarse* estimation of the number of triangles, respectively. The final algorithm is given in Section 6. Appendix A provides justifications in favor of TIS. Appendix B has the probabilistic results used in this paper.

▶ **Remark 1.** Note that the TRIANGLE-ESTIMATION can also be thought of as HYPEREDGE ESTIMATION problem in a 3-uniform hypergrah. Very recently, Dell et al. [15] and Bhattacharya et al. [8], independently, showed that the bound on Δ_E is not necessary to solve TRIANGLE-ESTIMATION by using polylogarithmic many TIS queries. Also, both Dell et al. [15] and Bhattacharya et al. [8], independently, generalized our result to c-uniform hypergraphs, where $c \in \mathbb{N}$ is a constant.

2 Overview of the algorithm

Our algorithmic framework is inspired by [6] but the detailed analysis is markedly different using *color coding* due to Alon et al. [3] and a relatively new concentration inequality, due to Janson [23], for handling sums of random variables with bounded dependency. Apart from Lemmas 5 and 13, all other proofs require different ideas.

We feel that the analysis in [6] does not go through in our case because of a subtle difference between counting the number of triangles and the number of edges in a graph. An edge is an explicit structure, whereas, a triangle is an implicit structure – triangles (a, b, x), (b, c, y) and (a, c, z) in G imply the existence of the triangle (a, b, c) in G.

[1] See `http://www.wisdom.weizmann.ac.il/~oded/MC/237.html` for a comment on BIS.

[2] $\tilde{\mathcal{O}}(\cdot)$ hides a polynomial factor of $\log n$ and $\frac{1}{\epsilon}$, where $\epsilon \in (0, 1)$ is such that $(1-\epsilon)t \leq \hat{t} \leq (1+\epsilon)t$; $\hat{t}$ and t denote the estimated and actual number of triangles in G, respectively.

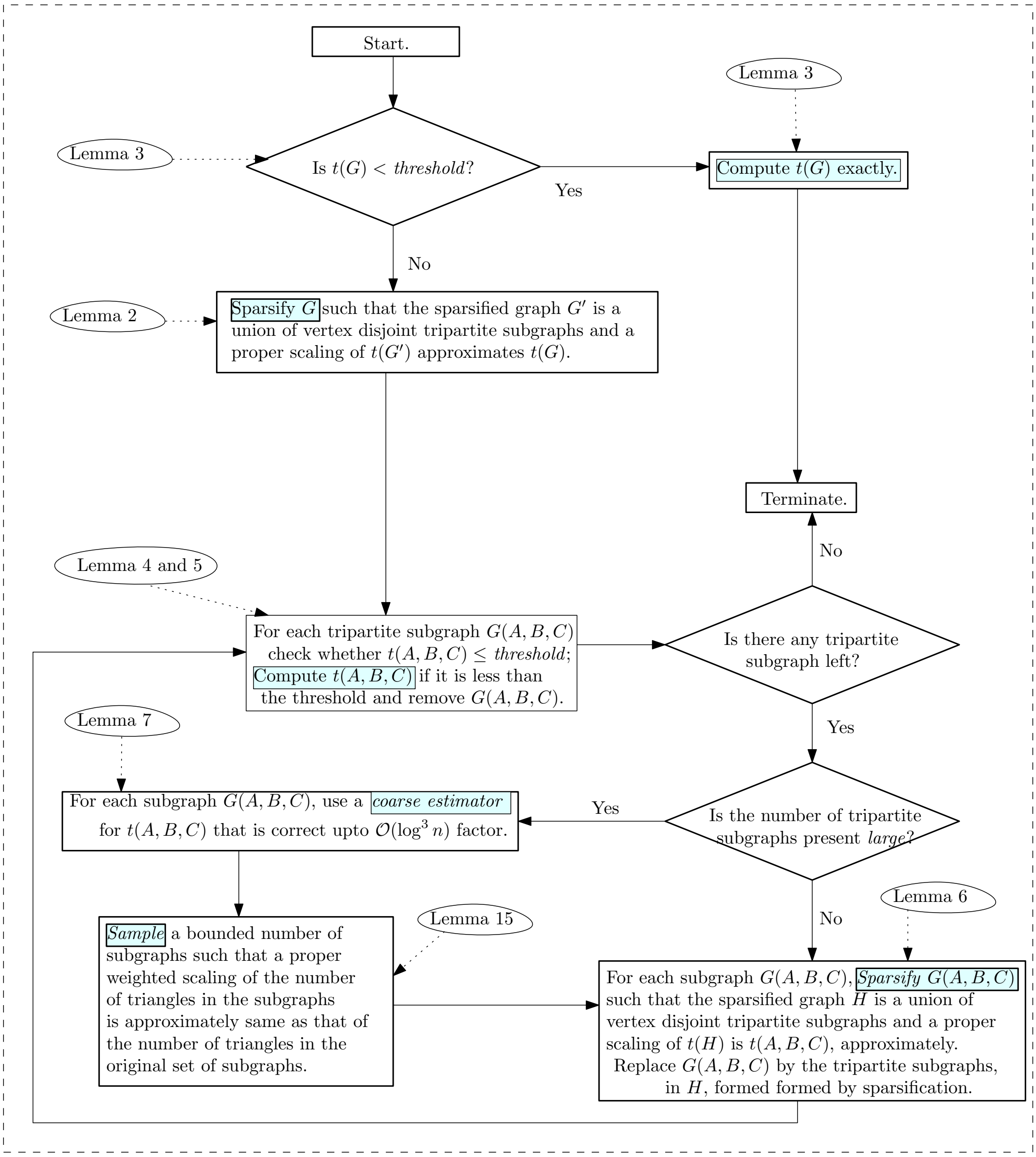
Start.
Lemma 3
Is $t(G) <$ *threshold*?
Yes
Compute $t(G)$ exactly.
Lemma 3
No
Lemma 2
Sparsify G such that the sparsified graph G' is a union of vertex disjoint tripartite subgraphs and a proper scaling of $t(G')$ approximates $t(G)$.
Terminate.
No
Lemma 4 and 5
For each tripartite subgraph $G(A,B,C)$ check whether $t(A,B,C) \leq$ *threshold*; Compute $t(A,B,C)$ if it is less than the threshold and remove $G(A,B,C)$.
Is there any tripartite subgraph left?
Yes
Lemma 7
For each subgraph $G(A,B,C)$, use a *coarse estimator* for $t(A,B,C)$ that is correct upto $\mathcal{O}(\log^3 n)$ factor.
Yes
Is the number of tripartite subgraphs present *large*?
No
Lemma 6
Lemma 15
Sample a bounded number of subgraphs such that a proper weighted scaling of the number of triangles in the subgraphs is approximately same as that of the number of triangles in the original set of subgraphs.
For each subgraph $G(A,B,C)$, *Sparsify* $G(A,B,C)$ such that the sparsified graph H is a union of vertex disjoint tripartite subgraphs and a proper scaling of $t(H)$ is $t(A,B,C)$, approximately. Replace $G(A,B,C)$ by the tripartite subgraphs, in H, formed formed by sparsification.

Figure 1 Flow chart of the algorithm. The highlighted texts indicate the basic building blocks of the algorithm. We also indicate the corresponding lemmas that support the building blocks.

In Figure 1, we give a *flowchart* of the algorithm and show the corresponding lemmas that support the steps of the algorithm. The main idea of our algorithm is as follows. We can figure out for a given G, if the number of triangles $t(G)$ is greater than or equal to a threshold τ (Lemma 3). If $t(G) < \tau$, i.e., G is sparse in triangles, we compute $t(G)$ exactly (Lemma 3). Otherwise, we sparsify G to get a disjoint union of tripartite subgraphs of G that maintain $t(G)$ up to a scaling factor (Lemma 2). For each tripartite subgraph, if the subgraph is sparse (decided by Lemma 4), we count the number of triangles exactly (Lemma 5). Otherwise, we again sparsify (Lemma 6). This repeated process of sparsification may create a huge number of tripartite subgraphs. Counting the number of triangles in them is managed by doing a coarse estimation (Lemma 7) and taking a sample of the subgraph that maintains the number of triangles *approximately*. Each time we sparsify, we ensure that the sum of the number of triangles in the subgraphs generated by sparsification is a constant fraction of the number of triangles in the graph before sparsification, making the number of iterations $O(\log n)$.

We sparsify G by considering the partition obtained when $V(G)$ is COLORED with $[3k]$. This sparsification is done such that: (i) the sparsified graph is a union of a set of vertex disjoint tripartite subgraphs and (ii) a proper scaling of the number of triangles in the sparsified graph is a *good* estimate of $t(G)$ with high probability[3]. The proof of the sparsification result stated next uses the *method of averaged bounded differences* and *Chernoff-Hoeffding type* inequality in bounded dependency setting by Janson [23]. The detailed proof is in Section 3. Recall that Δ_E is the maximum number of triangles on a particular edge.

▶ **Lemma 2** (General Sparsification). *Let $k, d \in \mathbb{N}$. There exists a constant κ_1 such that for any graph G with $\Delta_E \leq d$, if $V_1, \ldots, V_{3k}$ is a random partition of $V(G)$ obtained by $V(G)$ being* COLORED *with $[3k]$, then*

$$\mathbb{P}\left(\left|\frac{9k^2}{2}\sum_{i=1}^{k} t(V_i, V_{k+i}, V_{2k+i}) - t(G)\right| > \kappa_1 d k^2 \sqrt{t(G)} \log n\right) \leq \frac{2}{n^4}.$$

We apply the sparsification corresponding to Lemma 2 only when $t(G)$ is above a threshold[4] to ensure that the relative error is bounded. We can decide whether $t(G)$ is less than the threshold and if it is so, we compute the exact value of $t(G)$, using the following lemma, whose proof is inspired by color coding ideas [9] and given in Section 4.

▶ **Lemma 3** (Exact Counting). *There exists an algorithm that for any graph G and a threshold parameter $\tau \in \mathbb{N}$, determines whether $t(G) < \tau$ using $\mathcal{O}(\tau^6 \log n)$* TIS *queries with probability $1 - n^{-10}$. Moreover, the algorithm finds the exact value of $t(G)$ if $t(G) < \tau$.*

Assume that $t(G)$ is *large* [5] and G has undergone sparsification. We initialize a data structure with a set of vertex disjoint tripartite graphs that are obtained after the sparsification step. For each tripartite graph $G(A, B, C)$ in the data structure, we check whether $t(A, B, C)$ is less than a threshold using the algorithm corresponding to Lemma 4. If it is less than a threshold, we compute the exact value of $t(A, B, C)$ using Lemma 5 and remove $G(A, B, C)$ from the data structure. The proofs of Lemma 4 and 5 are given in the full version [7].

▶ **Lemma 4** (Threshold for Tripartite Graph). *There exists a deterministic algorithm that given any disjoint subsets $A, B, C \subset V(G)$ of any graph G and a threshold parameter $\tau \in \mathbb{N}$, can decide whether $t(A, B, C) \leq \tau$ using $\mathcal{O}(\tau \log n)$* TIS *queries.*

▶ **Lemma 5** (Exact Counting in Tripartite Graphs). *There exists a deterministic algorithm that given any disjoint subsets $A, B, C \subset V(G)$ of any graph G, can determine the exact value of $t(A, B, C)$ using $\mathcal{O}(t(A, B, C) \log n)$* TIS *queries.*

Now we are left with some tripartite graphs such that the number of triangles in each graph is more than a threshold. If the number of such graphs is not large, then we sparsify each tripartite graph $G(A, B, C)$ in a fashion almost similar to the earlier sparsification. This

[3] High probability means that the probability of success is at least $1 - \frac{1}{n^c}$ for some constant c.
[4] The threshold is a fixed polynomial in $d, \log n$ and $\frac{1}{\epsilon}$.
[5] Large refers to a fixed polynomial in $d, \log n$ and $\frac{1}{\epsilon}$

sparsification result formally stated in the following Lemma, has a proof similar to Lemma 2. We replace $G(A,B,C)$ by a constant (say, k) [6] many tripartite subgraphs formed after sparsification.

▶ **Lemma 6** (Sparsification for Tripartite Graphs). *Let $k, d \in \mathbb{N}$. There exists a constant κ_2 such that*

$$\mathbb{P}\left(\left|k^2 \sum_{i=1}^{k} t(A_i,B_i,C_i) - t(A,B,C)\right| > \kappa_2 d k^2 \sqrt{t(G)} \log n\right) \leq \frac{1}{n^8}$$

where A, B and C are disjoint subsets of $V(G)$ for any graph G with $\Delta_E \leq d$, and $A_1,\ldots,A_k$, $B_1,\ldots,B_k$ and $C_1,\ldots,C_k$ are the partitions of A,B,C formed uniformly at random, respectively.

If we have a large number of vertex disjoint tripartite subgraphs of G and each subgraph contains a large number of triangles, then we *coarsely* estimate the number of triangles in each subgraph which is correct up to $\mathcal{O}(\log^3 n)$ factor by using the algorithm corresponding to the following Lemma, whose proof is in Section 5. Our Coarse-Estimate algorithm is similar in structure to the coarse estimation algorithm for edge estimation, but the analysis involves sophisticated calculations.

▶ **Lemma 7** (Coarse Estimation). *There exists an algorithm that given disjoint subsets $A,B,C \subset V(G)$ of any graph G, returns an estimate $\hat{t}$ satisfying*

$$\frac{t(A,B,C)}{64 \log n} \leq \hat{t} \leq 64 t(A,B,C) \log^3 n$$

with probability $1 - n^{-9}$. Moreover, the query complexity of the algorithm is $\mathcal{O}(\log^4 n)$.

After estimating the number of triangles in each subgraph coarsely, we generate a bounded number of samples of the set of subgraphs using a sampling technique given by Beame et al. [6]. The sampling maintains the triangle count approximately. The Lemma corresponding to sampling is formally stated in Lemma 13 in Section 6. After getting the sample, we apply the sparsification algorithm corresponding to Lemma 6 for each subgraph in the sample.

Now again, for each tripartite graph $G(A,B,C)$, we check whether $t(A,B,C)$ is less than a threshold using the algorithm corresponding to Lemma 4. If yes, then we can compute the exact value of $t(A,B,C)$ using Lemma 5 and remove $G(A,B,C)$ from the data structure. Otherwise, we iterate on all the required steps discussed above as shown in Figure 1. Observe that the query complexity of each iteration is polylogarithmic [7]. Now, note that the number of triangles reduces by a constant factor after each sparsification step. So, the number of iterations is bounded by $\mathcal{O}(\log n)$. Hence, the query complexity of our algorithm is polylogarithmic. This completes the high level description of our algorithm.

[6] In our algorithm, k is a constant. However, Lemma 6 holds for any $k \in \mathbb{N}$.

[7] Polylogarithmic refers to a polynomial in $d, \log n$ and $\frac{1}{\epsilon}$

3 Sparsification Lemma

In this Section, we prove Lemma 2. The proof of Lemma 6 is similar.

Proof of Lemma 2. $V(G)$ is COLORED with $[3k]$. Let $V_1, \ldots, V_{3k}$ be the resulting partition of $V(G)$. Let Z_i be the random variable that denotes the color assigned to the i^{th} vertex. For $i \in [3k]$, $\pi(i)$ is a set of three colors defined as follows: $\pi(i) = \{i, (1 + (i + k - 1) \bmod 3k), (1 + (i + 2k - 1) \bmod 3k)\}$.

▶ **Definition 8.** *A triangle (a, b, c) is said to be* properly colored *if there exists a bijection in terms of coloring from $\{a, b, c\}$ to $\pi(i)$.*

Let $f(Z_1, \ldots, Z_n) = \sum_{i=1}^{k} t(V_i, V_{k+i}, V_{2k+i})$. Note that f is the number of triangles that are properly colored. The probability that a triangle is properly colored is $\frac{2}{9k^2}$. So, $\mathbb{E}[f] = \frac{2t(G)}{9k^2}$.

Let us focus on the instance when vertices $1, \ldots, t-1$ are already colored and we are going to color vertex t. Let S_ℓ (S_r) be the set of triangles in G having t as one of the vertices and other two vertices are from $[t-1]$ $([n] \setminus [t])$. $S_{\ell r}$ be the set of triangles in G such that t is a vertex and the second and third vertices are from $[t-1]$ and $[n] \setminus [t]$, respectively.

Given that the vertex t is colored with color $c \in [3k]$, let $N_\ell^c, N_r^c, N_{\ell r}^c$ be the random variables that denote the number of triangles in S_ℓ, S_r and $S_{\ell r}$ that are properly colored, respectively. Now, we can deduce the following about $\mathbb{E}_f^t$, the difference in the conditional expectation of the number of triangles that are properly colored whose t^{th} vertex is (possibly) differently colored, by considering the vertices in S_ℓ, S_r and $S_{\ell r}$ separately.

$$\begin{aligned}
\mathbb{E}_f^t &= \left|\mathbb{E}\left[f \mid Z_1, \ldots, Z_{t-1}, Z_t = a_t\right] - \mathbb{E}\left[f \mid Z_1, \ldots, Z_{t-1}, Z_t = a_t'\right]\right| \\
&= \left| N_\ell^{a_t} - N_\ell^{a_t'} + \mathbb{E}\left[N_r^{a_t} - N_r^{a_t'}\right] + \mathbb{E}\left[N_{\ell r}^{a_t} - N_{\ell r}^{a_t'}\right]\right| \\
&\leq \left| N_\ell^{a_t} - N_\ell^{a_t'}\right| + \mathbb{E}\left[\left|N_r^{a_t} - N_r^{a_t'}\right|\right] + \mathbb{E}\left[\left|N_{\ell r}^{a_t} - N_{\ell r}^{a_t'}\right|\right]
\end{aligned}$$

Now, consider the following claim, whose proof can be found in the full version [7].

▷ Claim 9. (a) $\mathbb{P}(\mid N_\ell^{a_t} - N_\ell^{a_t'} \mid < 8\sqrt{d\Delta_t \log n}) \geq 1 - 4n^{-8}$;
(b) $\mathbb{E}[\mid N_r^{a_t} - N_r^{a_t'} \mid] \leq \sqrt{d\Delta_t}/k$;
(c) $\mathbb{E}[\mid N_{\ell r}^{a_t} - N_{\ell r}^{a_t'} \mid] < 6d\sqrt{\Delta_t \log n}$. [8]
Let $c_t = 15d\sqrt{\Delta_t \log n}$. From the above claim, we have

$$\mathbb{E}_f^t < 8\sqrt{d\Delta_t \log n} + \frac{\sqrt{d\Delta_t}}{k} + 6d\sqrt{\Delta_t \log n} \leq 15d\sqrt{\Delta_t \log n} = c_t$$

with probability at least $1 - \frac{4}{n^8}$. Let $\mathcal{B}$ be the event that there exists $t \in [n]$ such that $\mathbb{E}_f^t > c_t$. By the union bound over all $t \in [n]$, $\mathbb{P}(\mathcal{B}) \leq \frac{4}{n^7}$.

Using the method of *averaged bounded difference* [16] (See Lemma 15 in Appendix B), we have

$$\mathbb{P}\left(|f - \mathbb{E}[f]| > \delta + t(G)\mathbb{P}(\mathcal{B})\right) \leq e^{-\delta^2 / \sum\limits_{t=1}^{n} c_t^2} + \mathbb{P}(\mathcal{B}).$$

[8] Note that Δ_t is the number of triangles having t as one of its vertices and we are not assuming any bound on Δ_t. We assume Δ_E, that is number of triangles on any edge, is bounded.

We set $\delta = 60d\sqrt{t(G)}\log n$. Observe that $\sum\limits_{t=1}^{n} c_t^2 = 225d^2\log n \sum\limits_{t=1}^{n} \Delta_t = 675d^2 t(G)\log n$. Hence,

$$\mathbb{P}\left(\left|f - \frac{2t(G)}{9k^2}\right| > 60d\sqrt{t(G)}\log n + t(G)\mathbb{P}(\mathcal{B})\right) \leq \frac{1}{n^4} + \frac{1}{n^7},$$

that is,

$$\mathbb{P}\left(\left|\frac{9k^2}{2}f - t(G)\right| > 270dk^2\sqrt{t(G)}\log n + \frac{9k^2}{2}\cdot\frac{t(G)}{n^7}\right) \leq \frac{1}{n^4} + \frac{1}{n^7}.$$

Since, $\frac{9k^2}{2}\cdot\frac{t(G)}{n^7} < dk^2\sqrt{t(G)}\log n$, we get

$$\mathbb{P}\left(\left|\frac{9k^2}{2}f - t(G)\right| > 271dk^2\sqrt{t(G)}\log n\right) \leq \frac{2}{n^4}. \qquad ◀$$

4 Proof of the Lemmas corresponding to exact estimation

In this Section, we prove Lemma 3. The proofs of 4 and 5 can be found in the full version [7].

Proof of Lemma 3. We color $V(G)$ with $[100\tau^2]$ colors. Let $h : V(G) \to [100\tau^2]$ be the coloring function and $V_i = \{v \in V(G) : h(v) = i\}$, i.e., the vertices with color i, where $i \in [100\tau^2]$. Note that $V_1, \ldots, V_{100\tau^2}$ forms a partition of $V(G)$. We make TIS queries with input V_i, V_j, V_k for each $1 \leq i < j < k \leq 100\tau^2$. Observe that we make $\mathcal{O}(\tau^6)$ TIS queries. We construct a 3-uniform hypergraph $\mathcal{H}$, where $U(\mathcal{H}) = \{V_1, \ldots, V_{100\tau^2}\}$ [9] and $(V_i, V_j, V_k) \in \mathcal{F}(\mathcal{H})$ if and only if TIS oracle answers yes with V_i, V_j, V_k given as input. We repeat the above procedure γ times, where $\gamma = 50\log n$. Let $\mathcal{H}_1, \ldots, \mathcal{H}_\gamma$ be the set of corresponding hypergraphs and h_i be the coloring function to form the hypergraph $\mathcal{H}_i$, where $i \in [\gamma]$. Then we compute $A = \max\{|\mathcal{F}(\mathcal{H}_1)|, \ldots, |\mathcal{F}(\mathcal{H}_\gamma)|\}$. If $A \geq \tau$, we report $t(G) \geq \tau$. Otherwise, we report A as $t(G)$. Note that the total number of TIS queries is $\mathcal{O}(\tau^6\log n)$. Now, we analyze the cases $t(G) \geq \tau$ and $t(G) < \tau$ separately.

(i) $t(G) \geq \tau$: Consider a fixed set T of τ triangles. Let T_v be the set of vertices that is present in some triangle in T. Observe that $|T_v| \leq 3\tau$. Let $\mathcal{E}_i$ be the event that the vertices in T_v are uniquely colored by the function h_i, i.e., $\mathcal{E}_i : h_i(u) = h_i(v)$ if and only if $u = v$, where $u, v \in T_v$. First we prove that $\mathbb{P}(\mathcal{E}) \geq \frac{9}{10}$ by computing $\mathbb{P}(\mathcal{E}_i^c)$.

$$\mathbb{P}\left(\mathcal{E}_i^c\right) \leq \sum_{u,v\in T_v} \mathbb{P}(h_i(u) = h_i(v)) \leq \sum_{u,v\in T_v} \frac{1}{100\tau^2} \leq \frac{|T_v|^2}{100\tau^2} < \frac{1}{10}.$$

Let $\textsc{Prop}_i$ be the property that for each triangle $z \in T$, there is a corresponding hyperedge in $\mathcal{F}(\mathcal{H}_i)$, where $i \in [\gamma]$. Specifically, for each triangle $(a_1, a_2, a_3) \in T$ there exists a hyperedge $(a_1', a_2', a_3') \in \mathcal{F}(\mathcal{H}_i)$ such that $h_i(a_j) = h_i(a_j')$ for each $j \in [3]$. Note that, if $\textsc{Prop}_i$ holds, then $|\mathcal{F}(\mathcal{H}_i)| \geq |T| \geq \tau$. By the definition of TIS oracle, $\textsc{Prop}_i$ holds when the event $\mathcal{E}_i$ occurs, i.e., $\textsc{Prop}_i$ holds with probability at least $\frac{9}{10}$. This implies, with probability $\frac{9}{10}$, $|\mathcal{F}(\mathcal{H}_i)| \geq \tau$. Recall that $A = \max\{|\mathcal{F}(\mathcal{H}_1)|, \ldots, |\mathcal{F}(\mathcal{H}_\gamma)|\}$ and $\gamma = 50\log n$. So, $\mathbb{P}(A < \tau) = \left(1 - \frac{9}{10}\right)^{50\log n} \leq \frac{1}{n^{10}}$. Hence, if $t(G) \geq \tau$, our algorithm detects it with probability at least $1 - \frac{1}{n^{10}}$.

[9] $U(\mathcal{H})$ and $\mathcal{F}(\mathcal{H})$ denote the set of vertices and hyperedges in a hypergraph $\mathcal{H}$, respectively.

(ii) $t(G) < \tau$: Let T be the set of all $t(G)$ triangles in G and T_v be the set of vertices that is present in some triangle in T. Observe that $|T_v| \leq 3 \cdot t(G) < 3\tau$. Let $\mathcal{E}_i$ be the event that the vertices in T_v are uniquely colored by the function h_i, i.e., $\mathcal{E}_i : h_i(u) = h_i(v)$ if and only if $u = v$, where $u, v \in T_v$. First we prove that $\mathbb{P}(\mathcal{E}_i) \geq \frac{9}{10}$ by computing $\mathbb{P}(\mathcal{E}_i^c)$.

$$\mathbb{P}\left(\mathcal{E}_i^c\right) \leq \sum_{u,v \in T_v} \mathbb{P}(h_i(u) = h_i(v)) \leq \sum_{u,v \in T_v} \frac{1}{100\tau^2} \leq \frac{|T_v|^2}{100\tau^2} < \frac{1}{10}.$$

Let PROP_i be the property that for each triangle $z \in T$, there is a corresponding hyperedge in $\mathcal{F}(\mathcal{H}_i)$, where $i \in [\gamma]$. Specifically, for each triangle $(a_1, a_2, a_3) \in T$ there exists a hyperedge $(a_1', a_2', a_3') \in \mathcal{F}(\mathcal{H}_i)$ such that $h_i(a_j) = h_i(a_j')$ for each $j \in [3]$. Note that, if PROP_i holds, then $|\mathcal{F}(\mathcal{H}_i)| = t(G)$. By the definition of TIS oracle, PROP_i holds when the event $\mathcal{E}_i$ occurs, i.e., PROP_i holds with probability at least $\frac{9}{10}$. This implies, with probability $\frac{9}{10}$, $|\mathcal{F}(\mathcal{H}_i)| = t(G)$. Recall that $A = \max\{|\mathcal{F}(\mathcal{H}_1), \ldots, \mathcal{F}(\mathcal{H}_\gamma)|\}$ and $\gamma = 50 \log n$. By the construction of $\mathcal{H}_i$, $|\mathcal{F}(\mathcal{H}_i)| \leq t(G)$. So, $A \leq t(G)$ and $\mathbb{P}(A \neq t(G)) = \mathbb{P}(A < t(G)) \leq \left(1 - \frac{9}{10}\right)^{50 \log n} \leq \frac{1}{n^{10}}$. Hence, if $t(G) < \tau$, our algorithm outputs the exact value of $t(G)$ with probability at least $1 - \frac{1}{n^{10}}$. ◀

5 Proof of the Lemma corresponding to coarse estimation

We now prove Lemma 7. Algorithm 2 corresponds to Lemma 7. Algorithm 1 is a subroutine in Algorithm 2. Algorithm 1 determines whether a given estimate $\hat{t}$ is correct upto a $\mathcal{O}(\log^3 n)$ factor. Lemmas 10 and 11 are intermediate results needed to prove Lemma 7.

Algorithm 1 VERIFY-ESTIMATE $(A, B, C, \hat{t})$.

```
Input: Three pairwise disjoint set $A, B, C \subseteq V(G)$ and $\hat{t}$.
Output: If $\hat{t}$ is a good estimate, then ACCEPT. Otherwise, REJECT.
begin
    for ($i = 2\log n$ to $0$) do
        for ($j = \log n$ to $0$) do
            Find $A_{ij} \subseteq A$, $B_{ij} \subseteq B$, $C_{ij} \subseteq C$ by sampling each element of $A$, $B$ and $C$,
              respectively with probability $\min\{\frac{2^i}{\hat{t}}, 1\}$, $\min\{\frac{2^j}{2^i}\log n, 1\}$, $\frac{1}{2^j}$,
              respectively.
            if ($t(A_{ij}, B_{ij}, C_{ij}) \neq 0$) then
                ACCEPT
    REJECT
```

▶ **Lemma 10.** *If $\hat{t} \geq 64 t(A, B, C) \log^3 n$, $\mathbb{P}(\text{VERIFY-ESTIMATE } (A, B, C, \hat{t}) \text{ accepts}) \leq \frac{1}{20}$.*

Proof. Let $T(A, B, C)$ denote the set of triangles having vertices $a \in A$, $b \in B$ and $c \in C$, where A, B and C are disjoint subsets of $V(G)$. For $(a, b, c) \in T(A, B, C)$ such that $a \in A, b \in B, c \in C$, let $X_{(a,b,c)}^{ij}$ denote the indicator random variable such that $X_{(a,b,c)}^{ij} = 1$ if and only if $(a, b, c) \in T(A_{ij}, B_{ij}, C_{ij})$ and $X_{ij} = \sum\limits_{(a,b,c) \in T(A,B,C)} X_{(a,b,c)}^{ij}$. Note that $t(A_{ij}, B_{ij}, C_{ij}) = X_{ij}$. (a, b, c) is present in $T(A_{ij}, B_{ij}, C_{ij})$ if $a \in A_{ij}$, $b \in B_{ij}$ and $c \in C_{ij}$. So,

$$\mathbb{P}\left(X_{(a,b,c)}^{ij} = 1\right) \leq \frac{2^i}{\hat{t}} \cdot \frac{2^j}{2^i} \log n \cdot \frac{1}{2^j} = \frac{\log n}{\hat{t}} \text{ and } \mathbb{E}\left[X_{ij}\right] \leq \frac{t(A, B, C)}{\hat{t}} \log n.$$

As $X_{ij} \geq 0$,

$$\mathbb{P}\left(X_{ij} \neq 0\right) = \mathbb{P}(X_{ij} \geq 1) \leq \mathbb{E}\left[X_{ij}\right] \leq \frac{t(A,B,C)}{\hat{t}} \log n.$$

Now using the fact that $\hat{t} \geq 64 t(A,B,C) \log^3 n$, we have $\mathbb{P}\left(X_{ij} \neq 0\right) \leq \frac{1}{64 \log^2 n}$. Observe that VERIFY-ESTIMATE accepts if and only if there exists $i, j \in \{0, \ldots, \log n\}$ such that $X_{ij} \neq 0$. Using the union bound, we get

$$\begin{aligned} \mathbb{P}(\text{VERIFY-ESTIMATE accepts}) &\leq \sum_{0 \leq i \leq 2\log n} \sum_{0 \leq j \leq \log n} \mathbb{P}(X_{ij} \neq 0) \\ &\leq \frac{(2\log n + 1)(\log n + 1)}{32 \log^2 n} \\ &\leq \frac{1}{20}. \end{aligned}$$

◀

▶ **Lemma 11.** *If $\hat{t} \leq \frac{t(A,B,C)}{32 \log n}$, $\mathbb{P}($VERIFY-ESTIMATE $(A, B, C, \hat{t})$ accepts$) \geq \frac{1}{5}$.*

Proof. For $p \in \{0, \ldots, 2\log n\}$, let $A^p \subseteq A$ be the set of vertices such that for each $a \in A_p$, the number of triangles of the form (a, b, c) with $(b, c) \in B \times C$, lies between 2^p and $2^{p+1} - 1$.

For $a \in A^p$ and $q \in \{0, \ldots, \log n\}$, let $B^{pq}(a) \subseteq B$ is the set of vertices such that for each $b \in B$, the number of triangles of the form (a, b, c) with $c \in C$ lies between 2^q and $2^{q+1} - 1$ We need the following Claim to proceed further.

▷ Claim 12.
(i) There exists $p \in \{0, \ldots, 2\log n\}$ such that $|A^p| > \frac{t(A,B,C)}{2^{p+1}(2\log n+1)}$.
(ii) For each $a \in A^p$, there exists $q \in \{0, \ldots, \log n\}$ such that $|B^{pq}(a)| > \frac{2^p}{2^{q+1}(\log n+1)}$.

Proof.
(i) Observe that $t(A,B,C) = \sum_{p=0}^{2\log n} t(A^p, B, C)$ as the sum takes into account all incidences of vertices in A. So, there exists $p \in \{0, \ldots, 2\log n\}$ such that $t(A^p, B, C) \geq \frac{t(A,B,C)}{2\log n+1}$. From the definition of A^p, $t(A^p, B, C) < |A^p| \cdot 2^{p+1}$. Hence, there exists $p \in \{0, \ldots, 2\log n\}$ such that

$$|A^p| > \frac{t(A^p, B, C)}{2^{p+1}} \geq \frac{t(A,B,C)}{2^{p+1}(2\log n + 1)}.$$

(ii) Observe that $\sum_{q=0}^{\log n} t(\{a\}, B^{pq}(a), C) = t(\{a\}, B, C)$. So, there exists $q \in \{0, \ldots, \log n\}$ such that $t(\{a\}, B^{pq}(a), C) \geq \frac{t(\{a\},B,C)}{\log n+1}$. From the definition of $B^{pq}(a)$, $t(\{a\}, B^{pq}(a), C) < |B^{pq}(a)| \cdot 2^{q+1}$. Hence, there exists $q \in \{0, \ldots, \log n\}$ such that

$$|B^{pq}(a)| > \frac{t(\{a\}, B^{pq}(a), C)}{2^{q+1}} \geq \frac{t(\{a\}, B, C)}{2^{q+1}(\log n + 1)} \geq \frac{2^p}{2^{q+1}(\log n + 1)}.$$

◁

We come back to the proof of Lemma 11. We will show that VERIFY-ESTIMATE accepts with probability at least $\frac{1}{5}$ when loop executes for $i = p$, where p is such that $|A^p| > \frac{t(A,B,C)}{2^{p+1}(2\log n+1)}$. The existence of such a p is evident from (i) of Claim 12.

Recall that $A_{pq} \subseteq A$, $B_{pq} \subseteq B$ and $C_{pq} \subseteq C$ are the samples obtained when the loop variables i and j in Algorithm 1 attain values p and q, respectively. Observe that

$$\mathbb{P}\left(A_{pq} \cap A^p = \emptyset\right) \leq \left(1 - \frac{2^p}{\hat{t}}\right)^{|A^p|} \leq e^{-\frac{2^p}{\hat{t}}|A^p|} \leq e^{-\frac{2^p}{\hat{t}} \frac{t(A,B,C)}{2^{p+1} \log n}} = e^{-\frac{t(A,B,C)}{2\hat{t}(2\log n+1)}}.$$

Now using the fact that $\hat{t} \leq \frac{t(A,B,C)}{32 \log n}$ and $n \geq 64$,

$$\mathbb{P}\left(A_{pq} \cap A^p = \emptyset\right) \leq \frac{1}{e^6}.$$

Assume that $A_{pq} \cap A^p \neq \emptyset$ and $a \in A_{pq} \cap A^p$. By (ii) of Claim 12, there exists $q \in \{0, \ldots, \log n\}$, such that $B^{pq}(a) \geq \frac{2^p}{2^{q+1}(\log n+1)}$. Observe that we will be done, if we can show that Verify-Estimate accepts when loop executes for $i = p$ and $j = q$. Now,

$$\mathbb{P}\left(B_{pq} \cap B^{pq}(a) = \emptyset \mid A_{pq} \cap A^p \neq \emptyset\right) \leq \left(1 - \frac{2^q}{2^p} \log n\right)^{|B^{pq}(a)|} \leq \frac{1}{e^{3/7}}.$$

Assume that $A_{pq} \cap A^p \neq \emptyset$, $B_{pq} \cap B^{pq}(a) \neq \emptyset$ and $b \in B_{pq} \cap B^{pq}(a)$. Let S be the set such that (a, b, s) is a triangle in G for each $s \in S$. Note that $|S| \geq 2^q$. So,

$$\mathbb{P}\left(C_{pq} \cap S = \emptyset \mid A_{pq} \cap A^p \neq \emptyset \text{ and } B_{pq} \cap B^{pq}(a) \neq \emptyset\right) \leq \left(1 - \frac{1}{2^q}\right)^{2^q} \leq \frac{1}{e}.$$

Observe that Verify-Estimate accepts if $t(A_{pq}, B_{pq}, C_{pq}) \neq 0$. Also, $t(A_{pq}, B_{pq}, C_{pq}) \neq 0$ if $A_{pq} \cap A^p \neq \emptyset$, $B_{pq} \cap B^{pq}(a) \neq \emptyset$ and $C_{pq} \cap S \neq \emptyset$. Hence,

$$\begin{aligned}
\mathbb{P}(\text{Verify-Estimate accepts}) &\geq \mathbb{P}\left(A_{pq} \cap A^p \neq \emptyset,\ B_{pq} \cap B^{pq}(a) \neq \emptyset \text{ and } C_{pq} \cap S \neq \emptyset\right) \\
&= \mathbb{P}\left(A_{pq} \cap A^p \neq \emptyset\right) \cdot \mathbb{P}\left(B_{pq} \cap B^{pq}(a) \neq \emptyset \mid A_{pq} \cap A^p \neq \emptyset\right) \\
&\quad \cdot \mathbb{P}\left(C_{pq} \cap S \neq \emptyset \mid A_{pq} \cap A^p \neq \emptyset \text{ and } B_{pq} \cap B^{pq}(a) \neq \emptyset\right) \\
&> \left(1 - \frac{1}{e^6}\right)\left(1 - \frac{1}{e^{3/7}}\right)\left(1 - \frac{1}{e}\right) > \frac{1}{5}.
\end{aligned}$$

◀

Algorithm 2 Coarse-Estimate (A, B, C).

```
Input: Three pairwise disjoint sets $A, B, C \subset V(G)$.
Output: An estimate $\hat{t}$ for $t(A, B, C)$.
begin
    for ( $\hat{t} = n^3, n^3/2, \ldots, 1$) do
        Repeat Verify-Estimate $(A, B, C, \hat{t})$ for $\Gamma = 2000 \log n$ times. If at least $\frac{\Gamma}{10}$
          many Verify-Estimate accepts, then output $\hat{t}$.
```

Proof of Lemma 7. Note that an execution of Coarse-Estimate for a particular $\hat{t}$, repeats Verify-Estimate for $\Gamma = 2000 \log n$ times and gives output $\hat{t}$ if at least $\frac{\Gamma}{10}$ many Verify-Estimate accepts. For a particular $\hat{t}$, let X_i be the indicator random variable such that $X_i = 1$ if and only if the i^{th} execution of Verify-Estimate accepts. Also take $X = \sum_{i=1}^{\Gamma} X_i$. Coarse-Estimate gives output $\hat{t}$ if $X > \frac{\Gamma}{10}$.

Consider the execution of Coarse-Estimate for a particular $\hat{t}$. If $\hat{t} \geq 32 t(A,B,C) \log^3 n$, we first show that Coarse-Estimate accepts with probability $1 - \frac{1}{n^5}$. Recall Lemma 10. If $\hat{t} \geq 64 t(A,B,C) \log^3 n$, $\mathbb{P}(X_i = 1) \leq \frac{1}{20}$ and hence $\mathbb{E}[X] \leq \frac{\Gamma}{20}$. By using Chernoff-Hoeffding's inequality (See (i) of Lemma 17 in Appendix B),

$$\mathbb{P}\left(X > \frac{\Gamma}{10}\right) = \mathbb{P}\left(X > \frac{\Gamma}{20} + \frac{\Gamma}{20}\right) \leq \frac{1}{n^{10}}.$$

By using the union bound for all $\hat{t}$, the probability that Coarse-Estimate outputs some $\hat{t} \geq 16 t(A,B,C) \log^3 n$, is at most $\frac{3 \log n}{n^{10}}$.

Now consider the instance when the for loop in Coarse-Estimate executes for a $\hat{t}$ such that $\hat{t} \leq \frac{t(A,B,C)}{32 \log n}$. In this situation, $\mathbb{P}(X_i = 1) \geq \frac{1}{5}$. So, $\mathbb{E}[X] \geq \frac{\Gamma}{5}$. By using Chernoff-Hoeffding's inequality (See (ii) of Lemma 17 in Appendix B),

$$\mathbb{P}\left(X \leq \frac{\Gamma}{10}\right) \leq \mathbb{P}\left(X < \frac{3\Gamma}{20}\right) = \mathbb{P}\left(X < \frac{\Gamma}{5} - \frac{\Gamma}{20}\right) \leq \frac{1}{n^{10}}.$$

By using the union bound for all $\hat{t}$, the probability that Coarse-Estimate outputs some $\hat{t} \leq \frac{t(A,B,C)}{16 \log n}$, is at most $\frac{3 \log n}{n^{10}}$.

Observe that, Coarse-Estimate gives output $\hat{t}$ that satisfies either $\hat{t} \geq 64 t(A,B,C) \log^3 n$ or $\hat{t} \leq \frac{t(A,B,C)}{32 \log n}$ is at most $\frac{3 \log n}{n^{10}} + \frac{3 \log n}{n^{10}} \leq \frac{1}{n^9}$.

Putting everything together, Coarse-Estimate gives some $\hat{t}$ as output with probability at least $1 - \frac{1}{n^9}$ satisfying

$$\frac{t(A,B,C)}{64 \log n} \leq \hat{t} \leq 64 t(A,B,C) \log^3 n.$$

From the description of Verify-Estimate and Coarse-Estimate, the query complexity of Verify-Estimate is $\mathcal{O}(\log^2 n)$ and Coarse-Estimate calls Verify-Estimate $\mathcal{O}(\log^2 n)$ times. Hence, Coarse-Estimate makes $\mathcal{O}(\log^4 n)$ many queries. ◀

6 The final triangle estimation algorithm: Proof of Theorem 1

Now we design our algorithm for $1 \pm \epsilon$ multiplicative approximation of $t(G)$. If $\epsilon \leq \frac{d \log^2 n}{n^{1/4}}$, we query for $t(\{a\},\{b\},\{c\})$ for all distinct $a, b, c \in V(G)$ and compute the exact value of $t(G)$. So, we assume that $\epsilon > \frac{d \log^2 n}{n^{1/4}}$.

We build a data structure such that it maintains two things at any point of time. (i) An accumulator ψ for the number of triangles. We initialize $\psi = 0$. (ii) A set of tuples $(A_1, B_1, C_1, w_1), \ldots, (A_\zeta, B_\zeta, C_\zeta, w_\zeta)$, where tuple (A_i, B_i, C_i) corresponds to the tripartite subgraph $G(A_i, B_i, C_i)$ and w_i is the weight associated to $G(A_i, B_i, C_i)$. Initially, there is no tuple in our data structure. The algorithm will proceed as follows.

(1) **(Exact Counting)** Fix the threshold τ as $\frac{36\kappa_1^2 d^2 \log^4 n}{\epsilon^2}$. Decide whether $t(G) < \tau$ by using the result of Lemma 3, where κ_1 is the constant mentioned in Lemma 2. If yes, we terminate by reporting the exact value of $t(G)$. Otherwise, we go to Step-2. The query complexity of Step-1 is $\mathcal{O}(\tau^6 \log n) = \mathcal{O}\left(\frac{d^{12} \log^{25} n}{\epsilon^{12}}\right)$.

(2) **(General Sparsification)** $V(G)$ is COLORED with $[3k]$ for $k = 1$. Let A, B, C be the partition generated by the coloring of $V(G)$. We initialize the data structure by setting $\psi = 0$ and adding the tuple $(A, B, C, 9/2)$ to the data structure. Note that no query is required in this step. The constant $9/2$ is obtained by putting $k = 1$ in Lemma 2.

(3) We repeat steps 4 to 7 until there is no tuple left in the data structure. We maintain an invariant that the number of tuples stored in the data structure, is at most $\frac{10\kappa_3 \log^{16} n}{\epsilon^2}$, where κ_3 is a constant to be fixed later.

(4) **(Threshold for Tripartite Graph and Exact Counting in Tripartite Graphs)** For each tuple (A, B, C, w) in the data structure, we determine whether $t(A,B,C) \leq \frac{36\kappa_2^2 d^2 \log^4 n}{\epsilon^2}$, the threshold, by using the deterministic algorithm coresponding to Lemma 3 with $\mathcal{O}(\frac{d^2 \log^4 n}{\epsilon^2} \cdot \log n) = \mathcal{O}(\frac{d^2 \log^5 n}{\epsilon^2})$ many queries, where κ_2 is the constant mentioned in Lemma 6. If yes, we find $t(A,B,C)$ using $\mathcal{O}(\frac{d^2 \log^5 n}{\epsilon^2})$ many queries and add $w \cdot t(A,B,C)$ to ψ. We remove all (A,B,C)'s for which the algorithm found that $t(A,B,C)$ is below the threshold. As there are at most $\mathcal{O}\left(\frac{\log^{16} n}{\epsilon^2}\right)$ many triples at any time, the number of queries made in each iteration of the algorithm is $\mathcal{O}\left(\frac{d^2 \log^5 n}{\epsilon^2} \cdot \frac{\log^{16} n}{\epsilon^2}\right) = \mathcal{O}\left(\frac{d^2 \log^{21} n}{\epsilon^4}\right)$.

(5) Note that each tuple (A, B, C, w) in this step is such that $t(A, B, C) > \frac{36\kappa_2^2 d^2 \log^4 n}{\epsilon^2}$. Let $(A_1, B_1, C_1, w_1), \ldots, (A_r, B_r, C_r, w_r)$ be the set of tuples stored at the current instant. If $r > \frac{10\kappa_3 \log^{16} n}{\epsilon^2}$, we go to Step 6. Otherwise, we go to Step 7.

(6) **(Coarse Estimation and Sampling)** For each tuple (A, B, C, w) in the data structure, we find an estimate $\hat{t}$ such that $\frac{t(A,B,C)}{64 \log^3 n} < \hat{t} < 64t(A, B, C) \log^3 n$. This can be done due to Lemma 7 and the number of queries is $\mathcal{O}\left(\log^4 n\right)$ per tuple. As the algorithm executes the current step, the number of tuples in our data structure is *large*. We take a sample from the set of tuples such that the sample maintains the required estimate *approximately* by using Lemma 13, that follows from a Lemma by Beame et al. [6]. The original statement of Beame et al. is given in Lemma 20 in Appendix B.

▶ **Lemma 13** ([6]). *Let $(A_1, B_1, C_1, w_1), \ldots, (A_r, B_r, C_r, w_r)$ be the tuples present in the data structure and e_i be the corresponding coarse estimation for $t(A_i, B_i, C_i), i \in [r]$, such that* ***(i)*** *$w_i, e_i \geq 1, \forall i \in [r]$;* ***(ii)*** *$\frac{e_i}{\rho} \leq t(A_i, B_i, C_i) \leq e_i\rho$ for some $\rho > 0$ and $\forall\ i \in [r]$; and* ***(iii)*** *$\sum_{i=1}^{r} w_i \cdot t(A_i, B_i, C_i) \leq M$. Note that the exact values $t(A_i, B_i, C_i)$'s are not known to us. Then there exists an algorithm that finds $(A_1', B_1', C_1', w_1'), \ldots, (A_s', B_s', C_s', w_s')$ such that all of the above three conditions hold and $\left|\sum\limits_{i=1}^{s} w_i' \cdot t(A_i', B_i', C_i') - \sum\limits_{i=1}^{r} w_i \cdot t(A_i, B_i, C_i)\right| \leq \lambda S$ with probability $1 - \delta$; where $S = \sum_{i=1}^{r} w_i \cdot t(A_i, B_i, C_i)$ and $\lambda, \delta > 0$. Also, $s = \mathcal{O}\left(\lambda^{-2}\rho^4 \log M \left(\log\log M + \log \frac{1}{\delta}\right)\right)$.*

We use the algorithm corresponding to Lemma 13 with $\lambda = \frac{\epsilon}{6 \log n}$, $\rho = 64 \log^3 n$ and $\delta = \frac{1}{n^{10}}$ to find a new set of tuples $(A_1', B_1', C_1', w_1'), \ldots, (A_s', B_s', C_s', w_s')$ such that $|S - \sum_{i=1}^{s} w_i' t(A', B', C')| \leq \lambda S$ with probability $1 - \frac{1}{n^{10}}$, where $S = \sum_{i=1}^{r} w_i t(A_i, B_i, C_i)$ and $s = \frac{\kappa_3 \log^{16} n}{\epsilon^2}$ for some constant $\kappa_3 > 0$. This κ_3 is same as the one mentioned in Step 3. No query is required to exucute the algorithm of Lemma 13. Recall that the number of tuples present at any time is $\mathcal{O}\left(\frac{\log^{16} n}{\epsilon^2}\right)$. Hence, the number of queries in this step in each iteration, is $\mathcal{O}(\frac{\log^{16} n}{\epsilon^2} \cdot \log^4 n) = \mathcal{O}(\frac{\log^{20} n}{\epsilon^2})$.

(7) **(Sparsification for Tripartite Graphs)** We partition each of A, B and C into 3 parts uniformly at random. Let $A = U_1 \uplus U_2 \uplus U_3$; $V = V_1 \uplus V_2 \uplus V_3$ and $W = W_1 \uplus W_2 \uplus W_3$. We delete (A, B, C, w) from the data structure and add $(U_i, V_i, W_i, 9w)$ for each $i \in [3]$ to our data structure. Note that no query is made in this step.

(8) Report ψ as the estimate for the number of triangles in G, when no tuples are left.

First, we prove that the above algorithm produces a $(1 \pm \epsilon)$ multiplicative approximation to $t(G)$ for any $\epsilon > 0$ with high probability. If $t(G) \leq \frac{36\kappa_1^2 d^2 \log^4 n}{\epsilon^2}$, then the algorithm terminates in Step-1 and reports the exact number of triangles with probability $1 - \frac{1}{n^{10}}$ by Lemma 3. Otherwise, the algorithm proceeds to Step-2. In Step-2, the algorithm colors $V(G)$ using three colors and incurs a multiplicative error of $1 \pm \epsilon_0$ to $t(G)$, where $\epsilon_0 = \frac{\kappa_1 d \log n}{\sqrt{t(G)}}$. As $t(G) > \frac{36\kappa_1^2 d^2 \log^4 n}{\epsilon^2}$ and $n \geq 64$, $\epsilon_0 \leq \lambda = \frac{\epsilon}{6 \log n}$. Note that the algorithm possibly performs Step-4 to Step-7 multiple times, but not more than $O(\log n)$ times, as explained below.

Let $(A_1, B_1, C_1, w_1), \ldots, (A_\zeta, B_\zeta, C_\zeta, w_\zeta)$ are the set of tuples present in the data structure currently. We define $\sum_{i=1}^{\zeta} t(A_i, B_i, C_i)$ as the number of *active* triangles. Let $\textsc{Active}_i$ be the number of triangles that are active in the i^{th} iteration. Note that $\textsc{Active}_1 \leq t(G) \leq n^3$. By Lemma 6 and Step-7, observe that $\textsc{Active}_{i+1} \leq \frac{\textsc{Active}_i}{2}$. So, after $3 \log n$ many iterations there will be at most constant number of active triangles and then we can compute the exact number of active triangles and add it to ψ. In each iteration, there can be a multiplicative error of $1 \pm \lambda$ in Step-5 and $1 \pm \epsilon_0$ due to Step-4. So, using the fact that $\epsilon_0 \leq \lambda$, the multiplicative approximation factor lies between $(1 - \lambda)^{3 \log n + 1}$ and $(1 + \lambda)^{3 \log n + 1}$. As $\lambda = \frac{\epsilon}{6 \log n}$, the required approximation factor is $1 \pm \epsilon$.

The query complexity of Step 1 is $\mathcal{O}(\epsilon^{-12}d^{12}\log^{25} n)$. The query complexity of Steps 4 to 6 is $\mathcal{O}\left(\epsilon^{-4}\log^{21} n\right)$ in each iteration and the total number of iterations is $\mathcal{O}(\log n)$. Hence, the total query complexity of the algorithm is $\mathcal{O}(\epsilon^{-12}d^{12}\log^{25} n)$.

Now, we bound the failure probability of the algorithm. The algorithm can fail in Step-1 with probability at most $\frac{1}{n^{10}}$, Step-2 with probability at most $\frac{2}{n^4}$, Step-6 with probability at most $\frac{10\kappa_3 \log^{16} n}{\epsilon^4}\cdot\frac{1}{n^9}+\frac{1}{n^{10}}$, and Step-7 with probability at most $\frac{10\kappa_3 \log^{16} n}{\epsilon^4}\cdot\frac{1}{n^8}$. As the algorithm might execute Steps 4 to 6 for $3\log n$ times, the total failure probability is bounded by $\frac{1}{n^{10}}+\frac{2}{n^4}+3\log n\left(\frac{10\kappa_3 \log^{16} n}{\epsilon^4}\cdot\frac{1}{n^8}+\frac{10\kappa_3 \log^{16} n}{\epsilon^4}\cdot\frac{1}{n^9}+\frac{1}{n^{10}}\right)\leq\frac{c}{n^2}$. Note that the above inequality holds because $\epsilon > \frac{d\log^2 n}{n^{1/4}}$ and $n \geq 64$.

References

1 Nesreen K Ahmed, Nick Duffield, Jennifer Neville, and Ramana Kompella. Graph sample and hold: A framework for big-graph analytics. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1446–1455. ACM, 2014.

2 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on Principles of Database Systems*, pages 5–14. ACM, 2012.

3 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

4 Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.

5 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*, pages 623–632. Society for Industrial and Applied Mathematics, 2002.

6 Paul Beame, Sariel Har-Peled, Sivaramakrishnan Natarajan Ramamoorthy, Cyrus Rashtchian, and Makrand Sinha. Edge Estimation with Independent Set Oracles. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 38:1–38:21, 2018. `doi:10.4230/LIPIcs.ITCS.2018.38`.

7 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Triangle Estimation using Tripartite Independent Set Queries. *CoRR*, abs/1808.00691, 2018. `arXiv:1808.00691`.

8 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Hyperedge Estimation using Polylogarithmic Subset Queries. *arXiv preprint*, 2019. `arXiv:1908.04196`.

9 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, ISAAC 2018, December 16-19, 2018, Jiaoxi, Yilan, Taiwan*, pages 25:1–25:12, 2018.

10 Andreas Björklund, Rasmus Pagh, Virginia Vassilevska Williams, and Uri Zwick. Listing triangles. In *International Colloquium on Automata, Languages, and Programming*, pages 223–234. Springer, 2014.

11 Luciana S Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. Counting triangles in data streams. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 253–262. ACM, 2006.

12 Sung-Soon Choi and Jeong Han Kim. Optimal query complexity bounds for finding graphs. *Artificial Intelligence*, 174(9-10):551–569, 2010.

13 Graham Cormode and Hossein Jowhari. A second look at counting triangles in graph streams (corrected). *Theoretical Computer Science*, 683:22–30, 2017.

14 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*, pages 281–288. ACM, 2018.

15 Holger Dell, John Lapinskas, and Kitty Meeks. Approximately counting and sampling small witnesses using a colourful decision oracle. *arXiv preprint*, 2019. `arXiv:1907.04826`.

16 Devdatt P Dubhashi and Alessandro Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, 2009.

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

18 Talya Eden, Dana Ron, and C Seshadhri. On approximating the number of k-cliques in sublinear time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 722–734. ACM, 2018.

19 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.

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

21 Mira Gonen, Dana Ron, and Yuval Shavitt. Counting stars and other small subgraphs in sublinear-time. *SIAM Journal on Discrete Mathematics*, 25(3):1365–1411, 2011.

22 Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *SIAM Journal on Computing*, 7(4):413–423, 1978.

23 Svante Janson. Large deviations for sums of partly dependent random variables. *Random Structures & Algorithms*, 24(3):234–248, 2004.

24 Madhav Jha, Comandur Seshadhri, and Ali Pinar. A space efficient streaming algorithm for triangle counting using the birthday paradox. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 589–597. ACM, 2013.

25 Hossein Jowhari and Mohammad Ghodsi. New streaming algorithms for counting triangles in graphs. In *International Computing and Combinatorics Conference*, pages 710–716. Springer, 2005.

26 John Kallaugher and Eric Price. A hybrid sampling scheme for triangle counting. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1778–1797. Society for Industrial and Applied Mathematics, 2017.

27 Daniel M Kane, Kurt Mehlhorn, Thomas Sauerwald, and He Sun. Counting arbitrary subgraphs in data streams. In *International Colloquium on Automata, Languages, and Programming*, pages 598–609. Springer, 2012.

28 A. Pavan, Kanat Tangwongsan, Srikanta Tirthapura, and Kun-Lung Wu. Counting and Sampling Triangles from a Graph Stream. *PVLDB*, 6(14):1870–1881, 2013.

29 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):15, 2016.

30 Aviad Rubinstein, Tselil Schramm, and S. Matthew Weinberg. Computing Exact Minimum Cuts Without Knowing the Graph. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 39:1–39:16, 2018.

31 Larry Stockmeyer. The complexity of approximate counting. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, pages 118–126. ACM, 1983.

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

33 Kanat Tangwongsan, Aduri Pavan, and Srikanta Tirthapura. Parallel triangle counting in massive streaming graphs. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 781–786. ACM, 2013.

A Scenario where Δ_E is bounded

In this Section, we discuss some scenarios where the number of triangles sharing an edge is bounded. An obvious example for such graphs are graphs with bounded degree. We explore some other scenarios.

(i) Consider a graph $G(P, E)$ such that the vertex set P corresponds to a subset of $\mathbb{R}^2$ and $(u, v) \in E$ if and only if the distance between u and v is exactly 1. The objective is to compute the number of triples of points from P forming an equilateral triangle having side length 1, that is, the number of triangles in G. Observe that there can be at most two triangles sharing an edge in G, that is, $\Delta_E \leq 2$.

(ii) Consider a graph $G(P, E)$ such that the vertex set P corresponds to a set of points inside an $N \times N$ square in $\mathbb{R}^2$ and $(u, v) \in E$ if and only if the distance between u and v is at most 1. The objective is to compute the number of triples of points from P forming a triangle having each side length at most 1, that is, the number of triangles in G. For *large* enough N there can be bounded number of triangles sharing an edge in G with high probability.

(iii) Consider a graph $G(V, E)$ representing a community sharing information. Each node has some information and two nodes are connected if and only if there exists an edge between the nodes. Nodes increase their information by sharing information among their neighbors in G. Observe that the information of a node is derived by the set of neighbors. So, if two nodes have *large* number of common neighbors in G, then there is no need of an edge between the two nodes. So, the number of triangles on any edge in the graph is bounded. The objective is to compute the number of triangles in G, that is, the number of triples of nodes in G such that each pair of vertices are connected.

In (i) and (ii), TIS oracle can be implemented very efficiently. We can report a TIS query by just running a standard plane sweep algorithm in Computational Geometry that takes $\mathcal{O}(n \log n)$ running time.

B Some probability results

▶ **Proposition 14.** *Let X be a random variable. Then $\mathbb{E}[X] \leq \sqrt{\mathbb{E}[X^2]}$.*

▶ **Lemma 15** (Theorem 7.1 from [16]). *Let f be a function of n random variables $X_1, \ldots, X_n$ such that*

(i) *Each X_i takes values from a set A_i,*

(ii) *$\mathbb{E}[f]$ is bounded, i.e., $0 \leq \mathbb{E}[f] \leq M$,*

(iii) *$\mathcal{B}$ be any event satisfying the following for each $i \in [n]$.*

$$|\mathbb{E}[f \mid X_1, \ldots, X_{i-1}, X_i = a_i, \mathcal{B}^c] - \mathbb{E}[f \mid X_1, \ldots, X_{i-1}, X_i = a_i', \mathcal{B}^c]| \leq c_i.$$

Then for any $\delta \geq 0$,

$$\mathbb{P}\left(|f - \mathbb{E}[f]| > \delta + M\mathbb{P}(\mathcal{B})\right) \leq e^{-\delta^2 / \sum\limits_{i=1}^{n} c_i^2} + \mathbb{P}(\mathcal{B}).$$

▶ **Lemma 16** (Hoeffding's inequality [16]). *Let $X_1, \ldots, X_n$ be n independent random variables such that $X_i \in [a_i, b_i]$. Then for $X = \sum\limits_{i=1}^{n} X_i$, the following is true for any $\delta > 0$.*

$$\mathbb{P}\left(|X - \mathbb{E}[X]| \geq \delta\right) \leq 2 \cdot e^{-2\delta^2 / \sum\limits_{i=1}^{n} (b_i - a_i)^2}.$$

▶ **Lemma 17** (Chernoff-Hoeffding bound [16]). *Let $X_1, \ldots, X_n$ be independent random variables such that $X_i \in [0,1]$. For $X = \sum\limits_{i=1}^{n} X_i$ and $\mu_l \leq \mathbb{E}[X] \leq \mu_h$, the followings hold for any $\delta > 0$.*

(i) $\mathbb{P}\left(X > \mu_h + \delta\right) \leq e^{-2\delta^2/n}$.

(ii) $\mathbb{P}\left(X < \mu_l - \delta\right) \leq e^{-2\delta^2/n}$.

▶ **Lemma 18** (Theorem 3.2 from [16]). *Let $X_1, \ldots, X_n$ be random variables such that $a_i \leq X_i \leq b_i$ and $X = \sum\limits_{i=1}^{n} X_i$. Let $\mathcal{D}$ be the* dependent *graph, where $V(\mathcal{D}) = \{X_1, \ldots, X_n\}$ and $E(\mathcal{D}) = \{(X_i, X_j) : X_i$ and X_j are dependent$\}$. Then for any $\delta > 0$,*

$$\mathbb{P}(|X - \mathbb{E}[X]| \geq \delta) \leq 2e^{-2\delta^2/\chi^*(\mathcal{D}) \sum\limits_{i=1}^{n}(b_i - a_i)^2},$$

where $\chi^*(\mathcal{D})$ denotes the fractional chromatic number *of $\mathcal{D}$.*

The following lemma directly follows from Lemma 18.

▶ **Lemma 19.** *Let $X_1, \ldots, X_n$ be indicator random variables such that there are at most d many X_j's on which an X_i depends and $X = \sum\limits_{i=1}^{n} X_i$. Then for any $\delta > 0$,*

$$\mathbb{P}(|X - \mathbb{E}[X]| \geq \delta) \leq 2e^{-2\delta^2/(d+1)n}.$$

▶ **Lemma 20** (Importance sampling [6]). *Let $(D_1, w_1, e_1), \ldots, (D_r, w_r, e_r)$ are the given structures and each D_i has an associated weight $c(D_i)$ satisfying*

(i) $w_i, e_i \geq 1, \forall i \in [r]$;

(ii) $\frac{e_i}{\rho} \leq c(D_i) \leq e_i \rho$ *for some* $\rho > 0$ *and all* $i \in [r]$; *and*

(iii) $\sum\limits_{i=1}^{r} w_i \cdot c(D_i) \leq M$.

Note that the exact values $c(D_i)$'s are not known to us. Then there exists an algorithm that finds $(D'_1, w'_1, e'_1), \ldots, (D'_s, w'_s, e'_s)$ such that all of the above three conditions hold and $\left|\sum\limits_{i=1}^{t} w'_i \cdot c(D'_i) - \sum\limits_{i=1}^{r} w_i \cdot c(D_i)\right| \leq \lambda S$ with probability $1 - \delta$; where $S = \sum\limits_{i=1}^{r} w_i \cdot c(D_i)$ and $\lambda, \delta > 0$. The time complexity of the algorithm is $\mathcal{O}(r)$ and $s = \mathcal{O}\left(\frac{\rho^4 \log M \left(\log\log M + \log \frac{1}{\delta}\right)}{\lambda^2}\right)$.