The Complexity of Counting Cycles in the Adjacency List Streaming Model

John Kallaugher
UT Austin
jmgk@cs.utexas.edu

Andrew McGregor
UMass Amherst
mcgregor@cs.umass.edu

Eric Price
UT Austin
ecprice@cs.utexas.edu

Sofya Vorotnikova
UMass Amherst
svorotni@cs.umass.edu

ABSTRACT

We study the problem of counting cycles in the adjacency list streaming model, fully resolving in which settings there exist sublinear space algorithms. Our main upper bound is a two-pass algorithm for estimating triangles that uses $\widetilde{O}(m/T^{2/3})$ space, where m is the edge count and T is the triangle count of the graph. On the other hand, we show that no sublinear space multipass algorithm exists for counting ℓ-cycles for $\ell \geq 5$. Finally, we show that counting 4-cycles is intermediate: sublinear space algorithms exist in multipass but not single-pass settings.

CCS CONCEPTS

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

KEYWORDS

Data streams, triangles, cycles.

ACM Reference Format:
John Kallaugher, Andrew McGregor, Eric Price, and Sofya Vorotnikova. 2019. The Complexity of Counting Cycles in the Adjacency List Streaming Model. In *38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS'19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3294052.3319706

PODS'19, June 30-July 5, 2019, Amsterdam, Netherlands

ACM ISBN 978-1-4503-6227-6/19/06…$15.00
https://doi.org/10.1145/3294052.3319706

1 INTRODUCTION

Subgraph counting is a fundamental graph problem and an important primitive in massive graph analysis. It has many applications in data mining and analyzing the structure of large networks. In particular, triangle counting and the related problems of estimating the transitivity and clustering coefficients arise in spam detection [4], community detection in social networks [7], identification of web pages with a common topic [14], and evaluation of large graph models [24]; see Tsourakakis et al. [33] for an overview of applications.

Triangle counting has been studied in various models of computation for large inputs, such as MapReduce [32], other parallel models [2, 6], and external memory models [1]. In the data stream model, triangle counting was first introduced by Bar-Yossef, Kumar, and Sivakumar [3]. Subsequent work has studied algorithms in a number of streaming models [5, 9, 12, 13, 16–18, 21, 23, 26–28]. Other subgraphs have received less attention in the streaming community, with most work focused on cycles and cliques [5, 25, 28] and a few papers considering arbitrary subgraphs of constant size [5, 18, 20].

1.1 Prior Work

Triangle Counting. Two adversarial *insertion-only* streaming models have been studied in the literature: *arbitrary order*, where the edges can arrive in any order, and *adjacency list order*, where edges incident to the same vertex arrive together (and therefore every edge is required to appear twice). In discussing related work, we use T to denote the number of triangles in the graph and $\widetilde{O}(\cdot)$ to hide factors polynomial in $\log n$ and ε^{-1}. In the single-pass arbitrary order model, $\Omega(m)$ space is required to even distinguish between graphs with 0 and T triangles [9] for any $T < n$, and thus work has largely concentrated on providing space bounds parameterized by properties of the graph, such as the maximum degree [17, 28], tangle coefficient [28], number of paths of length two [16], or the maximum number of triangles sharing an edge or a vertex [18, 27]; see [5] for a summary of these results. The best known results without such additional

Cycle	Passes	Space	Comments	Source
Triangle	1	$\widetilde{O}(P_2/T)$	P_2 = number of paths of length two	[12]
	1	$\widetilde{O}(m/\sqrt{T})$	-	[26]
	3	$\widetilde{O}(\sqrt{m} + m^{3/2}/T)$	-	[21]
	3	$\widetilde{O}(m^{3/2}/T)$	-	[26]
	2	$\widetilde{O}(m/T^{2/3})$	Distinguishing between 0 and T triangles	[26]
	2	$\widetilde{O}(m/T^{2/3})$	-	Theorem 3.7
	1	$\Omega(f_{pj}(m/\sqrt{T}))$	$\Omega(f_{pj}(r))$ = communication complexity of 3-PJ$_r$; $T = O(m)$	Theorem 5.1
	const	$\Omega(f_d(m/T^{2/3}))$	$\Omega(f_d(r))$ = communication complexity of 3-DISJ$_r$; $T = O(m^{3/2})$	Theorem 5.2
$\ell = 4$	2	$\widetilde{O}(m/T^{3/8})$	$O(1)$-factor approximation	Theorem 4.6
	1	$\Omega(m)$	$T = O(n^{1/3})$	Theorem 5.3
	const	$\Omega(m/T^{2/3})$	$T = O(n^{3/4})$	Theorem 5.4
$\ell \geq 5$	const	$\Omega(m)$	$T = O(n)$	Theorem 5.5

Table 1: Cycle counting in adjacency list insertion-only streams. Unless specified otherwise, upper bounds are for $(1+\varepsilon)$-estimating the number of cycles and lower bounds for distinguishing between graphs with 0 and T cycles.

parameters are either the trivial $O(m)$ or—for graphs with many triangles—$\widetilde{O}(mn/T)$ [12].

More recently, it has been shown that algorithms using a constant number of passes over the stream can achieve sublinear space for any $T = \omega(1)$. The optimal lower bound depending on m and T only was given by Bera and Chakrabarti in [5]. They proved an $\Omega(\min\{m^{3/2}/T, m/\sqrt{T}\})$ bound and gave an $\widetilde{O}(m^{3/2}/T)$ space algorithm matching one of the terms. Independently, McGregor, Vorotnikova, and Vu [26] gave two algorithms matching both terms of the lower bound, and Cormode and Jowhari [13] gave a different algorithm with $\widetilde{O}(m/\sqrt{T})$ space[1].

In the adjacency list model, on the other hand, it is possible to achieve sublinear space without additional parameters. Prior to this work, the two state-of-the-art algorithms for $(1+\varepsilon)$-approximating the number of triangles were a single-pass algorithm using $\widetilde{O}(m/\sqrt{T})$ space and a two-pass algorithm using $\widetilde{O}(m^{3/2}/T)$ space by McGregor et al. [26]. They also gave a two-pass algorithm for distinguishing triangle-free graphs from those with at least T triangles that uses only $\widetilde{O}(m/T^{2/3})$ space.

Counting Other Subgraphs. Prior to this work, counting constant size subgraphs other than triangles had only been considered in the arbitrary order setting.

Bera and Chakrabarti [5] describe an algorithm $(1+\varepsilon)$-approximating the number of occurrences of a particular subgraph H of size h using $\widetilde{O}(m^{\beta(H)}/T)$ space and two passes over the stream, where $\beta(H)$ denotes the size of an edge cover of H. They improve on this algorithm for cliques and cycles, achieving $\widetilde{O}(m^{h/2}/T)$ space, and provide a matching lower bound. For single-pass algorithms they show lower bounds of $\Omega(m^h/T^2)$ for cliques and odd-length cycles and $\Omega(m^{h/2}/T)$ for even-length cycles. Kallaugher and Price [18] presented an algorithm for counting copies of an arbitrary subgraph H which depends on how much the subgraphs overlap with each other; it is sublinear as long as no constant-size set of vertices contains a constant fraction of the copies of H.

1.2 Our Setting

Streaming model. We consider the *adjacency list* streaming model. In this model, we assume that the stream consists of a sequence of ordered pairs xy. For each edge $\{x, y\}$, both xy and yx will be present in the stream. The promise on the ordering is that all pairs with the same first vertex (the adjacency list of that vertex) appear consecutively in the stream. Within the adjacency lists the stream is ordered arbitrarily. For example, for the graph consisting of a cycle on three vertices $V = \{v_1, v_2, v_3\}$, a possible ordering of the stream could be $\langle v_3v_1, v_3v_2, v_1v_2, v_1v_3, v_2v_3, v_2v_1\rangle$. In this example, we say that the adjacency list for v_3 came first, then the adjacency list for v_1, and finally the adjacency list for v_2.

[1] Note that in the original version of the paper, Cormode and Jowhari initially claimed that their algorithm returned a $(1+\varepsilon)$-approximation but this claim was incorrect. Instead, their proposed algorithm returned a $(3+\varepsilon)$-estimate. Subsequently, they designed a new $(1+\varepsilon)$- approximation algorithm which appeared in the revised version of the paper.

Cycle Counting. For $\ell \geq 3$, we will consider algorithms for counting the number of ℓ-cycles in a graph G presented as an adjacency list stream. G has T cycles, with T unknown to the algorithm, and the problem is to compute a multiplicative approximation to T with probability $1-\delta$. The approximation factor may be $(1 \pm \varepsilon)$, with ε an arbitrary parameter, or it may be a fixed constant.

Our algorithms are parametrized in terms of T, which is a convention widely adopted in the literature on subgraph counting. While T cannot be assumed to be known in advance, it suffices to know a lower bound on $T' \leq T$, and then set our space usage based on T', as our bounds will only be tighter if $T > T'$. Furthermore, if T is in fact smaller than T', then we will not be able to accurately estimate T, but we will be able to determine that it is smaller than T'. This is because our bounds are variance-based, with variance that is increasing in T.

We will be primarily concerned with the *space* complexity of the algorithms, expressed in terms of the edge count m, T, ε, and δ. When ε and δ are omitted, this denotes the complexity when ε and δ are constant.

1.3 Our Results

We present new upper and lower bounds for cycle counting in adjacency list streams. See Table 1 for a summary of known results and our contributions. For triangle counting we give a two-pass $(1+\varepsilon)$-estimation algorithm using $\widetilde{O}(m/T^{2/3})$ space, improving on the previous best known multipass algorithm. We also present the first adjacency list algorithm for approximately counting 4-cycles, using $\widetilde{O}(m/T^{3/8})$ space and two passes over the stream to return a constant-factor approximation to T. We complement our upper bounds with lower bound results for all cycle lengths. These are the first lower bounds for subgraph counting in the adjacency list model and they fully resolve in which settings there exist sublinear space cycle counting/distinguishing algorithms.

Both of our algorithms are based on sampling. Sampling-based approaches to this problem are very sensitive to "heavy" edges, i.e., edges which are involved in a large number of cycles. Heavy edges often introduce unacceptably large variance to the estimator. In our triangle counting algorithm, we deal with this by defining, for each triangle, a "lightest" edge, and ensuring that the triangle is only counted if it is initially sampled at that edge. See Sections 2.1 and 3 for more in-depth discussion.

In our 4-cycle counting algorithm, we deal with this problem by showing that a sufficient fraction of the 4-cycles have at least one wedge that is sufficiently "light". The difference from triangle counting is that the algorithm cannot *identify* the "light" wedge during the stream; this leads to a constant factor approximation to the cycle count, rather than $1+\varepsilon$. See Sections 2.2 and 4 for details.

We present lower bounds for counting cycles of all length, both in one pass and constant number of passes. We use reductions from four known communication complexity problems: Index (INDEX_r), three-player Number on Forehead Pointer-jumping (3-PJ_r), two-player Disjointness (DISJ_r) and three-player NOF Disjointness (3-DISJ_r), where r indicates the size of the input. The communication complexity of the two multi-player Number on Forehead (NOF) problems have not yet been fully resolved [10, 31] and thus we present our lower bounds for triangle counting as conditional on the best communication complexity lower bounds for these problems. It is generally believed that both of these problems have communication complexity $\widetilde{\Omega}(r)$, which would imply lower bounds of $\widetilde{\Omega}(m/\sqrt{T})$ and $\widetilde{\Omega}(m/T^{2/3})$ for one pass and multipass triangle counting respectively, matching the corresponding algorithms (from [26] and here) up to polylog factors; certainly any result to the contrary would be a breakthrough. For 4-cycle counting we show that solving the problem in one pass requires $\Omega(m)$ space, but in two passes it can be done using sublinear amount of space. We then show that counting larger cycles requires $\Omega(m)$ space for any constant number of passes. The details can be found in Sections 2.3 and 5.

2 TECHNIQUES

2.1 Triangle Counting Algorithm

Our main upper bound is a two-pass adjacency list streaming algorithm that counts the number of triangles in a graph using $O\left(m/T^{2/3} \cdot \log n\right)$ space, where T is the number of triangles in the graph and m is the number of edges. To motivate our methods, consider first the following algorithm for distinguishing graphs with T triangles from graphs with no triangles described in [26]:

First Pass Sample m' edges from the graph.
Second Pass Check if any of these edges are in a triangle.

Note that the second pass can be accomplished with only $2m'$ extra bits of storage by, for each adjacency list presented in the second pass, flagging any endpoint of a sampled vertex if it appears in the adjacency list. If both endpoints of an edge are flagged in the adjacency list of some vertex v, that edge forms a triangle with v.

It can be shown (see, e.g., [15]) that any graph with T triangles must have at least $T^{2/3}$ edges involved in those triangles, so provided $m' \geq m/T^{2/3}$, at least one such edge will be sampled in the first pass with good probability, and so the algorithm will detect that the graph has at least one triangle.

This algorithm gives an unbiased estimator for T, as the number of triangles found by the algorithm is $3m'T/m$ in expectation (if triangles with multiple sampled edges are counted with multiplicity). However, this estimator has high variance, due to "heavy" edges—edges involved in many triangles. We can reduce the variance by discarding excessively heavy edges (for instance, if we discard every edge involved in at least $100T^{1/3}$ triangles we can reduce the variance to $O(T^2)$), but at the cost of biasing the estimator. This approach will allow us a constant-factor approximation to T, but not the $(1 \pm \varepsilon)$-multiplicative approximation we want.

Instead, consider the following three-pass algorithm, which only counts a triangle as being sampled if its "lightest" edge was sampled:

First Pass Sample a size-m' set S of edges from the graph.
Second Pass Collect the set Q of triangles that include at least one edge in S.
Third Pass For every triangle $uvw \in Q$, calculate T_{uv}, T_{vw}, T_{wu}, the number of triangles in G that use uv, vw, and wu, respectively.
Post-Processing For each edge $e \in S$, and each triangle τ using e, count τ iff $e = \arg\min_{e' \in \tau} T_{e'}$, breaking ties arbitrarily.

This algorithm discards most "heavy edges" (it is possible for a heavy edge to be the lightest edge of *some* triangle, but not of *most* triangles), while providing an unbiased estimator, as each triangle has an exactly m'/m chance of being counted. One can show that this leads to an unbiased estimator with good variance.

However, there are two problems with this algorithm. Firstly, it takes an extra pass, and secondly, we need to store Q, which is size $m'T/m$ in expectation. Therefore, its space complexity is, in the worst case,

$$O\left(\max\left(m/T^{2/3}, T^{1/3}\right)\log n\right)$$

To deal with the second problem, we sample a size-m' subset of the triangles we would collect in the second pass, or all of them if we see fewer than m'. This gives us a true $O\left(m/T^{2/3} \cdot \log n\right)$-space algorithm.

Now we consider the first problem. Our algorithm depends on, for each triangle sampled, determining whether it was sampled at its "lightest" edge. It is impossible to determine this exactly in only two passes, as for each edge e' in a triangle τ, we will only be able to count triangles that involve e' in the postfix of the stream that appears *after* τ is first detected.

To solve this, we first note that, if we sample the edge uv in our first pass, we will be able to add it to S when the first of the adjacency lists of u, v arrive (assuming we use a hash-based sampling method). Suppose this is u. Then, if uv is completed by a vertex w to form τ, either w appears after u, in which case τ can be found during the first pass, or it appears before u, in which case τ can be found in the second pass as soon as w arrives. In either case, we can find τ by the time its first vertex arrives in the second pass.

We can therefore replace $\arg\min_{e' \in \tau} T_{e'}$ from the three-pass algorithm with $\arg\min_{e' \in \tau} H_{e',\tau}$, where

$$H_{e',\tau} = |\{\tau' : e' \in \tau',\ (\tau' \setminus e') \text{ arrives after } (\tau \setminus e')\}|,$$

because we can compute $H_{e',\tau}$ in the second pass for every $\tau \in Q$ and edge $e' \in \tau$.

For any given edge e' and triangle, this can be arbitrarily small, but on average across all triangles that use an edge e', $H_{e',\tau}$ will be approximately $T_{e'}/2$, which will suffice to bound the variance of our algorithm (for further details, see the full discussion in Section 3). This gives us the following algorithm:

First Pass Sample a size-m' set S of edges from the graph.
First and Second Pass Collect the set Q of triangles that include at least one edge in S.
Second Pass For each $\tau \in Q$ and each $e' \in \tau$, calculate $H_{e',\tau}$.
Post-Processing For each edge $e \in S$, and each triangle τ using e, count τ iff $e = \arg\min_{e' \in \tau} H_{e',\tau}$, breaking ties arbitrarily.

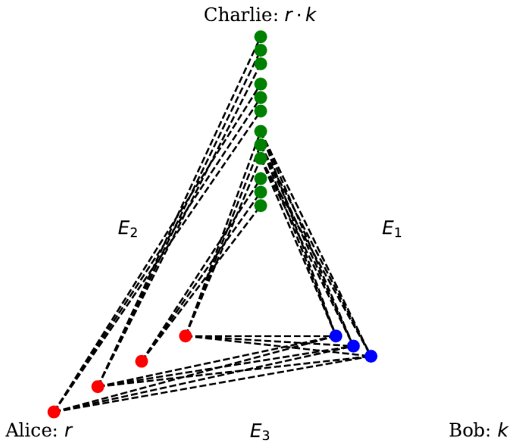
Combining with the aforementioned fix—only storing a sample of Q—gives our final algorithm.

2.2 4-Cycle Counting Algorithm

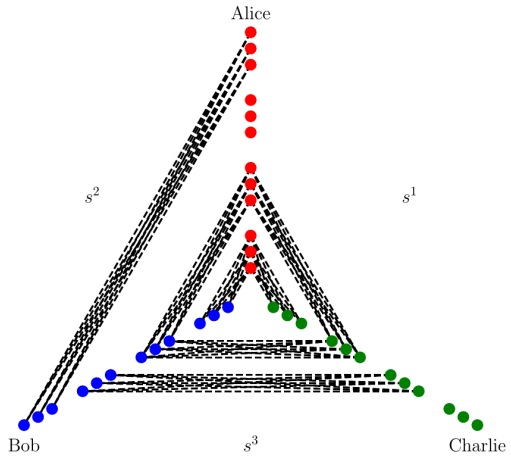
Our second result is a two-pass adjacency list streaming algorithm that returns an $O(1)$ multiplicative approximation to the 4-cycle count of a graph using $\widetilde{O}\left(m/T^{3/8}\right)$ space, where T is the number of 4-cycles in the graph and m is the number of edges. In the first pass, we collect a set of edges S and in the second, count how many 4-cycles G has that contain a wedge in S. To sample even one cycle reliably, we need S to be size at least $m/T^{3/8}$, as any wedge will be sampled with probability about $(|S|/m)^2$, and there can be as few as $T^{3/4}$ wedges in a graph with T 4-cycles. We show that this also gives a $O(1)$-factor multiplicative approximation to T by showing that at least a constant fraction of the cycles in G have at least one wedge that is "good", i.e. that is not used by too many 4-cycles and has neither of its edges used by too many 4-cycles.

2.3 Lower Bounds

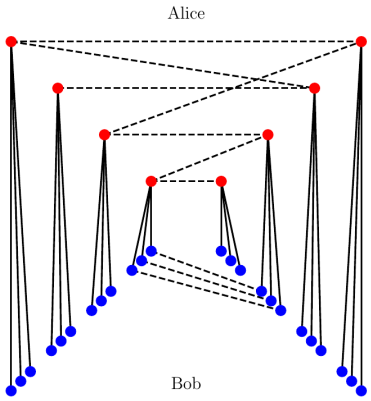
Our last set of results consists of five lower bounds on the space complexity of cycle counting in the adjacency list streaming model. All of them involve the standard method of using reductions from known communication complexity problems. We let C_r be a two or three-player communication complexity problem with input of size $\Theta(r)$ and a



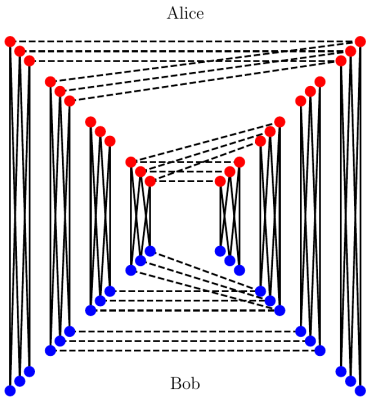
(a) One-pass Triangle Counting $\widetilde{\Omega}(m/\sqrt{T})$ **from** 3-PJ **(conditional)**



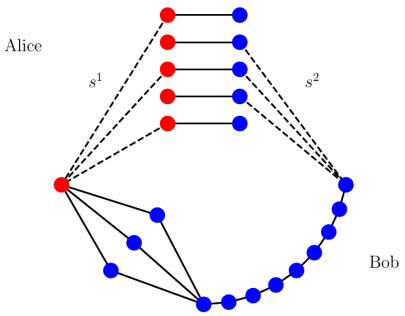
(b) Multi-pass Triangle Counting $\widetilde{\Omega}(m/T^{2/3})$ **from** 3-DISJ **(conditional)**



(c) One-pass 4-cycle Counting $\Omega(m)$ **from** INDEX



(d) Multi-pass 4-cycle Counting $\Omega(m/T^{2/3})$ **from** DISJ



(e) Multi-pass ℓ-cycle Counting, $\ell \geq 5$ $\Omega(m)$ **from** DISJ

Figure 1: Lower bound constructions.

single bit as output. Suppose $\mathcal{A}$ is an algorithm that can distinguish between graphs with T ℓ-cycles and ℓ-cycle-free graphs with probability 2/3 in the one-pass adjacency list streaming model. We then construct a protocol for C_r which sends messages whose size is the internal space of $\mathcal{A}$. Based on an instance of C_r we construct a graph G, such that G is ℓ-cycle-free if the output is 0 and has T ℓ-cycles if the output is 1. Alice, Bob (and possibly Charlie) each encode their input as adjacency lists of vertices in the graph G. The protocol is then for Alice to run $\mathcal{A}$ on her adjacency lists, and send the algorithm's state to Bob, who then runs it on his adjacency lists. If there are three players, Bob sends the state to Charlie, who then runs $\mathcal{A}$ on his lists. The last player can tell whether G has 0 or T ℓ-cycles and therefore whether the output of C_r is 0 or 1. This implies that if C_r requires $\Omega(f(r))$ one-way communication, $\mathcal{A}$ must use $\Omega(f(r))$ space.

To obtain a bound on the space used by multi-pass algorithms, we allow players to communicate for an arbitrary number of rounds and consider total communication. If total communication complexity of C_r is $\Omega(f(r))$, then any algorithm $\mathcal{A}$ with c passes must use $\Omega(f(r)/c)$ space, which is $\Omega(f(r))$ for constant c.

For our reductions, we use the following communication complexity problems (formally defined in Section 5):

- INDEX: Alice has $s \in \{0,1\}^r$, Bob has $x \in [r]$, and must output s_x.
- DISJ: Alice and Bob have $s^1, s^2 \in \{0,1\}^r$ and must determine if any $s^1_x = s^2_x = 1$.
- 3-PJ: Three-player Number on Forehead (NOF) Pointer-jumping. This problem can be viewed as an extension of Index to the multi-party NOF setting.
- 3-DISJ: Three-player NOF Disjointness.

Using reductions from Index and Disjointness is standard for proving lower bounds on the space of one-pass and multi-pass arbitrary order streaming algorithms. In adjacency list streams, they are only applicable for obtaining bounds on counting subgraphs with two disjoint edges, and thus cannot be used for triangles. Since the input of a given communication complexity problem is encoded in the edges of the graph and each player sees every edge on certain vertices, assigning two vertices of a triangle to Alice and the third one to Bob (or vice versa) would lead to the player with two vertices having information about the other player's input. On the other hand, in a 4-cycle (or a larger cycle) we can assign two adjacent vertices to Alice and two other adjacent vertices to Bob, which would insure that both of them have private input. To circumvent this problem for triangles, we employ three-player communication complexity problems, where each player knows two thirds of the input. This is a special case of the k-player Number on Forehead (NOF) model, where the input consists of k parts I_1 through I_k and i-th player knows all I_j *except* I_i. Showing bounds on communication complexity in this model has proven harder than in the more standard Number in Hand model. For multiparty NOF Pointer Jumping and Disjointness there is currently a gap between best known upper and lower bounds on the communication complexity. The best known lower bound for three players is currently $\Omega(\sqrt{r})$ for both problems, where r is the size of input [30, 34]. Thus, we reference $f_{pj}(r)$ and $f_d(r)$ as (currently unknown) complexities of NOF Pointer Jumping and Disjointness respectively in our triangle counting lower bounds. It is conjectured, that the communication complexity of both problems is $\widetilde{\Omega}(r)$, where $\widetilde{\Omega}(\cdot)$ notation hides inverse polylog factors. If that is the case, our triangle counting bounds become tight.

For $\ell \geq 4$, we encode classic one-way (for one-pass) or two-way (for multi-pass) communication complexity problems as subgraph counting problems. This requires us to encode the players' inputs in *disjoint* edges, as if we use incident edges the players will share information, and so it is not available for $\ell = 3$.

In two of our reductions concerning counting 4-cycles we aim to construct a graph with a large number of 4-cycles for instances with output 1 and a similar 4-cycle-free graph for 0-instances. To achieve that, we employ bipartite 4-cycle-free graphs on $2r$ vertices with $\Theta(r^{3/2})$ edges. An example of such a graph is an incidence graph of a special kind of projective plane called a field plane. If the order of the plane is q, it has $q^2 + q + 1$ points, the same number of lines, and each line passes through exactly $q + 1$ points. Thus, the incidence graph has $2(q^2 + q + 1)$ vertices, each with degree $q + 1$. The graph is 4-cycle-free since by definition of a projective plane, for any two distinct points, there is exactly one line passing through both of them, and for any two distinct lines, there is exactly one point both of them pass through. We also note that there are no 4-cycle-free graphs on r vertices with $\omega(r^{3/2})$ edges [8].

When $\ell \geq 5$ our lower bounds become $\Omega(m)$ for any constant number of passes regardless of the number of cycles in the graph. This is because we can encode Disjointness (which is known to require linear communication even in two-way complexity) as an ℓ-cycle-counting problem. This encoding will depend on the fact that a pair of disjoint edges can be shared by many different ℓ-cycles, which does not hold for $\ell = 4$.

For illustrations of the lower bound constructions see Figure 1. Solid edges are fixed and dashed edges correspond to players' input. For detailed definitions of communication complexity problems and descriptions of the reductions see Section 5.

3 TWO PASS TRIANGLE COUNTING ALGORITHM

3.1 Notation

For any edge or subset of edges x in G, write $L(x)$ for the set of triangles involving x, and write $T(x)$ for $|L(x)|$. Let $T = T(G)$. For each triangle $\tau \in L(G)$, and each edge $e \in \tau$, let τ^{-e} denote the unique vertex in τ that is not an endpoint of e.

3.2 Algorithm

Our algorithm will take two passes over the stream. Call these passes P_1, P_2. We will require that P_2 have the same ordering as P_1. For each $i \in [2]$, $u \in V(G)$ and $v \in \Gamma(u)$, let $P_i^{<uv}$ denote the prefix of P_i that appears before v appears in the adjacency list of u, and $P_i^{>uv}$ denote the postfix of P_i that appears afterwards. Similarly, let $P_i^{<u}$,$P_i^{>v}$ denote the prefix and postfix of P_i that occur before and after the adjacency list of v, respectively. For each $e \in G$, define an order on the triangles of the graph as follows: $\tau_1 <_e \tau_2$ if τ_2^{-e} arrives after τ_1^{-e} in the stream.

Now, let τ be a triangle and e an edge. Define:

$$H_{e,\tau} = |\{\tau' \in L(e) : \tau <_e \tau'\}|$$

Then, for each triangle τ, let $\rho(\tau)$ be the unique edge $e \in \tau$ that minimizes $H_{e,\tau}$, breaking ties arbitrarily. Let

$$\widetilde{T}_e = |\{\tau \in L(e) : \rho(\tau) = e\}|$$

Note that $\sum_{e \in E} \widetilde{T}_e = T$. Our algorithm is then as follows:

(1) Choose a sample size m'.
(2) While passing through P_1, keep a uniformly chosen size-m' subset S of $E(G)$, adding an edge to S the first time of the two times it appears in P_1. If $m' > m$, instead keep all of $E(G)$.
(3) While passing through P_1 and P_2, perform the following steps in parallel:
   (a) Calculate $m = |E(G)|$. Define $k = \max(m/m', 1)$.
   (b) Calculate $T' = \sum_{e \in S} T_e$.
   (c) Sample a size-m' subset Q uniformly from $\{(e, \tau) : e \in S, \tau \in L(e)\}$, or let Q be the entire set if it is smaller than m'.
   (d) For each $(e', \tau) \in Q$ and for each $e \in \tau$, calculate $H_{e,\tau}$.
(4) Return $\widetilde{T}' = \frac{k^2 T'}{m} |\{(e, \tau) \in Q : \rho(\tau) = e\}|$.

3.3 Analysis

3.3.1 Space Complexity. To see that this can be implemented, first note that, if we are storing an edge $e = uv$, whenever the adjacency list of a vertex w appears in the stream we can check whether uvw forms a triangle in G with the use of only two extra bits per such edge e. This is because, while scanning the adjacency list of w, we can flag u if uw appears, and v if vw appears, concluding at the end of the adjacency list that uvw exists iff v and w are flagged.

Therefore, we can perform step 3b with $O(|S| \log n)$ space, and step 3c with $O(|Q| \log n)$ space. We can perform step 3a in $O(\log n)$ space just by maintaining a counter.

Then, for step 3d, consider any $(e, \tau) \in Q$. The first time this can be added to Q is the first time the vertex $w = \tau^{-e}$ is seen after adding e to S. Suppose e appears in P_1 for the first time as the vertex v in the adjacency list of u. Then, as P_1, P_2 have the same order, the adjacency list of w appears either in $P_1^{>uv}$ or $P_2^{<uv}$. So in particular, all of $P_2^{>uv} \cup P_2^{>w}$ is seen *after* (e, τ) is added to Q. Since u arrives before v, this means that $P_2^{>u} \cup P_2^{>v} \cup P_2^{>w}$ is seen after (e, τ) is added to Q.

Then, for any $f \in \tau$ and $\sigma \in L(e')$ such that $\tau <_f \sigma$, σ^{-f} appears after τ^{-f} in P_2, by the definition of $<_f$. Since τ^{-f} must be one of u, v, w, σ^{-f} is in $P_2^{>u} \cup P_2^{>v} \cup P_2^{>w}$, and it appears after (e, τ) is added to Q. Therefore,

$$H_{e,\tau} = |\{\tau' \in L(e) : \tau <_e \tau'\}|$$

can be counted with only $O(|Q| \log n)$ space.

Therefore, the space complexity of this algorithm is:

$$O\left((|Q| + |S|) \log n\right) = O(m' \log n)$$

3.3.2 Correctness.

Lemma 3.1.

$$\mathbb{E}\left[\widetilde{T}'\right] = T$$

Proof. Fix any ordering of the stream. Condition on the set S of edges sampled from P_1. Then, for each triangle $\tau \in G$ such that $e \in S$ and $\rho(\tau) = e$, $(e, \tau) \in Q$ with probability $\frac{m'}{T'} = \frac{m}{kT'}$. Therefore,

$$\mathbb{E}\left[\widetilde{T}' \middle| S\right] = k \sum_{e \in S} \widetilde{T}_e$$

and as each $e \in G$ is included in S with probability $\frac{m'}{m} = 1/k$,

$$\mathbb{E}\left[\widetilde{T}'\right] = \sum_{e \in G} \widetilde{T}_e$$

and since for each τ there is exactly one $e \in \tau$ such that $\rho(\tau) = e$:

$$\mathbb{E}\left[\widetilde{T}'\right] = T$$

□

To bound the accuracy of this estimator, we will need the following combinatorial lemma concerning the values $\widetilde{T}_e$.

Lemma 3.2.

$$\sum_{e \in G} \widetilde{T}_e^2 = O\left(T^{4/3}\right)$$

Proof. For all $i \in \{0, \ldots, \lfloor \log T \rfloor\}$, let $E_i = \{e \in G : \widetilde{T}_e \in [2^i, 2^{i+1}]\}$, and let $A_i = \{\tau \in G : \rho(\tau) \in E_i\}$.

Then, let $B_i = \{\tau \in A_i : H_{\rho(\tau),\tau} \geq \widetilde{T}_{\rho(\tau)}/2\}$, and let $F_i = \bigcup B_i$, so F_i contains all of the edges of triangles in B_i. Then, for any $e \in E_i$, at least half of the triangles τ such that $\rho(\tau) = e$ have $H_{\rho(\tau),\tau} \geq \widetilde{T}_{\rho(\tau)}/2$, so $|B_i| \geq |A_i|/2$. Furthermore, by [15], any graph with m edges has at most $m^{3/2}$ triangles, and so $|F_i|^{3/2} \geq |A_i|/2$. Then,

$$\sum_{e \in G} \widetilde{T}_e^2 \leq \sum_{i=0}^{\lfloor \log T \rfloor} 2^{i+1} \sum_{e \in E_i} \widetilde{T}_e$$

While

$$\begin{aligned}\sum_{e \in E_i} \widetilde{T}_e &= |A_i| \\ &\leq 2|F_i|^{3/2}\end{aligned}$$

Then, for any $e \in F_i$, $\exists \tau \in B_i$ such that $e \in \tau$, and therefore $H_{e,\tau} \geq H_{\rho(\tau),\tau} \geq \widetilde{T}_{\rho(\tau)}/2$. Therefore:

$$\begin{aligned}T_e &\geq H_{e,\tau} \\ &\geq \widetilde{T}_{\rho(\tau)}/2 \\ &\geq 2^{i-1}\end{aligned}$$

Thus, $|F_i|2^{i-1} \leq 3T$, and so $|F_i|^{3/2} \leq T^{3/2}2^{-3(i-1)/2}/3^{3/2}$. At the same time, $\sum_{e \in E_i} \widetilde{T}_e$ is clearly at most T, and so by breaking our sum at $i = \lfloor \log T/3 \rfloor$:

$$\begin{aligned}\sum_{e \in G} \widetilde{T}_e^2 &\leq \sum_{i=0}^{\lfloor \log(T)/3 \rfloor} 2^i T + \sum_{i=\lfloor \log(T)/3 \rfloor}^{\lfloor \log T \rfloor} 2^{-i/2+5/2} T^{3/2}/3^{3/2} \\ &= O\left(T^{4/3}\right) + O\left(T^{4/3}\right) \\ &= O\left(T^{4/3}\right)\end{aligned}$$

□

We are now ready to analyze the accuracy of the algorithm. First, we will show that $k \sum_{e \in S} \widetilde{T}$ is a good estimator for T.

Lemma 3.3.

$$\forall \varepsilon > 0, \mathbb{P}\left[k \sum_{e \in S} \widetilde{T}_e \in ((1-\varepsilon)T, (1+\varepsilon)T)\right] \geq 1 - O\left(\frac{k}{\varepsilon^2 T^{2/3}}\right)$$

Proof. As each $e \in G$ is included in S with probability $m'/m = 1/k$,

$$\mathbb{E}\left[k \sum_{e \in S} \widetilde{T}_e\right] = T$$

We will now proceed to bound the variance. Let I_e be the indicator variable that is 1 if $e \in S$ and 0 otherwise. Then, since a fixed number of edges are included in S, the family $\{I_e\}_{e \in G}$ is negatively associated, and so

$$\begin{aligned}\mathrm{Var}\left(k \sum_{e \in S} \widetilde{T}_e\right) &= k^2 \sum_{e \in G} \widetilde{T}_e^2 \,\mathrm{Var}\,(I_e) \\ &\leq k^2 \sum_{e \in G} \widetilde{T}_e^2 (1/k - 1/k^2) \\ &\leq k \sum_{e \in G} \widetilde{T}_e^2 \\ &\leq kT^{4/3}\end{aligned}$$

by applying Lemma 3.2. The result then follows by Chebyshev's inequality. □

We will now bound, for any sample set S, the accuracy with which the algorithm estimates $k \sum_{e \in S} \widetilde{T}_e$.

Lemma 3.4.

$$\begin{aligned}\forall \varepsilon > 0, \mathbb{P}&\left[\widetilde{T}' \in \left(k \sum_{e \in S} \widetilde{T}_e - \varepsilon T, k \sum_{e \in S} \widetilde{T}_e + \varepsilon T\right) \middle| S\right] \\ &\geq 1 - \frac{k^3 T'}{\varepsilon^2 T^2 m} \sum_{e \in S} \widetilde{T}_e\end{aligned}$$

Proof. We may assume $|Q| = m'$, since otherwise $Q = \bigcup_{e \in S}(e, L(e))$ and so $\widetilde{T}' = k \sum_{e \in S} \widetilde{T}_e$ exactly. Conditioned on S, for any τ such that $\rho(\tau) \in S$, let J_τ be 1 if $(\rho(\tau), \tau) \in Q$, and 0 otherwise. As Q is fixed size, the family $\{J_\tau\}_{\tau \in G}$ is negatively associated, and so

$$\begin{aligned}\mathrm{Var}\left(\widetilde{T}' \middle| S\right) &\leq \left(\frac{kT'}{m'}\right)^2 \sum_{e \in S} \sum_{\tau : \rho(\tau) = e} \mathrm{Var}(J_\tau) \\ &= \left(\frac{kT'}{m'}\right)^2 \sum_{e \in S} \sum_{\tau : \rho(\tau) = e} \left(\frac{m'}{T'} - \left(\frac{m'}{T'}\right)^2\right) \\ &\leq \frac{k^2 T'}{m'} \sum_{e \in S} \widetilde{T}_e \\ &\leq \frac{k^3 T'}{m} \sum_{e \in S} \widetilde{T}_e\end{aligned}$$

The result then follows from Chebyshev's inequality. □

To make use of the bound above, we will need to bound the probability that T' is too large.

Lemma 3.5.

$$\mathbb{P}\left[kT' \leq 30T\right] \geq 9/10$$

Proof. Each edge in G is included in S with probability $1/k$, so

$$\begin{aligned}\mathbb{E}\left[T'\right] &= \sum_{e \in G} T_e/k \\ &= 3T/k\end{aligned}$$

The result then follows from Markov's inequality. □

Putting these results together will allow us to bound the accuracy of the estimator.

Lemma 3.6. *There exists a constant $D > 0$ such that, $\forall \varepsilon \in (0,1)$, if*

$$k \leq \varepsilon^2 D T^{2/3}$$

then

$$\mathbb{P}\left[\widetilde{T}' \in ((1-\varepsilon)T, (1+\varepsilon)T)\right] \geq 2/3$$

Proof. Using Lemmas 3.3 and 3.4, choose $D \leq 1$ such that

$$\mathbb{P}\left[k\sum_{e\in S}\widetilde{T}_e \in ((1-\varepsilon/2)T, (1+\varepsilon/2)T)\right] \geq 99/100$$

$$\mathbb{P}\left[\widetilde{T}' \in \left(k\sum_{e\in S}\widetilde{T}_e - \varepsilon T/2, k\sum_{e\in S}\widetilde{T}_e + \varepsilon T/2\right)\middle| S\right]$$
$$\geq 1 - \frac{kT'}{3000T^{4/3}m}k\sum_{e\in S}\widetilde{T}_e$$

Then, if $\mathcal{E}$ is the event that $k\sum_{e\in S}\widetilde{T}_e = (1\pm\varepsilon/2)T$ and $kT' \leq 30T$, by Lemma 3.5 and the union bound

$$\mathbb{P}[\mathcal{E}] \geq 89/100$$

Thus,

$$\mathbb{P}\left[\widetilde{T}' \in \left(k\sum_{e\in S}\widetilde{T}_e - \varepsilon T/2, k\sum_{e\in S}\widetilde{T}_e + \varepsilon T/2\right)\middle| S, \mathcal{E}\right]$$
$$\geq 1 - \frac{kT'}{2670T^{4/3}m}k\sum_{e\in S}\widetilde{T}_e$$
$$\geq 1 - \frac{1}{89T^{1/3}m}k\sum_{e\in S}\widetilde{T}_e$$
$$\geq 1 - \frac{1+\varepsilon}{89m}T^{2/3}$$
$$\geq \frac{87}{89}$$

as $T \leq m^{3/2}$. Therefore, with probability at least $89/100 \times 87/89 - 1/100 > 2/3$

$$\widetilde{T}' \in \left(k\sum_{e\in S}\widetilde{T}_e - \varepsilon T/2, k\sum_{e\in S}\widetilde{T}_e + \varepsilon T/2\right)$$

and

$$k\sum_{e\in S}\widetilde{T}_e \in ((1-\varepsilon/2)T, (1+\varepsilon/2)T)$$

Therefore,

$$\widetilde{T}' = ((1-\varepsilon)T, (1+\varepsilon)T)$$

□

Theorem 3.7. *For all $\varepsilon, \delta \in (0,1)$, there is a 2-pass triangle counting algorithm that uses*

$$O\left(\frac{m\log n}{\varepsilon^2 T^{2/3}}\log 1/\delta\right)$$

space and returns a $(1\pm\varepsilon)$ multiplicative approximation to T with probability $1-\delta$.

Proof. Let $m' = \Theta\left(m/(\varepsilon^2 T^{2/3})\right)$, with constants chosen so that $k = \Theta(\varepsilon^2 T^{2/3})$ meets the requirements of Lemma 3.6. This gives an algorithm that runs in

$$O\left(\frac{m\log n}{\varepsilon^2 T^{2/3}}\right)$$

space and returns a $(1\pm\varepsilon)$ multiplicative approximation to T with probability 2/3. Then, for an appropriately chosen constant $D' \in \mathbb{N}$, run $D'\log 1/\delta$ copies of the algorithm in parallel, and take the median of their outputs. If D' is chosen to be a sufficiently large constant, at least half the algorithms will return a result within εT of T with probability $1-\delta$, and so the median will give a $(1\pm\varepsilon)$ multiplicative approximation. □

4 TWO PASS 4-CYCLE COUNTING ALGORITHM

4.1 Notation

For any edge or wedge x, let T_x denote the number of 4-cycles that contain x, let T denote the number of 4-cycles in our graph G, and let m denote the number of edges in G.

4.2 Algorithm

Our algorithm will take two passes over the stream. Call these passes P_1, P_2. We will *not* require P_2 to have the same ordering as P_1. Our algorithm is then as follows:

(1) Choose a sample size m'.
(2) While passing through P_1, keep a uniformly chosen size-m' subset S of $E(G)$. If $m' > m$, instead keep all of $E(G)$. Record m.
(3) Let Q be the set of wedges consisting of edges in S.
(4) While passing through P_2, for each $w \in Q$, calculate T_w.
(5) Let $k = m/m'$.
(6) Return $T' = k^2 \cdot \sum_{w\in Q} T_w$.

4.3 Analysis

4.3.1 Space Complexity. For any wedge uvw in Q, and any vertex $z \in V(G)$, we may check whether $uvwz$ forms a 4-cycle in G while passing over the adjacency list of z with only extra $2|S|$ bits of storage, by flagging whether u and w are in the adjacency list of z. Therefore, the algorithm can be implemented with the use of

$$O(|S|\log n) = O(m'\log n)$$

space.

4.3.2 Correctness. To show that the algorithm obtains an $O(1)$ factor approximation to T, we will need to show that at least a constant fraction of cycles in our graph G are easy to find.

Definition 4.1. *We call an edge $e \in E(G)$ "heavy" if it is contained in at least $40\sqrt{T}$ 4-cycles, and "light" otherwise. We call a wedge w "overused" if it is contained in at least $40T^{1/4}$ 4-cycles, "heavy" if it contains a heavy edge, "bad" if it is either overused or heavy, and "good" otherwise. We call a cycle "good" if it contains at least one good wedge. We will denote the set of good cycles as F_G.*

Lemma 4.2. $|F_G| = \Omega(T)$

Proof. See appendix. □

For each cycle τ that has at least one good wedge, let $\rho(\tau)$ denote an arbitrarily chosen good wedge in τ. Let f_G denote the number of 4-cycles τ such that $\rho(\tau)$ is in the sampled wedge set Q. Let f_B denote the number of 4-cycles τ such that $\rho(\tau) \notin Q$, but some other wedge in τ is in Q. The estimate returned by the algorithm is then $k^2(f_G + f_B)$.

Lemma 4.3. *There exists a constant $D > 0$ such that, if*

$$k \leq DT^{3/8}$$

then $\mathbb{P}\left[k^2 f_G \in [|F_G|/4, 2|F_G|]\right] \geq 9/10$.

Proof. For each $\tau \in F_G$, there is a

$$\frac{m'(m'-1)}{m(m-1)} \in [1/(2k^2), 1/k^2]$$

probability that both edges in $\rho(\tau)$ will be included in S, so

$$\mathbb{E}\left[k^2 f_G\right] \in [|F_G|/2, |F_G|]$$

Next we bound the variance. For any two 4-cycles τ_1, τ_2, the probability that both $\rho(\tau_1)$ and $\rho(\tau_2)$ are in Q is $\prod_{i=0}^{r-1} \frac{m'-i}{m-i} \leq k^{-r}$, where r is the total number of distinct edges between $\rho(\tau_1)$ and $\rho(\tau_2)$ (varying from 4 if they are disjoint to 2 if they are the same wedge).

$$\begin{aligned}
\mathrm{Var}(k^2 f_G) &= k^4 \mathbb{E}\left[f_G^2\right] - k^4 \mathbb{E}\left[f_G\right]^2 \\
&\leq k^4 \sum_{\tau_1, \tau_2 \in F_G} \mathbb{P}\left[\rho(\tau_1), \rho(\tau_2) \in Q\right] - k^4 \mathbb{E}\left[f_G\right]^2 \\
&\leq k^4 \sum_{\substack{\tau_1, \tau_2 \in F_G: \\ \rho(\tau_1) \cap \rho(\tau_2) \neq \emptyset}} \mathbb{P}\left[\rho(\tau_1), \rho(\tau_2) \in Q\right] \\
&\leq k^4 \left(\sum_{\tau \in F_G} \left(k^{-2} T_{\rho(\tau)} + k^{-3} \sum_{e \in \rho(\tau)} T_e \right) \right) \\
&\leq \sum_{\tau \in F_G} \left(40k^2 T^{1/4} + 80kT^{1/2} \right) \\
&\leq 120DT^2
\end{aligned}$$

where the penultimate step uses the fact that $\rho(\tau)$ is good and the last step uses the fact that $k \leq DT^{3/8}$. The result then follows for sufficiently small $D > 0$ by Lemma 4.2 and Chebyshev's inequality. □

Lemma 4.4.

$$\mathbb{P}\left[k^2 f_B \leq 40T\right] \geq 9/10$$

Proof. Each cycle has 4 wedges, and each one has at most a $1/k^2$ probability of contributing to f_B, so $\mathbb{E}\left[k^2 f_B\right] \leq 4T$. The result then follows by Markov's inequality. □

Lemma 4.5. *There exists a constant $D > 0$ such that, if*

$$k \leq DT^{3/8}$$

then with probability $4/5$, the algorithm outputs an $O(1)$ multiplicative approximation to T.

Proof. Apply the union bound to the above two lemmas. □

Theorem 4.6. *For every $\delta \in (0, 1)$, there exists a two-pass 4-cycle counting algorithm that uses*

$$O\left(\frac{m \log n}{T^{3/8}} \log 1/\delta\right)$$

space and returns a $O(1)$ multiplicative approximation to T with probability $1 - \delta$.

Proof. Let $m' = \Theta(m/T^{3/8})$, with constants chosen so that $k = \Theta(T^{3/8})$ meets the requirement of Lemma 4.5. This gives an algorithm that runs in $O((m \log n)/T^{3/8})$ space and returns an $O(1)$ multiplicative approximation to T with probability $2/3$. Running $\Theta(\log 1/\delta)$ copies of the algorithm in parallel and taking the median of their outputs then gives the desired result. □

5 LOWER BOUNDS

We will make use of reductions from the following communication complexity problems. In each case, the players are allowed to use (shared) randomness, and the lower bounds mentioned will be for solving the problem with probability $2/3$.

Index (INDEX_r)
Alice holds a binary string s of length r and Bob holds an index $x \in [r]$. Alice sends a message to Bob, who must compute s_x. This requires $\Omega(r)$ communication[22].

Three Party NOF Pointer-Jumping (3-PJ$_r$)
Alice, Bob, and Charlie share edges from a graph with four layers of vertices: $V_1 = \{v^*\}$, $V_2 = \{v_{2i}\}_{i=1}^r$, $V_3 = \{v_{3i}\}_{i=1}^r$, $V_4 = \{v_{40}, v_{41}\}$ and three layers of edges E_1, E_2, E_3, where E_i contains edges from V_i to V_{i+1}. Edges of the graph are directed, and every vertex in layers 1 through 3 has out-degree exactly one. Vertices in V_4 have out-degree 0. Alice has the edges in E_2 and E_3, Bob has E_1 and E_3, and Charlie has E_1 and E_2. Using one way communication — Alice sends a message to Bob, who then sends a message to Charlie — the players must compute whether v^* is connected by a directed path to v_{40} or v_{41}, answering 0 in the first case and 1 in the second. The best known lower bound on the communication complexity of this problem is $\Omega(\sqrt{r})$ [34], while the best known upper bound is $\widetilde{O}(r \log\log r / \log r)$ [11], and it is conjectured that the true complexity of the problem is close to linear.

Two Party Disjointness (DISJ$_r$)
Alice holds a binary string s^1 of length r and Bob holds a string s^2 of the same length. Using multi-way communication, they must determine whether there exists an index x such that $s^1_x = s^2_x = 1$, answering 1 if there is and 0 otherwise. The communication complexity of this problem is $\Omega(r)$ [19, 29].

Three Party NOF Disjointness (3-DISJ$_r$)
Alice, Bob, and Charlie share three binary strings of length r: s^1, s^2, and s^3. Each of the three players holds two of the strings: Alice has s^1 and s^2, Bob s^2 and s^3, Charlie s^3 and s^1. Using multi-way communication, they must determine whether there exists an index x such that $s^1_x = s^2_x = s^3_x = 1$, answering 1 if there is and 0 otherwise. The best known lower bound on the communication complexity of this problem is $\Omega(\sqrt{r})$ [30]. No sublinear protocol is known, and it is conjectured that the true complexity is $\widetilde{\Omega}(r)$.

5.1 Reduction Structure

Each of the lower bounds will take the following form: an encoding of an instance of the problem into a graph where the vertices are "assigned" to players. Each player must be able to insert the adjacency list corresponding to their "assigned" vertices, and thus any edges between the assigned vertices of two different players must be determined by state shared by both players.

In each case, we will embed an instance of the problem in a graph which has no ℓ-cycles (for ℓ the length we are considering) if the correct output of the game is 0, and T cycles if it is 1. Therefore, any algorithm that can distinguish between 0 and T ℓ-cycles (in particular, any algorithm for counting ℓ-cycles) would provide a protocol for the communication problem, with communication complexity equal to the space cost of the algorithm. For one-pass lower bounds, we will reduce from one-way communication problems, and for multi-pass lower bounds we will reduce from multi-way communication problems, so that each pass over the input corresponds to one round of communication.

5.2 Girth-6 Graphs

In two of our reductions concerning counting 4-cycles we will need to construct a graph with a large number of 4-cycles for instances with output 1 and a similar 4-cycle-free graph for 0-instances. To achieve this, we employ bipartite 4-cycle-free graphs on $2r$ vertices with $\Theta(r^{3/2})$ edges. An example of such a graph is an incidence graph of a special kind of projective plane called field plane. If the order of the plane is q, it has $q^2 + q + 1$ points, the same number of lines, and each line passes through exactly $q + 1$ points. Thus, the incidence graph has $2(q^2 + q + 1)$ vertices, each with degree $q + 1$. The graph is 4-cycle-free since by the definition of a projective plane, for any two distinct points, there is exactly one line passing through both of them, and for any two distinct lines, there is exactly one point both of them pass through. We also note that there are no 4-cycle-free graphs on r vertices with $\omega(r^{3/2})$ edges [8].

5.3 One-Pass Triangle Counting

Theorem 5.1. *If 3-PJ$_r$ requires $\Omega(f_{pj}(r))$ communication, then for any m and $T \le m$, there exist $m' = \Theta(m)$ and $T' = \Theta(T)$ such that any adjacency list streaming algorithm that distinguishes between m'-edge graphs with 0 and T' triangles with at least 2/3 probability in one pass requires $\Omega(f_{pj}(m/\sqrt{T}))$ space.*

Proof. We will describe an encoding of an instance of 3-PJ$_r$ as a graph G, illustrated in Figure 1a. Let $r, k \in \mathbb{N}$ be variables to be fixed later, and consider an instance of 3-PJ$_r$ with edge sets E_1, E_2, E_3. $V(G)$ will be the union of the following sets:

- $A = \{a_i\}_{i=1}^r$ of size r, assigned to Alice.
- B of size k, assigned to Bob.
- $C_1, C_2, \ldots, C_r$, each of size k, assigned to Charlie.

The edges of the graph will be:

- For the edge $(v^*, v_{2i}) \in E_1$, k^2 edges connecting every vertex in B to every vertex in C_i.
- For each edge $(v_i, v_{3j}) \in E_2$, k edges connecting every vertex in C_i to a_j.
- For each edge $(v_i, v_{41}) \in E_3$, k edges connecting a_i to every vertex in B. Edges $(v_i, v_{40}) \in E_3$ will be ignored.

G will have $O(rk + k^2)$ edges. If v^* has a path to v_{41} it will have k^2 triangles. Otherwise it will have 0 triangles. We can

therefore set $k = \Theta(\sqrt{T})$ and $r = \Theta(m/\sqrt{T})$ to complete the proof. □

5.4 $O(1)$-Pass Triangle Counting

Theorem 5.2. *If 3-DISJ$_r$ requires $\Omega(f_d(r))$ communication, then for any m and $T \leq m^{3/2}$, there exist $m' = \Theta(m)$ and $T' = \Theta(T)$ such that any adjacency list streaming algorithm that distinguishes between m'-edge graphs with 0 and T' triangles with at least 2/3 probability in a constant number of passes requires $\Omega(f_d(m/T^{2/3}))$ space.*

Proof. We will describe an encoding of an instance of 3-DISJ$_r$ as a graph G, illustrated in Figure 1b. $V(G)$ will be the union of the following sets:

- $A_1, A_2, \ldots, A_r$, each of size k, assigned to Alice.
- $B_1, B_2, \ldots, B_r$, each of size k, assigned to Bob.
- $C_1, C_2, \ldots, C_r$, each of size k, assigned to Charlie.

The edges of the graph will be:

- For each $i \in [r]$, k^2 edges between A_i and C_i iff $s_i^1 = 1$.
- For each $i \in [r]$, k^2 edges between A_i and B_i iff $s_i^2 = 1$.
- For each $i \in [r]$, k^2 edges between B_i and C_i iff $s_i^3 = 1$.

G will have $\Theta(rk)$ vertices and $O(rk^2)$ edges. If there exists an index x such that $s_x^1 = s_x^2 = s_x^3 = 1$, A_x, then B_x, and C_x will form k^3 triangles. Otherwise, the graph will be triangle-free. Therefore, for any m and $T \leq m^{3/2}$, we may set $k = \Theta(T^{1/3})$ and $r = m/T^{2/3}$, and the result follows. □

5.5 One Pass 4-Cycle Counting

Theorem 5.3. *For any m and $T \leq m^{1/3}$, there exists $m' = m$ such that any adjacency list streaming algorithm that distinguishes between m'-edge graphs with 0 and T 4-cycles with at least 2/3 probability in one pass requires $\Omega(m)$ space.*

Proof. We will describe an encoding of an instance of INDEX$_{\Theta(r^{3/2})}$ as a graph G, illustrated in Figure 1c. $V(G)$ will be the union of the following sets:

- $A = \{a_i\}_{i=1}^r$ and $B = \{b_i\}_{i=1}^r$, each of size r, assigned to Alice.
- $C_1, C_2, \ldots, C_r, D_1, D_2, \ldots, D_r$, each of size k, assigned to Bob.

Using the construction in Section 5.2, fix a bipartite 4-cycle-free graph H with both partitions of size r and $\Theta(r^{3/2})$ edges. Let the size of Alice's string (and therefore the size of the instance) be $|E(H)|$, and associate each edge of H with a different index of Alice's string. The edges of our graph are then the following:

- A copy of H between A and B, with the edges corresponding to bits of Alice's string that are 0 removed.
- A matching of size k between C_i and D_j, where ij is the edge of H corresponding to Bob's index.
- For each $i \in [r]$, k edges between a_i and C_i.
- For each $i \in [r]$, k edges between b_i and D_i.

The resulting graph has $\Theta(rk)$ vertices and $O(r^{3/2}+rk)$ edges. If $s_x = 1$, the graph has k 4-cycles, otherwise it is 4-cycle-free. Therefore, by setting $k = T$ and $r = \Theta(m^{2/3})$, the result follows. □

5.6 $O(1)$-Pass 4-Cycle Counting

Theorem 5.4. *For any m and $T \leq m^{3/5}$, there exists $m' = \Theta(m)$ such that any any adjacency list streaming algorithm that distinguishes between m'-edge graphs with 0 and T 4-cycles with at least 2/3 probability in a constant number of passes requires $\Omega(m/T^{2/3})$ space.*

Proof. We will describe an encoding of an instance of DISJ$_{\Theta(r^{3/2})}$ as a graph G, illustrated in Figure 1d.

Using the construction in Section 5.2, fix a bipartite 4-cycle-free graph H_1 with both partitions of size r and $\Theta(r^{3/2})$ edges and another bipartite 4-cycle-free graph H_2 with both partitions of size k and $\Theta(k^{3/2})$ edges. Let the size of Alice and Bob's strings (and therefore the size of the instance) be $|E(H_1)|$, and associate each edge of H_1 with an index i from 1 to $|E(H_1)|$, so each edge in H_1 corresponds to bits s_i^1 and s_i^2 in Alice and Bob's strings.

$V(G)$ will be the union of the following sets:

- Sets $A_1, A_2, \ldots A_r$ and $B_1, B_2, \ldots B_r$, each of size k, assigned to Alice.
- Sets $C_1, C_2, \ldots C_r$ and $D_1, D_2, \ldots D_r$, each of size k, assigned to Bob.

The edges of the graph will be:

- A copy of H_2 between A_i and C_i for all $i \in [r]$
- A copy of H_2 between B_i and D_i for all $i \in [r]$.
- For each edge (v_i, u_j) in H_1, a matching of size k between A_i and B_j iff corresponding Alice's bit is 1.
- For each edge (v_i, u_j) in H_1, a matching of size k between C_i and D_j iff corresponding Bob's bit is 1.

G will have $\Theta(rk)$ vertices and $\Theta(rk^{3/2} + kr^{3/2})$ edges. If there exists an index x such that $s_x^1 = s_x^2 = 1$, G will have $k^{3/2}$ 4-cycles, and otherwise it will be 4-cycle-free. Therefore, the result follows by setting $k = T^{2/3}$ and $r^{3/2} = \Theta(m/T^{2/3})$, so that as $T \leq m^{3/5}$, $rk^{3/2} + kr^{3/2} = \Theta(m^{2/3}T^{5/9} + m) = \Theta(m)$. □

5.7 $O(1)$-Pass ℓ-cycle Counting for $\ell \geq 5$

Theorem 5.5. *For any constant $\ell \geq 5$, and for any m and $T \leq m$, there exists $m' = \Theta(m)$ such that any adjacency list streaming algorithm that distinguishes between m'-edge graphs with 0 and T ℓ-cycles with at least 2/3 probability in a constant number of passes requires $\Omega(m)$ space.*

Proof. We will describe an encoding of an instance of DISJ$_k$ as a graph G, illustrated in Figure 1e. $V(G)$ will be the union of the following sets:

- $A = \{a_i\}_{i=1}^{r+1}$ of size $r+1$, assigned to Alice.
- $B = \{b_i\}_{i=1}^{r}$ of size r, assigned to Bob.
- $C = \{c_i\}_{i=1}^{k}$ of size k, assigned to Bob.
- $D = \{d_i\}_{i=1}^{\ell-4}$ of size $\ell - 4$, assigned to Bob.

The edges of the graph will be:

- (a_i, b_i) for all $i \in [r]$.
- (a_{r+1}, c_i) for all $i \in [k]$.
- $(d_{\ell-4}, c_i)$ for all $i \in [k]$.
- A path $d_1 - d_2 - \cdots - d_{\ell-4}$. Note that for $\ell = 5$, this path has 1 vertex and 0 edges.
- (a_i, a_{r+1}) for every i such that the i-th bit of Alice's string is 1.
- (b_i, d_1) for every i such that the i-th bit of Bob's string is 1.

G will have $O(r+k)$ edges, as ℓ is a constant. If there exists an index x such that $s_x^1 = s_x^2 = 1$, then the graph will have k ℓ-cycles, otherwise it will be ℓ-cycle-free. Therefore, if $T \leq m$, the result follows by setting $r = m, k = T$. □

ACKNOWLEDGMENTS

This work was done in part while John Kallaugher, Eric Price, and Sofya Vorotnikova were visiting the Simons Institute for the Theory of Computing. Sofya Vorotnikova was supported by a Simons-Berkeley Research Fellowship. This material is based upon work supported by the National Science Foundation under Grant No. 1637536 and Grant No. 1751040.

REFERENCES

[1] Lars Arge, Michael T. Goodrich, and Nodari Sitchinava. 2010. Parallel external memory graph algorithms. In *24th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2010, Atlanta, Georgia, USA, 19-23 April 2010 - Conference Proceedings*. 1–11. https://doi.org/10.1109/IPDPS.2010.5470440

[2] Shaikh Arifuzzaman, Maleq Khan, and Madhav V. Marathe. 2013. PATRIC: a parallel algorithm for counting triangles in massive networks. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*. 529–538. https://doi.org/10.1145/2505515.2505545

[3] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. 2002. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA*. 623–632. http://dl.acm.org/citation.cfm?id=545381.545464

[4] Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. 2008. Efficient Semi-streaming Algorithms for Local Triangle Counting in Massive Graphs. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*. ACM, New York, NY, USA, 16–24. https://doi.org/10.1145/1401890.1401898

[5] 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) (Leibniz International Proceedings in Informatics (LIPIcs))*, Heribert Vollmer and Brigitte Vallée (Eds.), Vol. 66. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 11:1–11:14. https://doi.org/10.4230/LIPIcs.STACS.2017.11

[6] Jonathan W. Berry, Bruce Hendrickson, Simon Kahan, and Petr Konecny. 2007. Software and Algorithms for Graph Queries on Multithreaded Architectures. In *21th International Parallel and Distributed Processing Symposium (IPDPS 2007), Proceedings, 26-30 March 2007, Long Beach, California, USA*. 1–14. https://doi.org/10.1109/IPDPS.2007.370685

[7] Jonathan W. Berry, Bruce Hendrickson, Randall A. LaViolette, and Cynthia A. Phillips. 2011. Tolerating the community detection resolution limit with edge weighting. *Phys. Rev. E* 83 (May 2011), 056119. Issue 5. https://doi.org/10.1103/PhysRevE.83.056119

[8] J. Bondy and M. Simonovits. 1974. Cycles of even length in graphs. *Journal of Combinatorial Theory, Series B* (1974), 97–105.

[9] Vladimir Braverman, Rafail Ostrovsky, and Dan Vilenchik. 2013. How Hard Is Counting Triangles in the Streaming Model?. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*. 244–254. https://doi.org/10.1007/978-3-642-39206-1_21

[10] Joshua Brody and Amit Chakrabarti. 2008. Sublinear communication protocols for multi-party pointer jumping and a related lower bound. *arXiv preprint arXiv:0802.2843* (2008).

[11] Joshua Brody and Mario Sanchez. 2015. Dependent Random Graphs and Multiparty Pointer Jumping. *CoRR* abs/1506.01083 (2015). arXiv:1506.01083

[12] Luciana S. Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. 2006. Counting triangles in data streams. In *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*. 253–262. https://doi.org/10.1145/1142351.1142388

[13] Graham Cormode and Hossein Jowhari. 2014. A second look at counting triangles in graph streams. *Theor. Comput. Sci.* 552 (2014), 44–51. https://doi.org/10.1016/j.tcs.2014.07.025

[14] Jean-Pierre Eckmann and Elisha Moses. 2002. Curvature of co-links uncovers hidden thematic layers in the World Wide Web. *Proceedings of the National Academy of Sciences* 99, 9 (2002), 5825–5829. https://doi.org/10.1073/pnas.032093399

[15] emab (http://math.stackexchange.com/users/74964/emab). 2014. Number of triangles in a graph based on number of edges. Mathematics Stack Exchange. URL:http://math.stackexchange.com/q/823650 (version: 2014-06-07).

[16] David GarcÃŋa-Soriano and Konstantin Kutzkov. [n. d.]. Triangle counting in streamed graphs via small vertex covers. In *Proceedings of the 2014 SIAM International Conference on Data Mining*. 352–360. https://doi.org/10.1137/1.9781611973440.40

[17] Hossein Jowhari and Mohammad Ghodsi. 2005. New Streaming Algorithms for Counting Triangles in Graphs. In *Computing and Combinatorics, 11th Annual International Conference, COCOON 2005, Kunming, China, August 16-29, 2005, Proceedings*. 710–716. https://doi.org/10.1007/11533719_72

[18] 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 (SODA '17)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1778–1797. http://dl.acm.org/citation.cfm?id=3039686.3039802

[19] B. Kalyanasundaram and G. Schintger. 1992. The Probabilistic Communication Complexity of Set Intersection. *SIAM Journal on Discrete Mathematics* 5, 4 (1992), 545–557. https://doi.org/10.1137/0405044

[20] Daniel M. Kane, Kurt Mehlhorn, Thomas Sauerwald, and He Sun. 2012. Counting Arbitrary Subgraphs in Data Streams. In *Proceedings of the 39th International Colloquium Conference on Automata, Languages, and Programming - Volume Part II (ICALP'12)*. Springer-Verlag, Berlin, Heidelberg, 598–609. https://doi.org/10.1007/978-3-642-31585-5_53

[21] Mihail N. Kolountzakis, Gary L. Miller, Richard Peng, and Charalampos E. Tsourakakis. 2012. Efficient Triangle Counting in Large Graphs via Degree-Based Vertex Partitioning. *Internet Mathematics* 8, 1-2 (2012), 161–185. https://doi.org/10.1080/15427951.2012.625260

[22] Ilan Kremer, Noam Nisan, and Dana Ron. 1995. On Randomized One-round Communication Complexity. In *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing (STOC '95)*. ACM, New York, NY, USA, 596–605. https://doi.org/10.1145/225058.225277

[23] Konstantin Kutzkov and Rasmus Pagh. 2014. Triangle Counting in Dynamic Graph Streams. In *Algorithm Theory - SWAT 2014 - 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July 2-4, 2014. Proceedings*. 306–318. https://doi.org/10.1007/978-3-319-08404-6_27

[24] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. 2008. Microscopic evolution of social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*. 462–470. https://doi.org/10.1145/1401890.1401948

[25] Madhusudan Manjunath, Kurt Mehlhorn, Konstantinos Panagiotou, and He Sun. 2011. Approximate Counting of Cycles in Streams. In *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*. 677–688. https://doi.org/10.1007/978-3-642-23719-5_57

[26] 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, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*. 401–411. https://doi.org/10.1145/2902251.2902283

[27] Rasmus Pagh and Charalampos E. Tsourakakis. 2012. Colorful triangle counting and a MapReduce implementation. *Inf. Process. Lett.* 112, 7 (2012), 277–281. https://doi.org/10.1016/j.ipl.2011.12.007

[28] A. Pavan, Kanat Tangwongsan, Srikanta Tirthapura, and Kun-Lung Wu. 2013. Counting and Sampling Triangles from a Graph Stream. *PVLDB* 6, 14 (2013), 1870–1881. http://www.vldb.org/pvldb/vol6/p1870-aduri.pdf

[29] A. A. Razborov. 1992. On the Distributional Complexity of Disjointness. *Theor. Comput. Sci.* 106, 2 (Dec. 1992), 385–390. https://doi.org/10.1016/0304-3975(92)90260-M

[30] Alexander A. Sherstov. 2014. Communication Lower Bounds Using Directional Derivatives. *J. ACM* 61, 6, Article 34 (Dec. 2014), 71 pages. https://doi.org/10.1145/2629334

[31] Alexander A Sherstov. 2016. The multiparty communication complexity of set disjointness. *SIAM J. Comput.* 45, 4 (2016), 1450–1489.

[32] Siddharth Suri and Sergei Vassilvitskii. 2011. Counting triangles and the curse of the last reducer. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*. 607–614. https://doi.org/10.1145/1963405.1963491

[33] Charalampos E. Tsourakakis, Mihail N. Kolountzakis, and Gary L. Miller. 2011. Triangle Sparsifiers. *J. Graph Algorithms Appl.* 15, 6 (2011), 703–726. https://doi.org/10.7155/jgaa.00245

[34] Emanuele Viola and Avi Wigderson. 2009. One-way multiparty communication lower bound for pointer jumping with applications. *Combinatorica* 29, 6 (01 Nov 2009), 719–743. https://doi.org/10.1007/s00493-009-2667-z

A PROOFS OMITTED FROM 4-CYCLE COUNTING

Definition 4.1. *We call an edge $e \in E(G)$ "heavy" if it is contained in at least $40\sqrt{T}$ 4-cycles, and "light" otherwise. We call a wedge w "overused" if it is contained in at least $40T^{1/4}$ 4-cycles, "heavy" if it contains a heavy edge, "bad" if it is either overused or heavy, and "good" otherwise. We call a cycle "good" if it contains at least one good wedge. We will denote the set of good cycles as F_G.*

Lemma 4.2. $|F_G| = \Omega(T)$

We split the proof of this into three subsidiary lemmas.

Lemma A.1. *There are at least $\frac{13}{50}T$ cycles containing no more than one heavy edge.*

Proof. Call a wedge "very heavy" if it contains two heavy edges. We note that there are at most $\sqrt{T}/10$ heavy edges, as each cycle contains 4 edges. Therefore, there are at most $T/100$ distinct pairs of disjoint heavy edges, and so there are at most $T/50$ cycles containing a pair of disjoint heavy edges, as each such pair may participate in at most two distinct cycles.

Therefore, the remaining $\frac{49}{50}T$ cycles all either contain no more than one heavy edge, or consist of one light wedge joined with one very heavy wedge. Let the number of cycles of the second type be B.

For each pair of vertices uv, let g_{uv} be the number of light wedges of the form uwv for some vertex w, and b_{uv} be the number of very heavy wedges of that form. Then we have:

$$B = \sum_{uv \in V(G)^2} g_{uv} b_{uv}$$

Furthermore,

$$\sum_{\substack{uv \in V(G)^2: \\ g_{uv} \leq 2}} g_{uv} b_{uv} \leq 2 \sum_{uv \in V(G)^2} b_{uv} \leq T/50$$

and so we have:

$$B \leq \sum_{\substack{uv \in V(G)^2: \\ g_{uv} > 2}} g_{uv} b_{uv} + T/50$$

Furthermore, between any pair of vertices there are at least $\binom{b_{uv}}{2}$ cycles consisting of two very heavy wedges with endpoints at those vertices, and at least $\binom{g_{uv}}{2}$ cycles consisting of two light wedges with endpoints at those vertices. Furthermore, each of those cycles will appear between at most two pairs of vertices in this fashion. Therefore:

$$
\begin{aligned}
T &\geq B + \frac{1}{2} \sum_{uv \in V(G)^2} \left(\binom{g_{uv}}{2} + \binom{b_{uv}}{2} \right) \\
&\geq B + \frac{1}{2} \sum_{\substack{uv \in V(G)^2: \\ g_{uv} > 2}} \left(\binom{g_{uv}}{2} + \binom{b_{uv}}{2} \right) \\
&= B + \frac{1}{4} \sum_{\substack{uv \in V(G)^2: \\ g_{uv} > 2}} \left(g_{uv}^2 - g_{uv} + b_{uv}^2 - b_{uv} \right) \\
&\geq B + \frac{1}{4} \sum_{\substack{uv \in V(G)^2: \\ g_{uv} > 2}} \left(\frac{2}{3} g_{uv}^2 + b_{uv}^2 \right) - T/400 \\
&\geq B + \frac{1}{\sqrt{6}} \sum_{\substack{uv \in V(G)^2: \\ g_{uv} > 2}} g_{uv} b_{uv} - T/400 \\
&\geq B + \frac{1}{\sqrt{6}} (B - T/50) - T/400
\end{aligned}
$$

where the last step plugs in the previous expression. Hence:

$$B \leq T(1 + \frac{1}{50\sqrt{6}} + \frac{1}{400})/(1 + \frac{1}{\sqrt{6}}) < 0.72T$$

So the number of cycles that contain no more than one heavy edge is at least $\frac{49}{50}T - 0.72T = \frac{13}{50}T$. □

For the next two lemmas, note that there are at most $4T/40T^{1/4} = T^{3/4}/10$ overused wedges, as each cycle contains 4 distinct wedges. We will distinguish between two kinds of vertices. Let a vertex $v \in V(G)$ be "primary" if it is an endpoint of at least $\sqrt{T}$ overused wedges, and "secondary" otherwise. There are at most $2 \cdot T^{3/4}/10 \cdot 1/\sqrt{T} = T^{1/4}/5$ primary vertices.

Lemma A.2. *There are at most $\frac{3}{25}T$ 4-cycles containing all overused wedges.*

Proof. Any 4-cycle that contains all overused wedges *and* at least one primary vertex can be uniquely specified by one primary vertex and an overused wedge that is disjoint from it. There are therefore at most

$$T^{1/4}/5 \cdot T^{3/4}/10 \leq T/50$$

such cycles. Now, for any pair of secondary vertices $u, v \in V(G)$, let X_{uv} be the number of 4-cycles that contain u and v as opposite vertices (that is, not connected by any edge in the cycle), and have only overused wedges. As both u and v are secondary, $X_{uv} \leq \sqrt{T}$. Note also that there are at least $\sqrt{X_{uv}}$ overused wedges with u and v as their endpoints. Then, the number of 4-cycles containing only secondary vertices and all overused wedges is at most:

$$
\begin{aligned}
\sum_{\substack{uv \in V(G)^2: \\ u,v \text{ secondary}}} X_{uv} &\leq T^{1/4} \sum_{\substack{uv \in V(G)^2: \\ u,v \text{ secondary}}} \sqrt{X_{uv}} \\
&\leq T^{1/4} \# \text{ of overused wedges} \\
&\leq T/10
\end{aligned}
$$

So the total number of cycles that contain all overused wedges is at most $\frac{1}{50}T + \frac{1}{10}T = \frac{3}{25}T$. □

Lemma A.3. *There are at most $\frac{3}{25}T$ 4-cycles containing one heavy edge e and with the two wedges not containing e being overused.*

Proof. First we bound the number of such 4-cycles where e has at least one primary endpoint, v. For any such cycle, v has an overused wedge that is disjoint from it, and v together with this wedge uniquely determine the cycle, so the number of such cycles is at most:

$$T^{1/4}/5 \cdot T^{3/4}/10 = T/50$$

Second, we bound the number of such 4-cycles where e has at least one secondary endpoint v. Each such cycle can be characterised by the choices of e and the choice of overused wedge that connects to e at v. As v is secondary, there are at most $\sqrt{T}$ choices of this overused wedge, so the total number of such cycles is at most:

$$\sqrt{T}/10 \cdot \sqrt{T} = T/10$$

So the total number of cycles that contain one bad edge e and with the two wedges not containing e being overused is at most $\frac{1}{50}T + \frac{1}{10}T = \frac{3}{25}T$. □

By combining the previous three lemmas, there are at least $\frac{13}{50}$ cycles that contain no more than one heavy edge, no more than $\frac{3}{25}T$ of which have all overused cycles, and no more than $\frac{3}{25}T$ of which have one heavy edge e and both wedges that do not contain e overused. So the number of cycles which contain at least one good wedge is at least:

$$\frac{13}{50}T - \frac{3}{25}T - \frac{3}{25}T = \frac{1}{50}T = \Omega(T) \ .$$