How the Degeneracy Helps for Triangle Counting in Graph Streams

Suman K. Bera
sbera@ucsc.edu
UC Santa Cruz
Santa Cruz, California

C. Seshadhri
sesh@ucsc.edu
UC Santa Cruz
Santa Cruz, California

ABSTRACT

We revisit the well-studied problem of triangle count estimation in graph streams. Given a graph represented as a stream of m edges, our aim is to compute a $(1 \pm \varepsilon)$-approximation to the triangle count T, using a small space algorithm. For arbitrary order and a constant number of passes, the space complexity is known to be essentially $\Theta(\min(m^{3/2}/T, m/\sqrt{T}))$ (McGregor et al., PODS 2016, Bera et al., STACS 2017).

We give a (constant pass, arbitrary order) streaming algorithm that can circumvent this lower bound for *low degeneracy graphs*. The degeneracy, κ, is a nuanced measure of density, and the class of constant degeneracy graphs is immensely rich (containing planar graphs, minor-closed families, and preferential attachment graphs). We design a streaming algorithm with space complexity $\widetilde{O}(m\kappa/T)$. For constant degeneracy graphs, this bound is $\widetilde{O}(m/T)$, which is significantly smaller than both $m^{3/2}/T$ and $m/\sqrt{T}$. We complement our algorithmic result with a nearly matching lower bound of $\Omega(m\kappa/T)$.

CCS CONCEPTS

• **Theory of computation → Streaming, sublinear and near linear time algorithms**; Graph algorithms analysis.

KEYWORDS

Triangle counting, Streaming Model, Degeneracy

ACM Reference Format:
Suman K. Bera and C. Seshadhri. 2020. How the Degeneracy Helps for Triangle Counting in Graph Streams. In *Proceedings of the 39th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3375395.3387665

1 INTRODUCTION

Triangle counting is a fundamental algorithmic problem for graph streams. Indeed, the literature on this one problem is so rich, that its study is almost a subfield in of itself. Since the introduction of this problem by Bar-Yossef et al [9], there has been two decades of research on streaming algorithms for triangle counting [9, 11, 13, 14, 22, 34, 38, 39, 41, 45–48, 59, 60]. The significance of triangle counting is underscored by the wide variety of fields where it is studied: database theory, theoretical computer science, and data mining. From a practical standpoint, triangle counting is a core analysis task in network science. Given the scale of real-world graphs, this task is considered to be computationally intensive. In database systems, triangle counting is used for query size estimation in database join problems (see [5, 7] for details). These have led to the theoretical and practical study of triangle counting in a variety of computational models: distributed shared-memory, MapReduce, and streaming [4, 8, 18, 20, 42, 51, 52, 54, 56–58].

Despite the plethora of previous work in the streaming setting, the following question has not received much attention. *Are there "natural" graph classes that admit more efficient streaming algorithms for triangle counting?* This question has a compelling practical motivation. It is well known from network science that massive real-world graphs exhibit special properties. Could graph classes that contain such real-world graphs have "better than worst-case" streaming triangle algorithms?

Motivated by these considerations, we study the problem of streaming triangle counting, parametrized by the *graph degeneracy* (also called the maximum core number). We defer the formal definition for later, but for now, it suffices to think of degeneracy as a nuanced measure of graph sparsity. The class of constant degeneracy graphs is extremely rich: it contains all planar graphs, all minor-closed families of graphs, and preferential attachment graphs. The degeneracy of real-world graphs is well studied, under the concept of *core decompositions*. It is widely observed that the degeneracy of real-world graphs is quite small, many orders of magnitude smaller than worst-case upper bounds [24, 35, 36, 55].

In computational models other than streaming, the degeneracy is known to be relevant for triangle counting. From the perspective of running time of exact sequential algorithms, a seminal combinatorial algorithm of Chiba-Nishizeki gives an $O(m\kappa)$ time algorithm for exact triangle counting (m is the number of edges, and κ is the degeneracy) [18]. Thus, for (say) constant degeneracy, this algorithm beats the best known running time bounds of more sophisticated matrix multiplication based algorithm (of course, the latter work for all graphs) [2]. In distributed and query-based computational models, a number of results have shown that low degeneracy is helpful in bounding communication or query complexities [30, 33, 36, 56]. This inspires the main question addressed by this paper.

Do there exist streaming algorithms for approximate triangle counting on low degeneracy graphs that can beat known worst-case lower bounds?

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
PODS'20, June 14–19, 2020, Portland, OR, USA
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-7108-7/20/06…$15.00
https://doi.org/10.1145/3375395.3387665

1.1 Our results and significance

We focus on constant pass streaming algorithms, with arbitrary order. Thus, we think of the input graph $G = (V, E)$ represented as an arbitrary list of (unrepeated) edges. Our algorithm is allowed to make a constant number of passes over this list, but has limited storage. As is standard, we use n for the number of vertices, m for the number of edges, and T for the number of triangles in G.

We first define the graph degeneracy.

Definition 1.1. The degeneracy of a graph G, denoted $\kappa(G)$, is defined as $\max_{G' \text{ subgraph of } G}\{\text{min degree of } G'\}$. In words, it is the largest possible minimum degree of a subgraph of G.

In a low degeneracy graph, all induced subgraphs have low degree vertices. The following procedure that computes the degeneracy is helpful for intuition. Suppose one iteratively removed the minimum degree vertex from G (updating degrees after every removal). For any vertex v, consider the "observed" degree *at the time of removal*. One can prove that the degeneracy is the largest such degree [1]. Thus, even though G could have a large maximum degree, the degeneracy can be small if high degree vertices are typically connected to low degree vertices.

Our main theorem follows.

Theorem 1.2. *Consider a graph G of degeneracy at most κ, that is input as an arbitrary edge stream. There is a streaming algorithm that outputs a $(1 \pm \varepsilon)$-approximation to T, with high probability*[1]*, and has the following properties. It makes constant number of passes over the input stream and uses space $(m\kappa/T) \cdot \text{poly}(\log n, \varepsilon^{-1})$.*

To understand the significance of the bound $m\kappa/T$, note that space complexity of streaming triangle counting is known to be $\min(m^{3/2}/T, m/\sqrt{T})$ [11, 46]. Consider $\kappa = O(1)$, which as mentioned earlier, holds for all graphs in minor-closed families and preferential attachment graphs. In this case, the algorithm of Theorem 1.2 uses space $\widetilde{O}(m/T)$. This is significantly smaller than both $m^{3/2}/T$ and $m/\sqrt{T}$. We note that for all graphs, $\kappa \leq \sqrt{2m}$, and thus, the space is always $\widetilde{O}(m^{3/2}/T)$.

As an illustrative example, consider the wheel graph with n vertices (take a cycle with $n-1$ vertices, and add a central vertex connected to all other vertices). Note that $m = T = \Theta(n)$ and $\kappa = O(1)$ (G is planar). The space bound given in Theorem 1.2 is only polylogarithmic, while *all existing* streaming algorithms bounds (given in Table 1) are $\Omega(\sqrt{n})$.

Our bound of $\widetilde{O}(m\kappa/T)$ subsumes the term $\widetilde{O}(m^{3/2}/T)$, and dominates the term $\widetilde{O}(m/\sqrt{T})$ when $T = \Omega(\kappa^2)$. For real-world graphs, $T = \Omega(\kappa^2)$ is a naturally occurring phenomenon. In fact, real-world large graphs are often characterized by following two properties: (1) low sparsity, and (2) high triangle density [25, 50, 53, 61]. Thus, from a practical standpoint, our bound offers significant improvement over previously known bounds.

We complement Theorem 1.2 with a nearly matching lower bound.

Theorem 1.3. *Any constant pass randomized streaming algorithm for graphs with m edges, T triangles, and degeneracy at most κ, that provides a constant factor approximation to T with probability at least $2/3$, requires storage $\Omega(m\kappa/T)$.*

[1]We use "high probability" to denote errors less than $1/3$.

We remark that all our results in this paper can be equivalently stated in terms of *arboricity* as well. The *arboricity* of a graph G, denoted as α, is the smallest integer p such that the edge set $E(G)$ can be partitioned into p forests. It is asymptotically same as *degeneracy*: for every graph $\alpha \leq \kappa \leq 2\alpha - 1$.

1.2 Main ideas

We give a high-level description of our algorithm and proof. The final algorithm has a number of moving parts, and is based on recent advances in sublinear algorithms for clique counting [26, 29, 30].

The starting point for our algorithm (and indeed, most triangle counting results related to degeneracy) is the classic sequential procedure of Chiba-Nishizeki. For every edge $e = (u, v)$, the size of intersection on neighborhoods of u and v is the number of triangles containing e. This intersection can be determined easily in $\min(d_u, d_v)$ operations, by searching for elements of the smaller neighborhood in the larger one. (Here, d_u denotes the degree of vertex u.) For convenience, let us define the degree of edge e to be $d_e := \min(d_u, d_v)$. Thus, we can enumerate all triangles in $\sum_e d_e$ time. The classic bound of Chiba-Nishzeki asserts that $\sum_e d_e = O(m\kappa)$.

As a warmup, let us get an $O(m\kappa/T)$ space streaming algorithm, that uses a degree oracle. Define $d_E := \sum_e d_e = O(m\kappa)$. With the degree oracle, in a single pass, we can sample an edge e proportional to its degree. In the second pass, pick a uniform random neighbor w of the lower degree endpoint of e. In the third pass, determine if e and w form a triangle. The probability of finding a triangle is exactly $3T/d_E$. By sampling $O(d_E/T) = O(m\kappa/T)$ independent random edges in the first pass, we can estimate T with $O(m\kappa/T)$ space.

The main challenge is in removing the degree oracle. As a first step, can we effectively simulate sampling edges proportional to their degree? We borrow a key idea from recent sublinear algorithms for clique counting. First, we sample a set of uniform random edges, denoted R. In a second pass, we compute the degree of all edges in R. Now, we can run the algorithm described earlier, except we only sample edges of R. Observe that in the latter sample, taking expectations over R, we do sample edges proportional to their degree from the overall graph. Unfortunately, these samples are all correlated by the choice of R. How large should R be to ensure that this simulation leads to the right answer?

An alternate viewpoint is to observe that the above approach can give an accurate estimate to the number of triangles incident to R, denoted t_R. We require R to be large enough, so that t_R can be used to estimate T. Let t_e be the number of triangles incident to e. The $\{t_e\}$ values can exhibit large variance, even when $\kappa = O(1)$. Consider a graph formed by $(n-2)$ triangles that all share a common edge. The graph is planar, so $\kappa = O(1)$. But one edge is incident to $(n-2)$ triangles, and all other edges are incident to a single triangle. Thus, the $\{t_e\}$ values have the largest possible variance, and one cannot estimate T by computing $\sum_{e \in R} t_e$ for a small R. Note that, for this example graph, our desired streaming algorithm uses only polylogarithmic space ($T = \Theta(m)$, $\kappa = O(1)$).

Another key idea from sublinear clique counting saves the day: assignment rules. The idea is to assign triangles uniquely to edges, so that the distribution of assigned triangles has low variance. The

overall algorithm will estimate the number of triangles *assigned* to R, and not count the number of triangles *incident* to R. A natural, though seemingly circular, rule is to assign each triangle to the contained edge that itself participates in the fewest triangles. Using properties of graph degeneracy, it is shown in [30] that the maximum number of assigned triangles to any edge is $O(\kappa)$. (Technically, this is not true. We have to leave some triangles unassigned.)

This leads to another technical complication. In the overall algorithm, when a triangle incident to an edge is discovered, the algorithm needs to determine if the triangle is actually assigned to the edge. This requires estimating t_e for all edges e in the triangle, a potentially space intensive operation. To perform this estimation in $O(m\kappa/T)$ requires subtle modifications to the assignment procedure. It turns out we can ignore triangles containing edges of high degree, and thus, the above t_e estimation is only required for low degree edges. Furthermore, we only need to determine if $t_e = \Omega(\kappa)$, which allows for smaller storage algorithms.

All in all, by choosing parameters carefully, all steps can be implemented using $(m\kappa/T)\text{poly}(\log n, \varepsilon^{-1})$ storage.

2 RELATED WORK

The triangle counting problem, a special case of more general subgraph counting problem, has been studied extensively in the streaming setting. We present a summary of the significant prior works in Table 1. The upper bounds stated in the table are for randomized streaming algorithms that provide $(1 \pm \varepsilon)$-approximation to the true triangle count with probability at least 2/3. The $\widetilde{O}$ notion hides polynomial dependencies on $1/\varepsilon$ and $\log n$. The lower bounds are primarily based on the triangle detection problem — detect whether the input graph is triangle free or it contains at least T many triangles. All the results presented in the table are for the arbitrary order stream.

Jha et al. [37] designed a one pass $\widetilde{O}(m/\sqrt{T})$-space algorithm with $\pm W$-additive error approximation, where W is the number of two length paths (also called wedges). Bravermanet al. [13] gave a two-pass $\widetilde{O}(m/T^{1/3})$-space algorithm to detect if the input graph is triangle free or it has at least T many triangles. These results are not directly comparable to our work.

The triangle counting problem has been studied in the context of the adjacency list streaming model as well. This model is also known as the vertex arrival model: all the edges incident on a vertex arrive together. McGregor et al. [46] gave one-pass $\widetilde{O}(m/\sqrt{T})$-space and two pass $\widetilde{O}(m^{3/2}/T)$-space algorithm for the triangle counting problem in this model. We refer to [46] for other related work in this model.

Bounded degeneracy graph family is an important class of graphs from a practical point of view. Many real-world large graphs, specially from the domain of social networks and web graphs, often exhibit low degeneracy([12, 24, 35, 36, 55], also Table 2 in [12]). Naturally, designing algorithms that are parameterized by *degeneracy* has been a theme of many works in the streaming settings; some examples include matching size estimation [6, 23, 32], independent set size approximation [21], graph coloring [12]. In the general RAM model, the relation between degeneracy and subgraph counting problems has been explored in [10, 19, 31].

In the graph query model, where the goal is to design sub-linear time algorithms, Eden et al. [28] studied the triangle counting problem, and more generally the clique counting problem in *bounded degeneracy* graphs. Although the model is significantly different from the streaming model, we port some key ideas from there; see Section 1.2 for a detailed discussion. The relevance of *bounded degeneracy* has been further explored in the context of estimating degree moments [27] in this model.

3 NOTATIONS AND PRELIMINARIES

For an integer k, we denote the set $\{1, 2, \ldots, k\}$ by $[k]$. Throughout the paper, we denote the input graph as $G = (V, E)$. We assume G has n vertices, m edges and T many triangles. We denote the degree of a vertex $v \in V$ by d_v and its neighborhood by $N(v)$. For an edge $e = \{u, v\}$, we define its neighborhood $N(e)$ to be that of the lower degree end point: $N(e) = N(u)$ if $d_u < d_v$; $N(e) = N(v)$ otherwise. Similarly, we define the degree of an edge: $d_e = \min\{d_u, d_v\}$. For a collection of edges R, we define $d_R = \sum_{e \in R} d_e$. In particular, $d_E := \sum_{e \in E} d_e$.

Chiba and Nishizeki [19] proved the following insightful connection between the sum of degrees of the edges in a graph d_E and its *degeneracy* κ. [2]

Lemma 3.1 (Lemma 2 in [19]). *For a graph G with m edges and* degeneracy κ,

$$d_E = \sum_{e \in E} d_e \leq 2m\kappa\,.$$

As a corollary, we get the following result.

Corollary 3.2 ([19]). *For a graph G with m edges and* degeneracy κ, *the maximum number of triangles in G is at most $2m\kappa$.*

We use the notation $\widetilde{O}(\cdot)$ to hide polynomial dependencies on $(1/\varepsilon)$ and $\log n$ terms, where ε is the error parameter. For designing our algorithms, we focus on the expected space usage. This can be easily converted into a worst-case guarantee by applying Markov inequality — simply abort if the space usage runs beyond c times the expected space usage, for some constant c. This only increases the error probability by an additive $1/c$ amount.

We use the following variants of the Chernoff bound and Chebyshev inequality for analyzing our algorithms.

Theorem 3.3 (Chernoff Bound [17]). *Let $X_1, X_2, \ldots, X_r$ be mutually independent indicator random variables with expectation μ. Then, for every ε with $0 < \varepsilon < 1$, we have*

$$\Pr\left[\left|\frac{1}{r}\sum_{i=1}^{r} X_i - \mu\right| \geq \varepsilon\mu\right] \leq 2\exp\left(-\varepsilon^2 r\mu/3\right)$$

Theorem 3.4 (Chebyshev Inequality [3]). *Let X be a random variable with expectation μ and variance* $\mathbf{Var}[X]$. *Then, for every $\varepsilon > 0$,*

$$\Pr\left[|X - \mu| \geq \varepsilon\mu\right] \leq \frac{\mathbf{Var}[X]}{\varepsilon^2\mu^2}\,.$$

[2] Note that Chiba and Nishizeki [19] stated their results in terms of *arboricity*. As $\alpha \leq \kappa$ for each graph G with *arboricity* α, the same result holds with respect to *degeneracy* κ as well.

Space	Remarks	Source
$\widetilde{O}(mn/T)^2$	one pass	[9]
$\widetilde{O}(m\Delta^2/T)$	one pass, Δ = maximum degree	[38]
$\widetilde{O}(mn/T)$	one pass, n known a priori	[14]
$\widetilde{O}(m^3/T^2)$	one pass, dynamic stream	[41]
$\widetilde{O}(m\Delta/T)$	one pass, Δ = maximum degree	[48]
$\widetilde{O}(mJ/T + m/\sqrt{T})$	one pass, J = maximum triangles incident on a edge	[47]
$C + \widetilde{O}(P_2/T)$	one pass, C = vertex cover, P_2 = # of 2-paths	[34]
$\widetilde{O}(m/\sqrt{T})$	dependence on ε is $1/\varepsilon^{2.5}$	[22]
$\widetilde{O}(m^{3/2}/T)$	multi-pass	[11, 46]
$\widetilde{O}(m/\sqrt{T})$	multi-pass	[46]
$\Omega(n^2)$	one pass, $T = 1$	[9]
$\Omega(n/T)$	multi-pass, $T < n$	[38]
$\Omega(m)$	one pass, $m \in [c_1 n, c_2 n^2],\ T < n$	[13]
$\Omega(m/T)$	multi-pass	[13]
$\Omega(m^3/T^2)$	one pass, optimal	[44]
$\Omega(m/T^{2/3})$	multi-pass	[22]
$\Omega(m/\sqrt{T})$	multi-pass, for $m = \Theta(n\sqrt{T})$	[22]
$\Omega(\min\{m/\sqrt{T}, m^{3/2}/T\})$	multi-pass	[11]

Table 1: Prior work on the triangle counting problem

4 WARM-UP: AN ABSTRACT MODEL

In this section, we consider a streaming model equipped with a degree oracle: queried with a vertex v, the oracle returns d_v. Furthermore, we make a rather strong assumption: there is no cost associated with the queries. McGregor et al. [46] designed a $\widetilde{O}(m^{3/2}/T)$ space 3-pass streaming algorithm in this model — their algorithm makes $O(m)$ many degree queries. We describe an $\widetilde{O}(m\kappa/T)$-space 3 pass algorithm in this model. Our estimator makes $2m$ many degree queries and requires 3-pass. For *bounded degeneracy* graph families, this translates to a space reduction by a factor of $O(\sqrt{m})$. In the next section, we show how to design a $\widetilde{O}(m\kappa/T)$-space constant pass algorithm in the traditional streaming model.

Our main idea is to sample edges from the stream with probability proportional to its degree. In general streaming settings, this is not possible as we do not know the degree of the edges apriori. However, the model that we consider here is tailor-made for this purpose. It shows the effectiveness of degree-biased edge samples in estimating triangle count and provides motivation for taking up a similar sampling approach in the general streaming model.

We present our basic estimator in algorithm 1. In the full algorithm, we will run multiple instances of this estimator in parallel and report the "median of the mean" [15] as our final estimate.

Implementation Details. The degree proportional sampling is achieved by using weighted reservoir sampling [16]. On arrival of the edge $e = \{u, v\}$ in the stream, we make two degree queries to find d_e. The method IsAssigned is required to ensures that every triangle is uniquely associated with one of its three edges. Other than this, there is no constraints on the implementation of this method. For example, we can associate every triangle to the edge with lowest degree, breaking ties arbitrarily (but consistently). Let

Algorithm 1 A Triangle Estimator

procedure IdealEstimator(Graph $G = (V, E)$)
  Pass 1: Sample an edge e with probability d_e/d_E.
  Pass 2: Sample a vertex w from $N(e)$ u.a.r.
  Pass 3: Check if $\{e, w\}$ forms a triangle.
  if $\tau = \{e, w\}$ is a triangle **then**
    Call IsAssigned(τ, e).
    If returned YES, then set $Y = 1$; else set $Y = 0$.
  else
    Set $Y = 0$.
  Set $X = d_E \cdot Y$.
  return X.

t_e denote the number of triangles assigned to the edge e. Clearly, $\sum_{e \in E} t_e = T$.

Analysis. First, we show the estimator is unbiased.

$$\begin{aligned}\mathbb{E}[X] &= \sum_{e \in E} \frac{d_e}{d_E} \cdot \mathbb{E}[X|e] \\ &= \sum_{e \in E} d_e \cdot \mathbb{E}[Y|e] \\ &= \sum_{e \in E} d_e \cdot \frac{t_e}{d_e} = \sum_{e \in E} t_e = T\end{aligned}$$

Now we bound the variance of the estimator.

$$\begin{aligned}\mathbf{Var}[X] \le \mathbb{E}[X^2] &= \sum_{e\in E} \frac{d_e}{d_E}\cdot \mathbb{E}[X^2|e] \\ &= \sum_{e\in E} d_e \cdot d_E \cdot \mathbb{E}[Y^2|e] \\ &= \sum_{e\in E} d_e \cdot d_E \cdot \frac{t_e}{d_e} \\ &= d_E \cdot \sum_{e\in E} t_e = d_E \cdot T\end{aligned}$$

So, running $\widetilde{O}(\mathbf{Var}[X]/\mathbb{E}[X]^2) = \widetilde{O}(d_E/T) = \widetilde{O}(m\kappa/T)$-many estimators independently in parallel suffices for a $(1\pm\varepsilon)$-approximate estimate. Since each copy of the estimator requires constant space, the overall space usage is bounded by $\widetilde{O}(m\kappa/T)$.

5 OUR MAIN ALGORITHM

In this section, we present our streaming triangle estimator. As promised, our algorithm does not assume access to a degree oracle. If the model is indeed equipped with a degree oracle, then we can save a few passes over the stream. Perhaps more importantly, the number of queries to the oracle is upper bounded by the space usage of our algorithm. Our main algorithmic result is the following.

Theorem 5.1. *Consider a graph G of degeneracy at most κ, that is input as an arbitrary edge stream. There is a streaming algorithm that outputs a $(1\pm\varepsilon)$-approximation to T, with high probability*[3]*, and has the following properties. It makes six passes over the input stream and uses space $(m\kappa/T)\cdot \mathrm{poly}(\log n, \varepsilon^{-1})$.*

We describe our estimator in Algorithm 2. We set the parameters r and ℓ later in the analysis. In analyzing our algorithm, the procedure IsAssigned would play a crucial role. As discussed earlier, IsAssigned takes as input a triangle and an edge, and outputs whether the triangle is assigned to that edge. We want this procedure to possess four properties: (1) For a given triangle, it is either unassigned or assigned uniquely to one of its three participating edges. (2) *almost* all the triangles are assigned, (3) For any fixed edge, not too many triangles are assigned to it, and (4) The space complexity of the procedure is bounded by $\widetilde{O}(m\kappa/T)$. The first two properties are required to ensure the overall accuracy of the estimator. The third property would be central to bounding the variance of the estimator. The final property will ensure the overall space complexity of our triangle estimator is bounded by $\widetilde{O}(m\kappa/T)$.

We analyze our triangle estimator assuming a black-box access to a IsAssigned procedure that satisfies the above four properties. In the next section, we will take up the task of designing such an assignment procedure in the streaming setting. We make the above discussion rigorous and formal below.

Assume τ_e denotes the number of triangles assigned to the edge e by the procedure IsAssigned. We use $\tau_{\max}$ to denote the maximum number of triangles that any edge has been assigned to: $\tau_{\max} = \max_{e\in E}\tau_e$. Denote the number of triangles that IsAssigned assigns to some edge by $\mathcal{T} = \sum_{e\in E}\tau_e$. Then, for any positive constant ε and δ, we define an (ε,δ)-accurate IsAssigned procedure below.

[3]We use "high probability" to denote errors less than 1/3.

Definition 5.2 ((ε,δ)-accurate IsAssigned)**.** A procedure IsAssigned that assigns a triangle to an edge or leaves it unassigned, is (ε,δ)-accurate if it satisfies the following four properties.

(1) *Unique Assignment:* For each triangle, it is either unassigned or uniquely assigned to one of the three participating edges. This implies, $\mathcal{T} \le T$.
(2) *Almost All Assignment:* With probability at least $1-\delta$, $\mathcal{T} \ge (1-12\varepsilon)T$.
(3) *Bounded Assignment:* With probability at least $1-\delta$, $\tau_{\max} \le \kappa/\varepsilon$.
(4) *Bounded Space Complexity:* Each call requires $\widetilde{O}(m\kappa/T)$ bits of space.

We now analyze our algorithm assuming a black-box access to $(\varepsilon, O(1/n^5))$-accurate IsAssigned. The analysis consists of two parts. First, we show that for a certain settings of r and ℓ, our final estimate X is indeed a $(1\pm\varepsilon)$ approximation to the true triangle count. In the sequel, we bound the space complexity of our algorithm.

Algorithm 2 Estimation of triangle count

```
1:  procedure EstimateTraingle(Graph G = (V, E))
2:      Pass 1: Sample r many edges u.a.r: R = {e_i}_{i=1}^r.
3:      Pass 2: Compute d_e for each e ∈ R.
4:      for i = 1 to ℓ do
5:          Sample an edge e ∈ R independently with prob. d_e/d_R.
6:          Pass 3: Sample a vertex w from N(e) u.a.r.
7:          Pass 4: Check if {e, w} forms a triangle.
8:          if τ = {e, w} is a triangle then
9:              Call IsAssigned(τ, e).
10:             If returned YES, then set Y_i = 1; else set Y_i = 0.
11:         else
12:             Set Y_i = 0.
13:     Set Y = (1/ℓ) Σ_{i=1}^ℓ Y_i, and X = (m/r) · d_R · Y.
14:     return X.
```

We begin with analyzing the *quality* of the (multi)set of uniform random edges R. Collectively through Lemmas 5.3 and 5.5 and definition 5.4, we establish that for a suitable choice of the parameter r, the (multi)set R possesses desirable properties with high probability.

Lemma 5.3. *Let $d_R = \sum_{e\in R} d_e$ and $\tau_R = \sum_{e\in R}\tau_e$. For any constant $\varepsilon > 0$ we have*

(1) $\mathbf{E}[d_R] = r\cdot\frac{d_E}{m}$ *and* $\mathbf{E}[\tau_R] = r\cdot\frac{\mathcal{T}}{m}$,
(2) $\Pr\left[d_R \le \mathbf{E}[d_R]\cdot\frac{\log n}{\varepsilon}\right] \ge 1-\frac{\varepsilon}{\log n}$,
(3) $\Pr\left[|\tau_R - \mathbf{E}[t_R]| \le \varepsilon\mathbf{E}[\tau_R]\right] \ge 1-\frac{1}{\varepsilon^2}\cdot\frac{1}{r}\cdot\frac{m\cdot\tau_{\max}}{\mathcal{T}}$.

Proof. We first compute the expected value of d_R and τ_R. We define two sets of random variables, Y_i^d and Y_i^t for $i\in[r]$ as follows: $Y_i^d = d_{e_i}$, and $Y_i^t = \tau_{e_i}$. Then, $d_R = \sum_{i=1}^r Y_i^d$ and $\tau_R = \sum_{i=1}^r Y_i^t$.

We have

$$\begin{aligned}\mathbf{E}\left[Y_i^d\right] &= \sum_{e\in E}\Pr[e_i = e]\cdot\mathbf{E}\left[Y_i^d|e_i = e\right]\\ &= \frac{1}{m}\sum_{e\in E} d_e\\ &= \frac{d_E}{m}\,.\end{aligned}$$

Then, by linearity of expectation, we get $\mathbf{E}[d_R] = r\cdot d_E/m$. Analogously, we have $\mathbf{E}[\tau_R] = r\cdot\mathcal{T}/m$.

We now turn our focus on the concentration of d_R. This is achieved by a simple application of Markov inequality.

$$\Pr\left[d_R \geq \mathbf{E}\left[d_R\right]\cdot\frac{\log n}{\varepsilon}\right] \leq \frac{\varepsilon}{\log n}\,.$$

To prove a concentration bound on τ_R, we study the variance of τ_R. By independence, we have

$$\begin{aligned}\mathbf{Var}[\tau_R] &= \sum_{i=1}^{r}\mathbf{Var}[Y_i^t]\\ &\leq \sum_{i=1}^{r}\mathbf{E}[(Y_i^t)^2]\\ &= \sum_{i=1}^{r}\sum_{e\in E}\Pr[e_i = e]\,\mathbf{E}[(Y_i^t)^2|e_i = e]\\ &= \frac{r\cdot\sum_{e\in E}\tau_e^2}{m}\\ &\leq \frac{r\cdot\tau_{\max}\sum_{e\in E}\tau_e}{m}\\ &= \frac{r\cdot\tau_{\max}\mathcal{T}}{m}\,.\end{aligned}$$

Then, the item (3) of the lemma follows by an application of Chebyshev inequality (Theorem 3.4). □

We next define a collection of edges R as *good* if the conditions in items (2) and (3) in Lemma 5.3 are satisfied. Formally, we have the following definition.

Definition 5.4 (A *good* collection of edges)**.** We call a fixed collection of edges R, ε-good if the following two conditions are true.

$$d_R \leq \frac{\log n}{\varepsilon}\cdot|R|\cdot\frac{d_E}{m} \tag{1}$$

$$\tau_R \in \left[(1-\varepsilon)\cdot|R|\cdot\frac{\mathcal{T}}{m}\,,\ (1+\varepsilon)\cdot|R|\cdot\frac{\mathcal{T}}{m}\right] \tag{2}$$

Lemma 5.5 (Setting of r for an ε-good R)**.** *Let $0 < \varepsilon < 1/6$ and $c > 6$ be some constants, and $r = \frac{c\log n}{\varepsilon^2}\frac{m\tau_{\max}}{\mathcal{T}}$. Then, with probability at least $1 - \frac{1}{6\log n}$, R is ε-good.*

Proof. The lemma follows by plugging in $r = \frac{c\log n}{\varepsilon^2}\frac{m\tau_{\max}}{\mathcal{T}}$ in Lemma 5.3 and using the bounds on c and ε. □

We have established that the random collection of edges R is *good* with high probability. We now turn our attention to the random variable Y, as defined on Line 13 Algorithm 2. Together in Lemmas 5.6 and 5.7 we show that, if R is *good* then for a suitably chosen parameter ℓ, the random variable Y is well-concentrated around its mean.

Lemma 5.6. *Let R be a fixed collection of edges, and Y_R denote the value of the random variable Y as defined on Line 13 Algorithm 2 on R. Then,*

(1) $\mathbf{E}[Y_R] = \frac{\tau_R}{d_R}$,

(2) $\Pr[\ |Y_R - \mathbf{E}[Y_R]| \geq \varepsilon\mathbf{E}[Y_R]\ ] \leq \exp\left(-\ell\cdot\frac{\varepsilon^2}{3}\cdot\frac{\tau_R}{d_R}\right)$.

Proof. Let e_i be the edge sampled in the i-th iteration of the for loop at line 4 in Algorithm 2. Then,

$$\begin{aligned}\mathbb{E}[Y_i = 1] &= \sum_{e\in R}\Pr[e_i = e]\Pr[Y_i = 1|e_i = e]\,,\\ &= \sum_{e\in R}\frac{d_e}{d_R}\Pr[Y_i = 1|e_i = e]\,,\\ &= \sum_{e\in R}\frac{d_e}{d_R}\cdot\frac{\tau_e}{d_e}\\ &= \sum_{e\in R}\frac{\tau_e}{d_R}\\ &= \frac{\tau_R}{d_R}\,.\end{aligned}$$

By linearity of expectation, we have the item (1) of the lemma. For the second item, we apply Chernoff bound(Theorem 3.3). □

Lemma 5.7 (Setting of ℓ for concentration of Y_R)**.** *Let $0 < \varepsilon < 1/6$ and $c > 20$ be some constants, and $\ell = \frac{c\log n}{\varepsilon^2}\cdot\frac{m\cdot d_R}{r\cdot\mathcal{T}}$. Then, with probability at least $1 - \frac{1}{5\log n}$, $|Y_R - \mathbf{E}[Y_R]| \leq \varepsilon\mathbf{E}[Y_R]$.*

Proof. By Lemma 5.5, R is ε-good with probability at least $1 - \frac{1}{6\log n}$. Condition on the event that R is ε-good. By definition of a ε-good set, τ_R is tightly concentrated around its mean: $\tau_R \in [(1-\varepsilon)r\mathcal{T}/m\,,\ (1+\varepsilon)r\mathcal{T}/m]$ Then, by item 2. in lemma 5.6, we have

$$\begin{aligned}&\Pr[\ |Y_R - \mathbf{E}[Y_R]| \geq \varepsilon\mathbf{E}[Y_R]\ ]\\ &\leq \exp\left(-\frac{c\log n}{\varepsilon^2}\cdot\frac{md_R}{r\mathcal{T}}\cdot\frac{\varepsilon^2}{3}\cdot\frac{\tau_R}{d_R}\right)\\ &\leq \exp\left(-\frac{c\log n}{3}\cdot\tau_R\cdot\frac{m}{r\mathcal{T}}\right)\\ &= o(1/n^3)\,,\end{aligned}$$

where the last line follows from the concentration of τ_R for ε-good R. Removing the condition on R, we derive the lemma. □

We have now all the ingredients to prove that our final estimate is indeed close to the actual triangle count. The random variable Y is scaled appropriately to ensure that its expectation is close to the true triangle count.

Lemma 5.8. *Assume r and ℓ is set as in Lemma 5.5 and Lemma 5.7 respectively. Then, there exists a small constant ε' such that with probability at least $1 - \frac{1}{3\log n}$, $X \in [(1-\varepsilon')T), (1+\varepsilon')T)]$.*

Proof. With probability at least $1 - \frac{1}{5\log n}$, Y_R is closely concentrated around its expected value τ_R/d_R (by Lemma 5.7). More

formally,

$$Y_R \in \left[(1-\varepsilon)\frac{\tau_R}{d_R}, (1+\varepsilon)\frac{\tau_R}{d_R}\right].$$

Then, with high probability

$$X_R \in \left[(1-\varepsilon)\cdot\frac{m}{r}\cdot\tau_R, (1+\varepsilon)\cdot\frac{m}{r}\cdot\tau_R\right].$$

Since R is good, with probability at least $1-\frac{1}{c\log n}$,

$$t_R \in \left[(1-\varepsilon)r\cdot\frac{\mathcal{T}}{m}, (1+\varepsilon)r\cdot\frac{\mathcal{T}}{m},\right]$$

Then, with probability at least $1-\frac{2}{c\log n}$,

$$X_R \in [(1-2\varepsilon)\mathcal{T}, (1+2\varepsilon)\mathcal{T}]$$

Removing the conditioning on R, and using the bound on $\mathcal{T}$ as given in Definition 5.2, we have

$$X \in \left[(1-\varepsilon')T, (1+\varepsilon')T\right]$$

with probability at least $1-\frac{4}{c\log n}$, for suitable chosen parameter ε'. □

This completes the first part of the analysis. We now focus on the space complexity of Algorithm 2.

Lemma 5.9 (Space Complexity of Algorithm 2)**.** *Assuming an access to a (ε,δ)-accurate IsAssigned method, Algorithm 2 requires $\widetilde{O}(m\kappa/T)$ bits of storage in expectation.*

Proof. Clearly, $O(r+\ell)$ space is sufficient to store the set R and sample ℓ many edges from it at Line 4. Recall from Lemmas 5.5 and 5.7 that $r=\widetilde{O}(m\tau_{\max}/\mathcal{T})$ and $\ell=\widetilde{O}(md_R/(r\mathcal{T}))$, respectively. Using the bound on $\mathcal{T}$ and $\tau_{\max}$ from the definition of (ε,δ)-accurate IsAssigned method, we derive that $O(r+\ell)$ is $\widetilde{O}(m\kappa/T)$ with high probability.

We now account for the space complexity of the IsAssigned method. It is called if the the edge-vertex pair $\{e,w\}$ forms a triangle (if condition at Line 8). Let Z_i be an indicator random variable to denote if IsAssigned is called during the i-th iteration of the for loop at Line 4. Then,

$$\Pr[Z_i=1|R]=\sum_{e\in R}\frac{d_e}{d_R}\cdot\frac{t_e}{d_e}=\frac{t_R}{d_R}.$$

Then, the expected number of calls to IsAssigned is bounded by $\ell\cdot\frac{t_R}{d_R}=\widetilde{O}\left(\frac{m\,t_R}{r\,\mathcal{T}}\right)$. Note that t_R can be much different from τ_R, as it counts the exact number of triangles per edge. However, we show that with constant probability, t_R is at most $O(r\mathcal{T}/m)$, which bounds the expected number of calls by $\widetilde{O}(1)$. Since each call to IsAssigned takes $\widetilde{O}(m\kappa/T)$ bits of space, the lemma follows. We now bound t_R.

$$\mathbb{E}[t_R]=r\sum_{e\in E}\frac{t_e}{m}=\frac{3rT}{m}$$

, where the last equality follows from the fact that $\sum_{e\in E}t_e=3T$. An application of Markov inequality bounds the probability that t_R is more than $\frac{cr\mathcal{T}}{m}$ by a small constant probability, for any large constant $c>10$. □

Thus, assuming an access to a $(\varepsilon,o(1/n^5))$-accurate IsAssigned method, Lemmas 5.8 and 5.9 together prove our main result in Theorem 5.1.

5.1 Assigning triangles to edges

In this section we give an algorithm for the IsAssigned procedure in Algorithm 3. Recall from the previous section that we require IsAssigned to be (ε,δ)-accurate (see Definition 5.2).

The broad idea is to assign a triangle to the edge with smallest t_e. Recall that t_e is the number of triangles that the edge e participates in. However, computing t_e might be too expensive in terms of space required for certain edges. As evident from the analysis of Algorithm 2, we have a budget of $\widetilde{O}(m\kappa/T)$ in terms of bits of storage for each call to IsAssigned. In this regard, we define "heavy" and "costly" edges and it naturally leads to a notion of "heavy" and "costly" triangles. If a triangle is either "heavy" or "costly", then we do not attempt to assign it to any of its edges. Crucially, we show that the total number of "heavy" and "costly" triangles are only a tiny fraction of the total number of triangles in the graph.

We need to ensure that for any edge e, not too many triangles are assigned to e by IsAssigned. We achieve this by simply disregarding any edge with large t_e from consideration while assigning a triangle to an edge. Formally we capture this by defining *heavy* edges and triangles.

Definition 5.10 (*ε-heavy edge* and *ε-heavy triangle*)**.** An edge is defined *ε-heavy* if $t_e>\kappa/\varepsilon$. A triangle is deemed *ε-heavy* if *all* the three of its edges are ε-heavy.

If the ratio t_e/d_e is quite small for an edge e, then we need too many samples from the neighborhood $N(e)$ to estimate t_e. Roughly speaking, $O(d_e/t_e)$ many samples are required for an *accurate* estimation. In this regard, we define *costly edges* and *costly triangles* as follows.

Definition 5.11 (*ε-costly edge* and *ε-costly triangle*)**.** An edge e is defined *ε-costly* if $d_e/t_e>m\kappa/(\varepsilon T)$. A triangle is deemed *$\varepsilon$-costly* if *any* of its three edges is *ε-costly*.

We first show that the number *heavy triangles* and *costly triangles* are only a small fraction of the all triangles. Formally, we prove the following lemma.

Lemma 5.12. *The number of* ε-heavy *triangles and* ε-costly *triangles are bounded by $2\varepsilon T$ and εT respectively.*

Proof. We begin the proof by first showing that the number of costly triangles is bounded. To prove this, observe that for a *costly edge* e, $t_e<d_e\cdot(\varepsilon T/m\kappa)$. Then,

$$\sum_{e\text{ is costly}}t_e<\frac{\varepsilon T}{m\kappa}\sum_{e\text{ is costly}}d_e<\frac{\varepsilon T}{m\kappa}\cdot d_E=2\varepsilon T$$

, where the last inequality follows from lemma 3.1.

We now turn our attention to bounding the number of *heavy triangles*. By a simple counting argument, the number of *heavy edges* in G is at most $\varepsilon T/\kappa$. Consider the subgraph of G induced by the set of *heavy edges*, denoted as G_{heavy}. It follows from the definition of degeneracy that $\kappa_{G_{\text{heavy}}}\le\kappa_G$. By Corollary 3.2, the number of triangles in G_{heavy} is then at most $\kappa_{G_{\text{heavy}}}\cdot E(G_{\text{heavy}})=\varepsilon T$. Since any *heavy triangle* in G is present in G_{heavy}, the lemma follows. □

We give the details of the procedure in Algorithm 3. The technical part of this method is handled by Assignment subroutine at Line 7. Given a triangle τ, Assignment either returns $\perp$ (τ is not assigned to any edges) or returns an edge e. We remark here that the method Assignment as described is randomized and may return different e on different invocations. To ensure that every triangle is assigned to an unique edge, as demanded in the item (1) of Definition 5.2, we maintain a table of (key,value) pairs that maps triangles (key) to edges or $\perp$ symbol(value). In particular, when Assignment is invoked with input τ, we first look up in the table to check if there is an entry for the triangle τ. If it is there, then we simply return the corresponding value from the table. Otherwise, we execute Assignment with input τ; create an entry for τ and and store the return value together with τ in the table. Since the expected number of calls to the IsAssigned routine is bounded by $\widetilde{O}(1)$ (by Lemma 5.9), this only adds a constant space overhead.

Algorithm 3 Detecting Edge-Triangle Association

1: **procedure** IsAssigned(triangle $\tau = \{e_1, e_2, e_3\}$, edge e)
2: Let $e_{\min} =$ Assignment(τ)
3: **if** $e_{\min} = \perp$ or $e_{\min} \neq e$ **then**
4: return NO.
5: **else**
6: return YES.
7: **procedure** Assignment(triangle $\tau = \{e_1, e_2, e_3\}$)
8: **for** each edge $e \in \tau$ **do**
9: **if** $d_e > \frac{m\kappa^2}{\varepsilon^2 T}$ **then**
10: $Y_e = \infty$
11: **else**
12: **for** $j = 1$ to s **do**
13: sample w from $N(e)$ u.a.r.
14: If $\{e, w\}$ forms a triangle, set $Y_j = 1$;
15: Else set $Y_j = 0$.
16: Let $Y_e = \frac{d_e}{s} \sum_{j=1}^{\ell} Y_j$.
17: Let $e_{\min} = \text{argmin}_e\, Y_e$.
18: **if** $Y_{e_{\min}} > \kappa/(2\varepsilon)$ **then**
19: return $\perp$.
20: **else**
21: return $e_{\min}$.

We next analyze Algorithm 3. The following theorem captures the theoretical guarantees of IsAssigned procedure.

Theorem 5.13. *Let $\varepsilon > 0$ and $c > 60$ be some positive constants and $s = \frac{c \log n}{\varepsilon^2} \cdot \frac{m\kappa}{T}$. Then, Algorithm 3 leads to an $(\varepsilon, o(1/n^5))$-accurate IsAssigned procedure.*

In the remaining part of this section, we prove the above theorem. We have already discussed how to ensure IsAssigned satisfies item (1) in Definition 5.2. We next take up item (3). We show that not too many triangles are assigned to any fixed edge. In particular, we show that if an edge is "heavy", then with high probability no triangles are assigned to it. In other words, if an edge e is assigned a triangle by Assigned, then with high probability $t_e \leq \kappa/\varepsilon$. It follows then, that for any edge e, the number of triangles that are assigned to e, denoted as τ_e, is at most κ/ε with high probability.

Lemma 5.14. *Let e be an ε-heavy edge. Then with probability at least $1 - \frac{1}{n^5}$, no triangles are assigned to e.*

Proof. First assume e is not an ε-costly edge. Let τ be some triangle that e participates in. We consider an execution of Assignment on input τ. Clearly, $\Pr[Y_j = 1] = t_e/d_e$. By linearity of expectation, $\mathbb{E}[Y_e] = t_e$. An application of Chernoff bound(Theorem 3.3) yields

$$\begin{aligned}\Pr[Y_e < \kappa/(2\varepsilon)] &\leq \Pr[Y_e < t_e/2] \\ &\leq \exp\left(-\frac{1}{12} \cdot \frac{sd_e}{t_e}\right) \\ &\leq \exp\left(-\frac{c \log n}{\varepsilon^2} \cdot \frac{st_e}{d_e}\right) \\ &\leq \frac{1}{n^5}\end{aligned}$$

where the last inequality uses the fact that e is not ε-costly and hence $d_e/t_e \leq m\kappa/(\varepsilon T)$.

Next assume e is an ε-costly edge. Since $t_e > \kappa/\varepsilon$, it follows that $d_e > \frac{m\kappa^2}{\varepsilon^2 T}$. Then, the if condition on Line 9 is true and hence $Y_e = \infty$. So no triangles that e participates in, will be assigned to it. □

We now consider item (2) in Definition 5.2. Let τ be a triangle such that τ is neither ε-heavy nor ε-costly. We prove that, with high probability, Assigned does not return $\perp$ when invoked with τ. By Lemma 5.12, this implies that $\mathcal{T} \geq (1 - 3\varepsilon)T$.

Lemma 5.15. *Let τ be a triangle that is neither 4ε-heavy nor 4ε-costly. Then with probability at least $1 - o(1/n^5)$, Assigned $(\tau) \neq \perp$.*

Proof. Since τ is not 4ε-*heavy*, for each edge $e \in \tau$, $t_e \leq \kappa/(4\varepsilon)$. Since τ is not 4ε-*costly*, at least one edge is not 4ε-*costly* — let e denote that edge. Then, $d_e/t_e \leq m\kappa/(4\varepsilon T)$. Together, they imply $d_e \leq m\kappa^2/(16\varepsilon^2 T)$, and the if condition on Line 8 is not met. We next show that, with high probability $Y_e < \kappa/(2\varepsilon)$.

By linearity of expectation, $\mathbb{E}[Y_e] = t_e$. An application of Chernoff bound(Theorem 3.3), similar to the previous lemma, shows that with probability at least $1 - \frac{1}{n^5}$, $Y_e \leq 2t_e \leq \kappa/(2\varepsilon)$. Hence, with high probability, the triangle τ is assigned to the edge e, proving the lemma. □

We now have all the necessary ingredients to complete the proof of the theorem.

Proof of Theorem 5.13. We have already argued how to ensure that IsAssigned procedure satisfies item (1) in Definition 5.2. Lemmas 5.14 and 5.15 proves that IsAssigned satisfies item (2) and item (3) of Definition 5.2. Finally, the space bound in item (4) is enforced by the setting of the parameter s. □

6 LOWER BOUND

In this section we prove a multi-pass space lower bound for the triangle counting problem. Our lower bound, stated below, is effectively optimal.

Theorem 6.1. *Any constant pass randomized streaming algorithm for graphs with m edges, T triangles, and degeneracy at most κ, that provides a constant factor approximation to T with probability at least $2/3$, requires storage $\Omega(m\kappa/T)$.*

Our proof strategy follows along the expected line of reduction from a suitable communication complexity problem. We reduce from the much-studied SET-DISJOINTNESS problem in communication complexity. It is perhaps a canonical problem that has been used extensively to prove multi-pass lower bounds for various problems, including triangle counting [11, 13]. We consider the following promise version of this problem. Alice and Bob have two N-bit binary strings x and y respectively, each with exactly R ones. They want to decide whether there exists an index $i \in [N]$ such that $x_i = 1 = y_i$. We denote this as the DISJ_R^N problem.

The basis of the reduction is the following lower bound for the DISJ_R^N problem. Assume $\mathbb{R}(\text{DISJ}_R^N)$ denote the randomized communication complexity for the DISJ_R^N problem. [4]

THEOREM 6.2 (BASED ON [40, 49]). *For all $R \leq N/2$, we have $\mathbb{R}(\text{DISJ}_R^N) = \Omega(R)$.*

To prove our lower bound, we reduce the DISJ_R^N problem to the following TRIANGLE-DETECTION problem. Consider two graph families $\mathcal{G}_1$ and $\mathcal{G}_2$; $\mathcal{G}_1$ is a collection of triangle-free graphs on n vertices and m edges with degeneracy κ, and $\mathcal{G}_2$ consists of graphs on same number of vertices and edges, and with degeneracy $\Theta(\kappa)$ and has at least T many triangles. Given a graph $G \in \mathcal{G}_1 \cup \mathcal{G}_2$ as a streaming input, the goal of the TRIANGLE-DETECTION problem is to decide whether $G \in \mathcal{G}_1$ or $G \in \mathcal{G}_2$ with probability at least $2/3$, making a constant number of passes over the input stream. A lower bound for the TRIANGLE-DETECTION problem immediately gives a lower bound for the triangle counting problem.

To set the context for our lower bound result, it is helpful to compare against prior known lower bounds. The multi-pass space complexity of the triangle counting problem is $\Theta(\min\{m^{3/2}/T, m/\sqrt{T}\})$ [11, 22, 46] for graphs with m edges and T triangles. We first consider the first term $m^{3/2}/T$. Since $\kappa = O(\sqrt{m})$, our result subsume the bound of $\Theta(m^{3/2}/T)$. Compared between $m\kappa/T$ and $m/\sqrt{T}$, the former is smaller when $T > \kappa^2$. Necessarily, in our lower bound proofs, we will be dealing with graph instances such that $T > \kappa^2$.

For the purpose of proving a $\Omega(m\kappa/T)$ lower bound, it is sufficient to show that the TRIANGLE-DETECTION problem requires $\Omega(m\kappa/T)$ bits of space for some specific choice of parameters. However, we cover the entire possible range of spectrum. Fix two parameters κ and r, such that $r \geq 2$. Then, we can construct an instance of the TRIANGLE-DETECTION problem with degeneracy $\Theta(\kappa)$ and $T = \kappa^r$ such that solving it requires $\Omega(m\kappa/T)$ bits of space. So in effect, we prove a more nuanced and arguably more general theorem than the one give in Theorem 6.1. Formally, we show the following.

THEOREM 6.3. *Let κ and r be parameters such that $r \geq 2$. Then there is a family of instances with* degeneracy $\Theta(\kappa)$ *and $T = \kappa^r$ such that solving the TRIANGLE-DETECTION problem requires $\Omega(m\kappa/T)$ bits of space.*

PROOF. We reduce from the $\text{DISJ}_{N/3}^N$ problem. Let (x, y) be the input instance for this problem. We then construct an input G for the TRIANGLE-DETECTION problem such that if (x, y) is a YES instance, then $G \in \mathcal{G}_1$, and otherwise $G \in \mathcal{G}_2$. The graph G has a fixed part and a variable part that depends on x and y. We next describe the construction of the graph G.

[4] See [43] for the definition of the notion *randomized communication complexity.*

Let $G_{\text{fixed}} = (A \cup B, E_{\text{fixed}})$ be a complete bipartite graph on the bi-partition A and B. Then, $E_{\text{fixed}} = \{\{a, b\} : a \in A,\ b \in B\}$. Further assume $|A| = |B| = p$. We add N blocks of vertices to G_{fixed} and denote them as $V_1, V_2, \ldots, V_N$. Assume $|V_i| = q$ for each $i \in [N]$. We will set the parameters p and q later in the analysis. For each index $i \in [N]$ such that $x_i = 1$, Alice connects each every vertex in V_i to each vertex in A. Denote this edge set as E_A. For each index $i \in [N]$ such that $y_i = 1$, Bob connects each every vertex in V_i to each vertex in B. Denote this edge set as E_B. This completes the construction of the graph G. To summarize, $G = (V, E)$ where $V = A \cup B \cup V_1 \cup \ldots V_N$, and $E = E_{\text{fixed}} \cup E_A \cup E_B$.

It is easy to see that the graph G is triangle-free if and only if there does not exits any $i \in [N]$ such that $x_i = 1 = y_1$. We now analyze various parameters of G. In both YES and NO case for the $\text{DISJ}_{N/3}^N$ problem, we have

$$n = |V| = 2p + Nq$$
$$m = |E| = p^2 + 2 \cdot \frac{N}{3} \cdot pq\,.$$

In the NO instance, the number of triangles T is at least $p^2 q$. As argued above, in the YES instance, $T = 0$. Finally, we compute the degeneracy κ in both the cases. Note that $\kappa(G_{\text{fixed}}) = p$, and by definition (see Definition 1.1) $\kappa(G) \geq p$. We claim that $\kappa = p$ in the YES instance and $\kappa \leq 2p$ in the NO instance. To prove the claim, we use the following characterization of degeneracy. Let $\prec$ be a total ordering of the vertices and let $d_v^{\prec}$ denote the number of neighbors of v that appears after v according to the ordering $\prec$. Let $d_{\max}^{\prec} = \max_{v \in V} d_v^{\prec}$. Then, $\kappa \leq d_{\max}^{\prec}$. Now consider the following ordering: $V_1 \prec V_2 \prec \ldots V_N \prec A \prec B$, and inside each set the vertices are ordered arbitrarily. Then, in the YES instance, $d_{\max}^{\prec} \leq p$ and in the NO instance $d_{\max}^{\prec} \leq 2p$, proving our claim.

We now set the parameters p and q as $p = \kappa$ and $q = \kappa^{r-2}$. Then, $m = \Theta(Npq)$ since $p = O(Nq)$. Assume there is a constant pass $o(m\kappa/T)$-space streaming algorithm $\mathcal{A}$ for the TRIANGLE-DETECTION problem. Then, following standard reduction, $\mathcal{A}$ can be used to solve the $\text{DISJ}_{N/3}^N$ problem with $o(Npq \cdot p/p^2 q) = o(N)$ bits of communication, contradicting the lower bound for the $\text{DISJ}_{N/3}^N$ problem. □

7 FUTURE DIRECTIONS

In this paper, we studied the streaming complexity of the triangle counting problem in bounded degeneracy graphs. As we emphasized in the introduction, low degeneracy is an often observed characteristics of real-world graphs. Designing streaming algorithms with better bounds on such graphs (compared to the worst case) is an important research direction. There have been some recent successes in this context — graph coloring [12], matching size estimation [6, 23, 32], independent set size approximation [21]. It would be interesting to explore what other problems can admit better streaming algorithms in bounded degeneracy graphs. One natural candidate is the arbitrary fixed size subgraph counting problem, which asks for the number of occurrences of the subgraph in the given input graph.

Do there exist streaming algorithms for approximate subgraph counting on low degeneracy graphs that can beat known worst-case lower bounds?

We conclude this exposition with the following conjecture about the fixed size clique-counting problem.

Conjecture 7.1. *Consider a graph G with degeneracy κ that has T many ℓ-cliques. There exists a constant pass streaming algorithm that outputs a $(1 \pm \varepsilon)$-approximation to T using $\widetilde{O}(m\kappa^{\ell-2}/T)$ bits of space.*

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable feedback. The authors are supported by NSF TRIPODS grant CCF-1740850, NSF CCF-1813165, CCF-1909790, and ARO Award W911NF1910294.

REFERENCES

[1] [n.d.]. Graph Degeneracy. https://en.wikipedia.org/wiki/Degeneracy_(graph_theory).
[2] N. Alon, R. Yuster, and U. Zwick. 1997. Finding and counting given length cycles. *Algorithmica* 17, 3 (1997), 209–223.
[3] Gerold Alsmeyer. 2011. *Chebyshev's Inequality.* Springer Berlin Heidelberg, Berlin, Heidelberg, 239–240.
[4] S. Arifuzzaman, M. Khan, and M. Marathe. 2013. Patric: A parallel algorithm for counting triangles in massive networks. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*. 529–538.
[5] Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. 2018. A Simple Sublinear-Time Algorithm for Counting Arbitrary Subgraphs via Edge Sampling. In *Proc. 10th Conference on Innovations in Theoretical Computer Science*.
[6] Sepehr Assadi, Sanjeev Khanna, and Yang Li. 2017. On estimating maximum matching size in graph streams. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 1723–1742.
[7] Albert Atserias, Martin Grohe, and Dániel Marx. 2008. Size bounds and query plans for relational joins. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 739–748.
[8] H. Avron. 2010. Counting triangles in large graphs using randomized matrix trace estimation. In *Workshop on Large-scale Data Mining: Theory and Applications (LDMTA)*, Vol. 10. 10–9.
[9] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. 2002. Reductions in Streaming Algorithms, with an Application to Counting Triangles in Graphs. In *Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms*. 623–632.
[10] Suman Bera, Noujan Pashanasangi, and C. Seshadhri. 2020. Linear Time Subgraph Counting, Graph Degeneracy, and the Chasm at Size Six. In *Conference on Innovations in Theoretical Computer Science Proc. 11th*.
[11] Suman K. Bera and Amit Chakrabarti. 2017. Towards Tighter Space Bounds for Counting Triangles and Other Substructures in Graph Streams. In *34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*, Vol. 66. 11:1–11:14.
[12] Suman K Bera, Amit Chakrabarti, and Prantar Ghosh. 2019. Graph Coloring via Degeneracy in Streaming and Other Space-Conscious Models. *arXiv preprint arXiv:1905.00566* (2019).
[13] Vladimir Braverman, Rafail Ostrovsky, and Dan Vilenchik. 2013. How Hard Is Counting Triangles in the Streaming Model?. In *Proc. 40th International Colloquium on Automata, Languages and Programming*. 244–254.
[14] Luciana S. Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. 2006. Counting Triangles in Data Streams. In *Proc. 25th ACM Symposium on Principles of Database Systems*. 253–262.
[15] Amit Chakrabarti. [n.d.]. CS49: Data Stream Algorithms Lecture Notes, Fall 2011. http://www.cs.dartmouth.edu/~ac/Teach/data-streams-lecnotes.pdf
[16] MT Chao. 1982. A general purpose unequal probability sampling plan. *Biometrika* 69, 3 (1982), 653–656.
[17] H. Chernoff. 1952. A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations. *Annals of Mathematical Statistics* 23, 4 (1952), 493–507.
[18] N. Chiba and T. Nishizeki. 1985. Arboricity and subgraph listing algorithms. *SIAM J. Comput.* 14, 1 (1985), 210–223.
[19] Norishige Chiba and Takao Nishizeki. 1985. Arboricity and Subgraph Listing Algorithms. *SIAM J. Comput.* 14, 1 (1985), 210–223.
[20] S. Chu and J. Cheng. 2011. Triangle listing in massive networks and its applications. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 672–680.
[21] Graham Cormode, Jacques Dark, and Christian Konrad. 2017. Independent set size approximation in graph streams. *arXiv preprint arXiv:1702.08299* (2017).
[22] Graham Cormode and Hossein Jowhari. 2014. A second look at counting triangles in graph streams. *Theoretical Computer Science* 552 (2014), 44–51.
[23] Graham Cormode, Hossein Jowhari, Morteza Monemizadeh, and Shanmugavelayutham Muthukrishnan. 2017. The sparse awakens: Streaming algorithms for matching size estimation in sparse graphs. In *25th European Symposium on Algorithms, ESA 2017*. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 29.
[24] M. Danisch, O. D. Balalau, and M. Sozio. 2018. Listing k-cliques in Sparse Real-World Graphs. In *Conference on the World Wide Web (WWW)*. 589–598.
[25] Nurcan Durak, Ali Pinar, Tamara G Kolda, and C Seshadhri. 2012. Degree relations of triangles in real-world networks and graph models. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 1712–1716.
[26] T. Eden, A. Levi, D. Ron, and C Seshadhri. 2015. Approximately counting triangles in sublinear time. In *Foundations of Computer Science (FOCS)*. 614–633.
[27] Talya Eden, Dana Ron, and C Seshadhri. 2017. Sublinear Time Estimation of Degree Distribution Moments: The Degeneracy Connection. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
[28] T. Eden, D. Ron, and C. Seshadhri. 2018. Faster sublinear approximations of k-cliques for low arboricity graphs. arXiv:cs.DS/1811.04425
[29] T. Eden, D. Ron, and C. Seshadhri. 2018. On approximating the number of k-cliques in sublinear time. In *Symposium on Theory of Computing (STOC)*. 722–734.
[30] T. Eden, D. Ron, and C. Seshadhri. 2020. Faster sublinear approximation of the number of k-cliques in low-arboricity graphs. In *Symposium on Discrete Algorithms (SODA)*.
[31] David Eppstein. 1994. Arboricity and bipartite subgraph listing algorithms. *Information processing letters* 51, 4 (1994), 207–211.
[32] Hossein Esfandiari, Mohammadtaghi Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. 2018. Streaming algorithms for estimating the matching size in planar graphs and beyond. *ACM Transactions on Algorithms (TALG)* 14, 4 (2018), 48.
[33] I. Finocchi, M. Finocchi, and E. G. Fusco. 2015. Clique Counting in MapReduce: Algorithms and Experiments. *ACM Journal of Experimental Algorithmics* 20 (2015), 1–7. https://doi.org/10.1145/2794080
[34] David Garcıa-Soriano and Konstantin Kutzkov. 2014. Triangle counting in streamed graphs via small vertex covers. *Tc* 2 (2014), 3.
[35] G. Goel and J. Gustedt. 2006. Bounded arboricity to determine the local structure of sparse graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*. Springer, 159–167.
[36] S. Jain and C. Seshadhri. 2017. A Fast and Provable Method for Estimating Clique Counts Using Turán's Theorem. In *Conference on the World Wide Web (WWW)*. 441–449.
[37] Madhav Jha, C Seshadhri, and Ali Pinar. 2013. A space efficient streaming algorithm for triangle counting using the birthday paradox. In *Proc. 19th Annual SIGKDD International Conference on Knowledge Discovery and Data Mining*. 589–597.
[38] Hossein Jowhari and Mohammad Ghodsi. 2005. New streaming algorithms for counting triangles in graphs. In *Computing and Combinatorics*. Springer, 710–716.
[39] John Kallaugher and Eric Price. 2017. A hybrid sampling scheme for triangle counting. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 1778–1797.
[40] B. Kalyanasundaram and G. Schintger. 1992. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics* 5, 4 (1992), 545–557.
[41] Daniel M Kane, Kurt Mehlhorn, Thomas Sauerwald, and He Sun. 2012. Counting arbitrary subgraphs in data streams. In *Proc. 39th International Colloquium on Automata, Languages and Programming*. 598–609.
[42] M. N. Kolountzakis, G. L. Miller, R. Peng, and C. E. Tsourakakis. 2012. Efficient triangle counting in large graphs via degree-based vertex partitioning. *Internet Mathematics* 8, 1-2 (2012), 161–185.
[43] Eyal Kushilevitz. 1997. Communication complexity. In *Advances in Computers*. Vol. 44. 331–360.
[44] Konstantin Kutzkov and Rasmus Pagh. 2014. Triangle counting in dynamic graph streams. In *Proc. 14th Scandinavian Symposium and Workshops on Algorithm Theory*. 306–318.
[45] Madhusudan Manjunath, Kurt Mehlhorn, Konstantinos Panagiotou, and He Sun. 2011. Approximate Counting of Cycles in Streams. In *Proc. 19th Annual European Symposium on Algorithms*. 677–688.
[46] Andrew McGregor, Sofya Vorotnikova, and Hoa T. Vu. 2016. Better Algorithms for Counting Triangles in Data Streams. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 401–411.
[47] Rasmus Pagh and Charalampos E. Tsourakakis. 2012. Colorful triangle counting and a mapreduce implementation. *Inform. Process. Lett.* 112, 7 (2012), 277–281.
[48] Aduri Pavan, Kanat Tangwongsan, Srikanta Tirthapura, and Kun-Lung Wu. 2013. Counting and sampling triangles from a graph stream. *Proceedings of the VLDB Endowment* 6, 14 (2013), 1870–1881.

[49] A. A. Razborov. 1992. On the distributional complexity of disjointness. *Theoretical Computer Science* 106, 2 (1992), 385–390.

[50] Alessandra Sala, Lili Cao, Christo Wilson, Robert Zablit, Haitao Zheng, and Ben Y Zhao. 2010. Measurement-calibrated graph models for social network experiments. In *Proceedings of the 19th international conference on World wide web*. ACM, 861–870.

[51] T. Schank and D. Wagner. 2005. Approximating Clustering Coefficient and Transitivity. *Journal of Graph Algorithms and Applications* 9 (2005), 265–275. Issue 2.

[52] T. Schank and D. Wagner. 2005. Finding, Counting and Listing All Triangles in Large Graphs, an Experimental Study. In *Experimental and Efficient Algorithms*. 606–609.

[53] Comandur Seshadhri, Tamara G Kolda, and Ali Pinar. 2012. Community structure and scale-free collections of Erdős-Rényi graphs. *Physical Review E* 85, 5 (2012), 056109.

[54] C. Seshadhri, A. Pinar, and T. G. Kolda. 2013. Fast Triangle Counting through Wedge Sampling. arXiv:1202.5230. In *Proceedings of the International Conference on Data Mining (ICDM)*, Vol. 4. 5. http://arxiv.org/abs/1202.5230

[55] K. Shin, T. Eliassi-Rad, and C. Faloutsos. 2018. Patterns and anomalies in k-cores of real-world graphs with applications. *Knowledge and Information Systems* 54, 3 (2018), 677–710.

[56] S. Suri and S. Vassilvitskii. 2011. Counting triangles and the curse of the last reducer. In *Proceedings of the International Conference on World Wide Web (WWW)*. 607–614. https://doi.org/10.1145/1963405.1963491

[57] K. Tangwongsan, A. Pavan, and S. Tirthapura. 2013. Parallel Triangle Counting in Massive Streaming Graphs. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*. ACM, 781–786.

[58] C. E. Tsourakakis. 2008. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *International Conference on Data Mining (ICDM)*. 608–617.

[59] C. E. Tsourakakis, U. Kang, G.L. Miller, and C. Faloutsos. 2009. Doulion: counting triangles in massive graphs with a coin. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 837–846.

[60] C. E. Tsourakakis, M. N. Kolountzakis, and G. L. Miller. 2011. Triangle Sparsifiers. *Journal of Graph Algorithms and Applications* 15, 6 (2011), 703–726.

[61] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world'networks. *nature* 393, 6684 (1998), 440.