Triangle and Four Cycle Counting in the Data Stream Model

Andrew McGregor
mcgregor@cs.umass.edu
University of Massachusetts

Sofya Vorotnikova
svorotni@gmail.com
Dartmouth College

1 INTRODUCTION

The problem of estimating the number of cycles in a graph is one of the most widely studied graph problems in the data stream model. Three relevant variants of the data stream model include: the arbitrary order model in which the stream consists of the edges of the graph in arbitrary order, the random order model in which the edges are randomly permuted, and the adjacency list order model in which all edges incident to the same vertex appear consecutively. In this paper, we focus on the problem of triangle and 4-cycle counter in these models. We improve over the state-of-the-art results as follows where \( n \) is the number of vertices, \( m \) is the number of edges and \( T \) is a lower bound on the quantity being estimated:

- **Random Order Model**: We present a single-pass algorithm that \((1 + \epsilon)\)-approximates the number of triangles using \( O(\epsilon^{-2}m/\sqrt{T}) \) space and prove that this is optimal in the range \( T \leq \sqrt{m} \). The best previous result, a \((3+\epsilon)\)-approximation using \( O(\epsilon^{-4}m/\sqrt{T}) \) space, was presented by Cormode and Jowhari (Theor. Comput. Sci. 2017).
- **Adjacency List Model**: We present an algorithm that returns a \((1 + \epsilon)\)-approximation of the number of 4-cycles using two passes and \( O(\epsilon^{-4}m/\sqrt{T}) \) space. The best previous result, a constant approximation using \( O(m/T^{3/8}) \) space, was presented by Kallaugher et al. (PODS 2019). We also show that \((1 + \epsilon)\)-approximation in a single pass is possible in \( \polylog(n) \) space if \( T = \Omega(n^2) \) and \( O(n) \) space if \( T = \Omega(n) \).
- **Arbitrary Order Model**: We present a three-pass algorithm that \((1 + \epsilon)\)-approximates the number of 4-cycles using \( O(\epsilon^{-2}m(T^{1/4})^{1/2}) \) space and a one-pass algorithm that uses \( O(\epsilon^{-2}n) \) space when \( T = \Omega(n^2) \). The best existing result, a \((1 + \epsilon)\)-approximation using \( O(\epsilon^{-2}m^2/T) \) space, was presented by Bera and Chakrabarti (STACS 2017). We also show a multi-pass lower bound and another algorithm for distinguishing graphs with no four cycles and graphs with many 4-cycles.

ACM Reference Format:

License: Copyright © 2018 Association for Computing Machinery. ACM ISBN 978-1-4503-XXXX-X/18/06. $15.00
https://doi.org/10.1145/1122445.1122456

Footnote 1: 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 \( (v_1, v_1, v_2, v_3, v_1, v_2, v_2, v_3, v_1, v_1) \). In this example, we say that the adjacency list for \( v_3 \) came first, then the adjacency list for \( v_2 \), and finally the adjacency list for \( v_1 \).

1.1 Our Results

We improve the state-of-the-art results for triangle counting in random order streams and four-cycle counting in both arbitrary order streams and adjacency list streams. All of our approximation algorithms yield a \((1+\epsilon)\)-approximation for arbitrary \( 0 < \epsilon < 1/100 \). In what follows, \( n \) and \( m \) are the number of vertices/edges in the input graph and \( T \) is a lower bound on the quantity being estimated, i.e., either the number of triangles or the number of four-cycles.

1.1.1 Random Order Model. We present a single pass algorithm that \((1 + \epsilon)\)-approximates the number of triangles \( T \) using

\[
\tilde{O}(\epsilon^{-2}m/\sqrt{T})
\]
space and prove that this is optimal in the range \( T \leq \sqrt{m} \). The best previous result, a \((3 + \epsilon)\)-approximation using \( O(e^{-4.5m/\sqrt{T}}) \) space, was presented by Cormode and Jowhari \[12\].

The main idea in the upper bound is a new method for identifying edges that participate in many triangles; this turns out to be crucial to break the factor 3 approximation barrier. The main idea in the lower bound is a reduction from a communication complexity problem in the "random partition setting" where the input is randomly partitioned between the players, in contrast to the standard setting where a worst case partition is assumed. We present the random order model results in Section 2.

1.1.2 Adjacency List Model. We present a two-pass algorithm that \((1 + \epsilon)\)-approximates the number of 4-cycles using \( O(e^{-4m/T^{1/2}}) \) space. This best previous result, a constant approximation using two passes and \( O(m/T^{3/8}) \) space, was presented by Kallaugh et al. \[18\]; our result achieves a better approximation in less space.

The main idea is to group 4-cycles into what we call "diamonds" where the \((u, v)\)-diamond is a complete bipartite graph between \([u, v]\) and \(\Gamma(u) \cap \Gamma(v)\); such a graph corresponds to \( \frac{|\Gamma(u) \cap \Gamma(v)|^2}{2} \)-cycles. By estimating the diamonds of different sizes rather than individual four cycles, we are able to reduce the variance of the estimation process and this results in a more space efficient algorithm.

We also show that \((1 + \epsilon)\)-approximation in a single pass is possible in a) \(\text{polylog}(n)\) space if \( T = \Omega(n^2) \) and b) \(\Omega(n)\) space if \( T = \Omega(n) \). The main idea for both of these algorithms is to observe that

\[
\frac{|\Gamma(u) \cap \Gamma(v)|^2}{2} \geq \frac{|\Gamma(u) \cap \Gamma(v)|^2}{2}
\]

to exploit connections to frequency moment estimation and \(\ell_2\) sampling. We present the adjacency order model results in Section 4.

1.1.3 Arbitrary Order Model. We present a three-pass algorithm that \((1 + \epsilon)\)-approximates the number of 4-cycles using \( O(e^{-2m/T^{1/4}}) \) space and a one-pass algorithm that uses \( O(e^{-2m}) \) space when \( T = \Omega(n^2) \). Estimating the number of 4-cycles in the arbitrary order model is significantly more challenging than estimating the number of triangles because of locality issues. Specifically, it is relatively easy to determine whether an edge \((u, v)\) exists in many triangles: we sample a set of nodes and compute the fraction that would form a triangle with \((u, v)\). For 4-cycles, things get more complicated since there are two other nodes in the 4-cycle. The best existing result, a \((1 + \epsilon)\)-approximation using \( O(e^{-2m^2/T}) \) space and four passes, was presented by Bera and Chakraborti \[4\]; our first algorithm uses fewer passes and uses less space when \( T \leq m^{4/3} \).

We also present a two-pass algorithm for distinguishing graphs with no 4-cycles and graph with at least \( T \) 4-cycles using \( O(m^{3/2}/T^{3/4}) \) space. The algorithm leverages a result in extremal graph theory and uses less space than our approximation algorithm when \( T \geq m \). Lastly, we present a \( O(m/\sqrt{T}) \) space lower bound for any constant pass algorithm that solves this problem. We present the arbitrary order results in Section 5.

Notation. Throughout this paper \( V \) is the vertex set of the input graph, \( E \) is the set of edges, and \( n = |V| \) and \( m = |E| \). A wedge is a length two path.

2 TRIANGLES IN RANDOM ORDER MODEL

Our main algorithmic result for counting triangles in the random order model is:

**Theorem 2.1.** There exists an algorithm which takes one pass over a randomly ordered stream, uses \( O(e^{-2m/\sqrt{T}}) \) space, and returns a \((1 + \epsilon)\)-approximation of the number of triangles in the graph with probability at least 99/100.

We then show that the above result is optimal when \( 1 \leq T \leq \sqrt{m} \).

2.1 One Pass Algorithm using \( O(m/\sqrt{T}) \) space

2.1.1 Basic Idea. One of the main ideas implicit in previous work \[12, 27\] is that, assuming no edge is involved in too many triangles, the number of triangles can be approximated by sampling roughly \( O(m/\sqrt{T}) \) edges of the graph and counting the number of triangles that include two edges from the sampled set. The threshold for "too many" is roughly \( \sqrt{T} \) and such edges are referred to as being "heavy" in the literature; we will also use this term but defer an exact definition until the next section. A natural approach is to a) use the above idea to count triangles without heavy edges and b) recognize heavy edges and account for triangles involving these edges separately. It is possible to do this in two passes \[12, 27\] if the stream is ordered arbitrarily. The main technical breakthrough of our new algorithm is a novel way to identify potential heavy edges as they arrive in the stream, assuming the stream is randomly ordered.

2.1.2 Algorithm. It will be helpful to define the following notation: For an edge \( e \) and a set of edges \( F \), let \( t_e^F \) be the number of triangles in \( \{e\} \cup F \) that include \( e \). Let \( t_e := t_e^E \). During a single pass, the algorithm a) stores a set of edges that will include most edges involved in many triangles and b) collects all edges in triangles that include two edges in a prefix of the stream.

- **Finding Potentially Heavy Edges:** For \( i = 0, 1, \ldots, \log \sqrt{T} \), let \( V_i \) be a subset of vertices where each vertex is sampled with probability \( p_i := \min\{1, 10ce^{-2m/(\log n)^2}\} \) and let \( E_i \) be defined as the set of edges incident to \( V_i \) amongst the first \( q_i \) elements of the stream, where \( q_i := 2^i/\sqrt{T} \). During a pass over the stream, store the following set of edges:

\[ P := \{e \in E : \text{position of } e \text{ in stream is } > q_i m \text{ and } t_e^E \geq 1\} \]

We will later show that edges involved in many triangles are very likely to be included in \( P \).

- **Rough Estimator:** Let \( S \) be the first \( rm \) elements of the stream where \( r = ce^{-1}/\sqrt{T} \). During a pass over the stream, store the following set of edges:

\[ C := \{e \in E : t_e^C \geq 1\} \]

We will later show that \( S \) and \( C \) are sufficient to estimate the number of "light" triangles in the graph, i.e., triangles that do not include any edges that occur in many triangles.

- **Post-Processing:** Let \( O = E_{\log(\sqrt{T})} \) and \( p = P_{\log(\sqrt{T})} \):

\[
\text{oracle}(e) = \begin{cases} 
L & \text{if } t_e^O < pT \\
H & \text{otherwise}
\end{cases}
\]
The oracle will henceforth be used to define whether an edge is heavy or light. That is, while we will later show that $e$ being defined as heavy/light will roughly correspond to whether $t_e$ is larger/smaller than $\sqrt{T}$, the actual definition is a function of the edges sampled by the algorithm; this will help significantly with the analysis. Note that the oracle is defined independently of the ordering of the data stream because $E_{\log(T)}$ is constructed based on the entire stream rather than a strict prefix. Let $S^4, C^4, P^4$ be the subsets of the respective sets restricted to heavy or light edges as appropriate. Return
\[
\frac{1}{3\sqrt{2}} \sum_{e \in C} t_e^{C} + \frac{1}{p} \sum_{e \in P} \left( t_e^{O} + t_e^{I} + t_e^{O} + t_e^{I}/3 \right)
\]
where $t_e^{O}$ is the number of triangles including $e$ where $i$ of the other two edges in $O$ in heavy. The coefficients of $t_e^{O}$ take into account that a triangle with multiple heavy edges can be counted from the perspective of multiple edges and hence we need to compensate for overcounting.

2.1.3 Accuracy Analysis. To prove that the algorithm returns a $1 + \epsilon$ approximation we will argue that a) the oracle distinguishes between edges $e$ with large or small $t_e$ values with sufficient accuracy, b) the algorithm stores almost all edges with large $t_e$ values, and c) the number of triangles made of edges with small $t_e$ values, can be accurately estimated given the “rough estimator.” We then prove a space bound for the algorithm.

Properties of the Oracle. Note that for any edge $e$, $t_e^O \sim \text{Bin}(t_e, p)$. The following lemma then follows from the Chernoff bound.

**Lemma 2.2 (Oracle Guarantees).** With high probability, $t_e^O = (1 \pm \epsilon) t_e p$ for all heavy edges $e$ and $t_e \leq 2\sqrt{T}$ for all light edges.

Note that the number edges that are designated as heavy is at most $4\sqrt{T}$ since if $e$ is heavy then $t_e \geq p\sqrt{T}/(p(1 + \epsilon)) = \sqrt{T}/(1 + \epsilon)$ and summing $t_e$ over all heavy edges in at most $37$. Finding Potentially Heavy Edges. We now argue that $P$, the set of potentially heavy edges we construct, will contain most of the edges that ultimately will be defined to be heavy by the oracle. In particular, let $i = \lfloor \log_2 T/(ce^{-2}t_e) \rfloor$ and note that if $e$ does not arrive within the first $q_m$ then the probability $e$ is not added to $P$ is at most
\[
1 - q_m^2 p_l t_e \leq e^{-t_e \cdot 10e^{-3}(\log n)^2/T} \leq 1/n^{10}
\]
because the events that different triangles are formed between $e$ and edges in $E_l$ are negatively associated. Therefore, the probability that we do not include heavy edge $e$ in $P$ is
\[
P \left[ e \in E^I \setminus P \right] \leq 1/n^{10} + P \left[ e \text{ appears in a prefix of length } q_m \right] = 1/n^{10} \cdot 2^i / \sqrt{T} \leq 2 \cdot \frac{4\sqrt{T}}{ce^{-2}t_e},
\]
where we used the fact that $1/n^{10}$ was dominated by the second term for sufficiently large $n$.

The following lemma establishes that $P$ includes most of the heavy edges in terms of their contribution to the sum of their $t_e$ values.

**Lemma 2.3 (Missing Heavy Edges).** $\sum_{e \in E^I \setminus P} t_e \leq \epsilon T$ with probability at least $1 - 1/c$.

**Proof.**
\[
\mathbb{E} \left[ \sum_{e \in E^I \setminus P} t_e \right] \leq \sum_{e \in E^I} \frac{4\sqrt{T}}{ce^{-2}t_e} \cdot t_e \leq \left| E^I \right| \cdot \frac{4\sqrt{T}}{ce^{-2}t_e} \leq \frac{16|T|}{ce^{-2}t_e}.
\]
using the fact $|E^I| \leq 4\sqrt{T}$ (see discussion following Lemma 2.2). The result then follows by an application of the Markov bound assuming $\epsilon < 1/16$.

Accuracy. The total number of triangles can be written as:
\[
T_0 + T_1 + T_2 + T_3 = T_0 + \sum_{e \in \text{heavy}} (t_e^O + t_e^I/2 + t_e^{O}/3)
\]
where $T_i$ is the number of triangles with $i$ heavy edges and $t_e^i$ is the number of triangles including $e$ where exactly $i$ of the other edges are heavy. The next two lemmas consider errors incurred when estimating the terms $T_0$ and $T_1 + T_2 + T_3$.

**Lemma 2.4.** With probability at least $1 - 1/c^2$,
\[
\frac{1}{3\sqrt{2}} \sum_{e \in C} t_e^C = T_0 \pm \epsilon T.
\]

**Proof.** Let $X$ be the number of light wedges (paths of length 2) in $S$ that can be completed by another light edge. Note that $X = \sum_{e \in C} t_e^C$ and $\mathbb{E}[X] = 3e^2 T_0$. Since each edge in $S^4$ can only occur in at most $2\sqrt{T}$ triangles, the variance can be calculated as
\[
\mathbb{V}[X] \leq 3e^2 T_0 + 6T_0 e^2 \sqrt{T} < 9e^2 T_0 < 9e^2 \epsilon^{-2}
\]
Hence, by an application of the Chebychev bound,
\[
\mathbb{P} \left[ |X - \mathbb{E}[X]| \geq 3e^2 T \right] \leq 1/c^2.
\]

**Lemma 2.5.** With high probability,
\[
\frac{1}{p} \sum_{e \in \text{heavy}} (t_e^O + t_e^I/2 + t_e^{O}/3) = (1 \pm \epsilon)(T_1 + T_2 + T_3).
\]

**Proof.** By an application of the Chernoff bound, for each heavy edge $e$, with high probability
\[
t_e^O + t_e^I/2 + t_e^{O}/3 = (1 \pm \epsilon)p(t_e^O + t_e^I/2 + t_e^{O}/3).
\]
The result follows by summing over the heavy edges.

Therefore, by combining Lemmas 2.3, 2.4, and 2.5 we establish that the estimator gives a $1 + \epsilon$ approximation with probability at least $1 - 1/c - 1/c^2 - 1/poly(n) \geq 1 - 3/c$.

2.1.4 Space Analysis. It remains to analyze the space used by the algorithm. The expected space to store all $E_l$ is at most
\[
m \sum_{i=0}^{\log \sqrt{T}} \frac{2^i}{\sqrt{T}} \cdot 2e^{-2}/2^i = 2e^{-2}m/\sqrt{T}.
\]
The expected size of $CS$ is at most $3e^2 T$. Hence, the expected space used by the algorithm is $\tilde{O}(e^{-2}m/\sqrt{T})$ as claimed. Note that it exceeds this space by a factor $1/\delta$ with probability at most $\delta$ using a standard Markov Bound argument.
2.2 Lower Bound

We now prove the following space lower bound for distinguishing between triangle-free graphs from graphs with many triangles. Note that this implies no multiplicative approximation is possible in less space.

**Theorem 2.6.** Assume \(1 \leq T \leq \sqrt{m}\). Any algorithm that distinguishes between 0 and \(T\) triangles in a graph presented in random order with probability at least \(1 - 1/m^2\) requires \(\Omega(m/\sqrt{T})\)

The proof will be via a reduction from communication complexity in the random partition setting [8]. The proof involves constructing a graph based on a binary matrix and the index to an entry of this matrix such that the graph has \(T\) triangles if this entry is 1 and is triangle-free otherwise. Note that it is relatively easy to find a construction with this property and this sufficed to prove previous bounds in the arbitrary order model. The main challenge in proving the result here is to ensure that if the edges of the graph arrive in random order, then the identity of the entry of interest is not revealed too early in the stream.

We start by presenting the construction. Given \(x \in \{0, 1\}^{n \times n}\) let \(S_x = \{(x_i, j, u_i, v_j) : 1 \leq i, j \leq n\}\) and let \(E_x = \{(u_i, v_j) : x_{i,j} = 1\}\). Let

\[
P_{A_1, ..., A_n, B_1, ..., B_n} = \{(u_i, w) : w \in A_i, i \in [n]\}\cup\{(v_j, w) : w \in B_j, j \in [n]\}
\]

Let \(P_{r, j'}\) be the collection of all such sets where \(A_{r'} = B_{j'}\) and all the sets aside from the \(B_{j'}\) are disjoint sets of size \(T\). Note that the graph consisting of edges \(E_x \cup P\) for any \(P \in P_{r, j'}\) has \(T\) triangles if \(x_{r', j'} = 1\) and is triangle-free otherwise. See Figure 1.

**Theorem 2.7.** Let \(p = c/\sqrt{T}\) for some small constant \(c\) and assume \(np\) is an integer. Let \(x \in \{0, 1\}^{n \times n}\) and \(r', j' \in [n]\). Suppose each entry of \(S_x\) and a random set \(P \in P_{r, j'}\) is revealed to Alice with probability \(p\) and revealed to Bob otherwise. Then the randomized one-way communication from Alice required for Bob to determine whether \(E_x \cup P\) contains 0 or \(T\) triangles with probability at least \(1 - 1/n^2\) is \(\Omega(n^2/\sqrt{T})\).

**Proof.** Suppose \(A\) is protocol that succeeds with probability \(\mathbb{P}[\text{success}] \geq 1 - \delta\). Let \(C\) be the event that \((x_{r', j'}, r', j')\) is revealed by Alice along with exactly \(pn^2 - 1\) other elements from \(S_x\). Then

\[
\mathbb{P}[C] = p \times p^{n^2 - 1} \times (1 - p)^{(n^2 - 1)/2} \geq \frac{p}{n^2}
\]

and the success probability of \(A\) conditioned on \(C\) is \(\mathbb{P}[\text{success} | C] = 1 - \mathbb{P}[\text{fail} | C] \geq 1 - n^{-2}\delta/p\).

Consider a reduction from a random instance of the Index problem where Alice has a random binary string \(z \in \{0, 1\}^{n^2p}\) and Bob has a random index \(k \in [n^2p]\). Recall, that the one-way communication complexity of solving Index with probability at least 4/5 is \(\Omega(n^2p)\).

- Using public randomness, Alice and Bob determine a random partition of elements of the form \((u, v, j)\) where exactly \(n^2p\) elements of this form are received by Alice. Again using public randomness, they determine an ordering of these elements and let \((u_{f_k(1)}, v_{f_k(1)})\) be the \(l\)th element according to this ordering. Bob sets \(z' = f_k(k)\) and \(z'' = f_k(1)\). Using public randomness, the players define \(2n\) random variables \(b_r \sim \text{Bin}(T, p)\) for all \(r \in U \cup V\).
  - Alice uses the bits of \(z\) to populate the entries of the form \((u, v, j)\) received by Alice, i.e., she sets the first argument of \((u_{f_k(1)}, v_{f_k(1)})\) is set to \(z'\).
  - Bob uses iid binary values to populate the entries of the form \((u, v, j)\) received by Bob.
  - Alice adds an edge between each \(r \in U \cup V\) and \(b_r\) random neighbors random vertices in \(W\) and such that all vertices in \(W\) have degree at most 1.
  - For each \(r \in U \cup V\), add \(T - b_r\) random vertices in \(W\) such that all vertices in \(W\) still have degree at most 1. Add edges from \(u_{r'}\) to all neighbors of \(v_{r'}\) in \(W\) and from \(v_{r'}\) to all neighbors of \(u_{r'}\) in \(W\). Amongst the isolated vertices in \(W\), select \(T - b_{u_{r'}} - b_{v_{r'}}\) at random and add edges to \(u_{r'}\) and \(v_{r'}\) (if \(T - b_{u_{r'}} - b_{v_{r'}} < 0\) then output fail).

Note that in the distribution defined above, there is at most one incident edge to each vertex in \(W\) amongst the edges determined by Alice. However, according to the required distribution, for each of the \(T\) vertices in \(W\) of degree 2, there should be \(\text{Bin}(2, p)\) incident edges amongst the edges defined by Alice. However, the variational
distance between this distribution and the required distribution is $T_p^2 \leq c^2$. Hence, the algorithm succeeds with probability at least $1 - n^2\delta/|p| - c^2$. Setting $c = 1/\sqrt{10}$ and $\delta = p/(10n^2)$ ensures that any one-way-protocol for the triangle problem that succeeds with probability at least $1 - \delta$ uses at least $\Omega(n^2p)$ communication since it also solves the Index problem with probability at least $4/5$. \hfill $\Box$

We are now ready to prove Theorem 2.6.

**Proof of Theorem 2.6.** Let $n = \sqrt{m}$ and note that the graph $E_x \cup P$ has at most $m + 2\sqrt{m} |P| \leq 3m$ edges. The players define a random stream: each player randomly permutes the tokens they have already seen. Hence, the algorithm succeeds with probability at least $\Omega(1/\sqrt{m})$ for all heavy $v$. Thus, the number of heavy vertices is at most $2W/\sqrt{M}$. Then the expected number of heavy vertices in $R_2$ is $2Wp/\sqrt{M} = O(W/M)$. To sum it up, in expectation, the total space used by the algorithm is $O(|R_1| + |R_2| + W/M)$.

### 3 4-CYCLE COUNTING PRELIMINARIES

In this section, we describe an algorithm which is going to be used as a subroutine in two of our 4-cycle counting results. Initially, the problem was motivated by the adjacency list algorithm in Section 4.1. However, we present it with enough abstraction for it to apply to the arbitrary order algorithm in Section 5.1 as well.

Let $G = (V, E)$ be a weighted graph with edge weights between 1 and $\lambda$ (for constant $\lambda$), and let $W$ be the total weight of edges in the graph. The goal is to estimate $W$ in the following scenario. Let $R_1$ and $R_2$ be subsets of vertices determined independently by sampling vertices with probability $\lambda/2$. We show that using one pass over the stream, we can $\approx M$ additively approximate $W$ if $W$ is smaller than $M$, and can also distinguish between cases when $W \geq 2M$ and $W \leq M/2$ with high probability.

Direct all edges towards the vertex that comes earlier in the stream. Let $w^{fr}(v)$ and $w^{af}(v)$ be the total weight of edges going into and out of $v$ respectively. Then $\sum_v w^{fr}(v) = W$. Also let $w^{fr}(v)$ and $w^{af}(v)$ be the weight of edges on vertices in $R_1$ pointing towards/away from $v$. We use edges on vertices in $R_1$ to distinguish between vertices with large and small $w^{fr}(v)$. If $w^{fr}(v) \geq pM$, we say that $v$ is heavy and call all such vertices $V_H$. Otherwise, $v$ is light and $V \setminus V_H = V_L$. Then, using edges on $R_2$, we approximate the sum of $w^{fr}(v)$ values of light vertices and estimate each $w^{fr}(v)$ individually for heavy $v$. Two separate sets $R_1$ and $R_2$ are used for independence reasons, which allows for simpler analysis.

#### 3.0.1 The “Useful” Algorithm

Initialize: $A \leftarrow 0$, $A_L \leftarrow 0$, $A_H \leftarrow 0$, $V_H \leftarrow \emptyset$

For every vertex $v$ in the stream:
- $A \leftarrow A + w^{af}(v)$
- For each $u \in V_H$, if there is an edge $uv$:
  - $A_H \leftarrow A_H + w(vu)$
- If $w^{fr}(v) \geq p\sqrt{M}$:
  - $V_H \leftarrow V_H \cup \{v\}$ and initialize $a(v) \leftarrow 0$
  - $A_H \leftarrow A_H + w^{fr}(v)$
  - $A_L \leftarrow A - \sum_u V_H a(u)$

Return: $\hat{W} = (A_L + A_H)/p$

### 3.0.2 Error Analysis

The proof of the following lemma can be found in the appendix.

**Lemma 3.1.** With high probability,

a. If $W \leq M$ then $\hat{W} = W \pm \epsilon M$

b. If $\hat{W} < M$ then $W \leq 2M$

c. If $\hat{W} \geq M$ then $W \geq M/2$

### 3.0.3 Space Analysis

The algorithm stores sets $R_1$ and $R_2$ and keeps one extra bit per vertex in those sets to mark which of them we have seen in the stream. In addition, we are keeping track of a few global counters and a set of heavy vertices in $R_2$ with one counter per such vertex. We now analyze the expected size of this set. In the proof of Lemma 3.1, we show that $w^{fr}(v) \geq \sqrt{M}/2$ for all heavy $v$. To sum it up, in expectation, the total space used by the algorithm is $O(|R_1| + |R_2| + W/M)$.

### 4 4-CYCLES IN ADJACENCY LIST MODEL

#### 4.1 Two-pass $\tilde{O}(m/\sqrt{T})$ Space Algorithm

In this section, we present an algorithm which takes two passes over an adjacency list stream, uses space $\tilde{O}(m/\sqrt{T})$, and returns a $(1 + \epsilon)$-approximation of the number of 4-cycles in the graph.

**4.1.1 Basic Idea.** The main idea of our algorithm is that instead of counting 4-cycles individually, as it was done in previous work, we count them “grouped together” into subgraphs we call *diamonds*. A diamond is a structure consisting of two vertices $u$ and $v$ and $h$ paths of length 2 connecting them. Essentially, a diamond is a complete bipartite graph $K_{2,h}$, and it includes $\binom{h}{2}$ 4-cycles. We call $u$ and $v$ the *endpoints* of the diamond and $d(u, v) = h$ its size.

We show that a diamond can be identified in the stream by first sampling one of its endpoints (say $u$) and then sampling at least 2 edges involved in the diamond which are incident to $u$. When $v$ arrives in the stream in the second pass, we observe at least one 4-cycle in the diamond. With appropriate edge sampling probability, it is possible to also estimate the size of the diamonds we discover.

We then partition the diamonds in the graph into log $n$ groups by size (group $j$ contains diamonds of size roughly between $2^j$ and $2^{j+1}$) and show how to approximate the number of 4-cycles within each group. Note, that each 4-cycle belongs to exactly two diamonds. A small caveat is that we don’t want to count a diamond towards two groups of similar size. Thus, we disregard diamonds with size being too close to the group “boundary”. In order to avoid potentially missing too many 4-cycles, we also try $\log k$ different shifts of group boundaries and show that at least one of those shifts allows us to account for most 4-cycles.

**4.1.2 Algorithm for Estimating Diamond Size.** Consider the following procedure for estimating diamonds of size roughly $sk$:

1. **Pass 1:**
We now refer to the Useful Algorithm in Section 3 to approximate Woodstock ‘18, June 03–05, 2018, Woodstock, NY Andrew McGregor and Sofya Vorotnikova

For each vertex $v$ in the stream:
- Let $\widehat{d}(u,v) = a(u,v)/p_e$, which is the estimate of the size of the diamond between $u$ and $v$.

4.1.3 Accuracy Analysis.

**Lemma 4.1.** If a diamond with endpoints $u \in V_{sk}$ and $v \in V$ has size $d(u,v) \geq sk$, then $\widehat{d}(u,v) = (1 + \epsilon/10) d(u,v)$ whp.

**Proof.** $\mathbb{E}[\widehat{d}(u,v)] = d(u,v)$. By an application of the Chernoff bound,

$$\mathbb{P}\left[|\widehat{d}(u,v) - d(u,v)| \geq \frac{\epsilon d(u,v)}{10}\right] \leq \exp\left(-\frac{\epsilon^2 d(u,v) p_e}{300}\right) \leq \frac{1}{\text{poly}(n)}$$

since $d(u,v) \geq sk$ and $p_e = \frac{\log n}{sk}$ with sufficiently large constant $c$.

Let $D_{sk}$ be the set of diamonds of size between $sk$ and $2sk$. Let $D_{sk}' \subseteq D_{sk}$ be the set of diamonds of size $(1 + \epsilon/3)sk$ to $(1 - \epsilon/3)2sk$. We want to avoid potentially counting the same diamond towards two sets $D_{sk}$ and $D_{sk}'$ with diamonds of close size, thus we only consider ones with

$$(1 + \epsilon/6)sk \leq \widehat{d}(u,v) < 2(1 - \epsilon/6)sk,$$

call that set $D_{sk}''$. Note that $D_{sk}' \subseteq D_{sk}'' \subseteq D_{sk}$.

Define the following weighted graph $H_{sk}$. The vertex set is $V$.

The edges correspond to pairs $(u,v)$ such that

$$(1 + \epsilon/6)sk \leq \widehat{d}(u,v) < 2(1 - \epsilon/6)sk,$$

and the weight of the edge is $\big(\frac{\widehat{d}(u,v)}{2}\big) / \binom{sk}{2}$. The weight is then the approximate number of 4-cycles in the diamond between $u$ and $v$ scaled down by $\binom{sk}{2}$ so that it is between 1 and a constant.

We now refer to the Useful Algorithm in Section 3 to approximate the total weight of edges in $H_{sk}$. Note that the algorithm calls for two random sets of vertices, thus we run two copies in parallel, collecting sets $V_{sk}^1, V_{sk}^2, E_{sk}^1,$ and $E_{sk}^2$ and pass them to the Useful Algorithm. Call the estimate returned by the Useful Algorithm $\widehat{w}_{sk}$ and let $\widehat{t}_{sk} = \frac{w_{sk}}{\binom{sk}{2}}$.

Let $T(D)$ be the number of 4-cycles involved in the diamonds in the set $D$. According to Lemma 3.1, with high probability

$$\widehat{w}_{sk} = \left(1 + \epsilon/10\right)T(D_{sk}') + \epsilon T/3 \left(\binom{sk}{2}\right) \log n$$

where the $(1 + \epsilon/10)$ factor comes from the fact that we are using $\widehat{d}(u,v)$ instead of $d(u,v)$. Then

$$\widehat{t}_{sk} = \left(1 + \epsilon/10\right)T(D_{sk}''') + \epsilon T/(3 \log n)$$

By summing over log $n$ levels, we obtain

$$(1 - \epsilon^2/10) \sum_k T(D_{sk}') - \epsilon T/3 \leq \sum_k \widehat{t}_{sk} \leq (1 + \epsilon^2/10) \sum_k T(D_{sk}) + \epsilon T/3$$

We now show that we can find a shift $s$ such that $\sum_k T(D_{sk}') \geq 2(1 - 2\epsilon)T$. Consider $\log_{1+2\epsilon}$ 2 shifts of the form $(1 + \epsilon)/\ell$. Note that for shift $s_i = (1 + \epsilon)/\ell$ we might miss diamonds with $(1 - \epsilon/3)(1 + \epsilon)/\ell k < d(u,v) < (1 + \epsilon/3)(1 + \epsilon)/\ell k$ for every $k$. Call the set of diamonds we might miss with this shift $M_i$. Also note that $(1 + \epsilon/3)(1 + \epsilon)/\ell k < (1 - \epsilon/3)(1 + \epsilon)/\ell k + 1$ and thus sets $M_i$ are disjoint. Since $\sum_i T(M_i) \leq 2\epsilon T$, there exists a set $M_T$ such that $T(M_T) \leq 2T/\log_{1+2\epsilon} 2 \leq 4\epsilon T$.

To sum it up, there exists a shift $s_T$ such that

$$2(1 - 3\epsilon)T \leq \sum_k \widehat{t}_{sk} \leq 2(1 + \epsilon)T$$

Therefore, we can run $\log_{1+2\epsilon}$ 2 copies of the algorithm in parallel (one for each shift value) and then pick the run which outputs the largest estimate. We then scale it down by 2 to account for each 4-cycle belonging to two diamonds.

4.1.4 Space Complexity. For each level $sk$ the expected space use is

$$mp_e p_e = O(m e^{-4} (\log^2 n) / \sqrt{T}).$$

The value $s$ takes $\log_{1+2\epsilon} 2 = O(1/\epsilon)$ values and $k$ takes $\log n$ values. Thus, the total expected space of the algorithm is $O(m e^{-4} (\log^2 n) / \sqrt{T})$.

We now state the final result.

**Theorem 2.** There exists an adjacency order streaming algorithm that returns a $(1 + \epsilon)$-approximation of the number of 4-cycles in the graph with high probability using two passes and $O(\epsilon^{-4} m / \sqrt{T})$ space.

4.2 One-pass algorithms when $T$ is large

We now present single-pass algorithms for four-cycle counting when $T$ is large. The main approach is to reduce the problem to the problem of estimating frequency moments and the related $\ell_2$ sampling problem.

**Theorem 3.** There exists a single pass algorithms for $1 + \epsilon$ approximating $T$ in the adjacency list model using:

- $\tilde{O}(\epsilon^{-4} m / T^2)$ space.
- $\tilde{O}(\Delta + e^{-4} n^2 / T)$ space where $\Delta$ is the maximum degree.

Note that if $T = \Omega(n^2 / \epsilon^2)$, polylog($n$) space is sufficient and if $T = \Omega(n)$, $\tilde{O}(n)$ space is sufficient.

4.2.1 Basic Idea. We reduce the problem of 4-cycle counting to analyzing the vector $x \in \mathbb{N}^{|V|}$ where $x_{uv}$ is the number of wedges, i.e., length two paths, with endpoints $u, v$. Note that $x_{uv} \leq n - 2$ and

$$\sum x_{uv} = 2T$$

where $T$ is the number of 4-cycles. The factor two is present because a 4-cycle on vertices $v_1, v_2, v_3, v_4$ is counted once via $x_{v_1 v_2}$ and once via $x_{v_3 v_4}$. Let $z_{uv} = \min(x_{uv}, 1/\epsilon)$ and consider the quantity

$$F_1(z) = 2 + 2 \sum x_{uv} / 2 = F_1(z) + 4T.$$
Lemma 4.4. \( F_2(x) - 4\epsilon T \leq F_1(x) + 4T \leq F_2(x) \).

Proof. The lemma follows by considering each term in the summations. Let \( a \in N \). If \( a \leq 1/\epsilon \) then, \( \min(a, 1/\epsilon) + 2 \left( \frac{a}{2} \right) = a^2 \). If \( a > 1/\epsilon \), then
\[
a^2 \geq \min(a, 1/\epsilon) + 2 \left( \frac{a}{2} \right) = a^2 - a + 1/\epsilon > a^2 - a + 1 > a^2 - \epsilon a (a - 1)
\]
using the assumption that \( \epsilon < 1 \).

It follows that if we can approximate each of \( F_1(x) \) and \( F_2(x) \) up to an additive \( \pm O(\epsilon T) \) term then we can get a \( 1 + O(\epsilon) \) multiplicative approximation of \( T \). In the next two sections we show that it is possible to approximate both terms with this additive error using \( O(\epsilon^{-2}n^2T^2) \) space; this establishes the first part of Theorem 4.3.

The main challenge in using existing frequency moment algorithms is that an adjacency list of length \( \ell \) corresponds to \( \binom{\ell}{2} \) updates to \( x \) and it appears that an algorithm would have to store the entire adjacency list in order to perform these updates. This would not allow us to claim a polylog\( (n) \) space algorithm when \( T = \Omega(n^2) \).

4.2.2 \( F_2(x) \) Algorithm and Analysis. First note that \( F_1(x) \leq n^2/\epsilon \) and hence by Lemma 4.4,
\[
F_2(x) \leq 4\epsilon T + n^2/\epsilon + 4T .
\]

Therefore, a \( 1 + \gamma \) approximation of \( F_2(x) \) where \( \gamma = \epsilon \min(1, \epsilon T/n^2) \) yields an additive
\[
\gamma F_2(x) \leq 4\epsilon T + n^2/\epsilon + 4T \leq 4\epsilon^2 T + \epsilon T + 4\epsilon T < 9\epsilon T
\]

approximation of \( F_2(x) \). Our algorithm for a \( 1 + \gamma \) approximating \( F_2(x) \) will be based on the following basic estimator that a) can be computed in small space and b) has the correct expectation and low variance.

1. For each \( t \in V \), compute
\[
A_t = \sum_{u \in \Gamma(t)} \alpha_u \quad B_t = \sum_{u \in \Gamma(t)} \beta_u \quad C_t = \sum_{u \in \Gamma(t)} \alpha_u \beta_u
\]
where \( \alpha, \beta \in [-1, 1]^n \) and the entries of \( \alpha \) and \( \beta \) are determined by 4-wise independent hash function.

2. Return \( X = (\sum_{t \in V} (A_t B_t/2 - C_t/2))^2 \).

This basic estimator can be computed in \( O(\log n) \) space in the adjacency list model since it suffices to maintain 4 counters at any one time. Specifically, while processing the adjacency list for \( v \), we compute \( A_v, B_v, C_v \) and once the adjacency list is completed, we compute \( A_v B_v - C_v/2 \) and add the value to a fourth counter. These first three counters can be reused to process the next adjacency list, and at the end of the stream, squaring the fourth counter returns the value of \( X \).

We can calculate the expectation of \( X \) as follows:
\[
\mathbb{E}[X] = \mathbb{E} \left[ \left( \sum_{t, u, v \in \Gamma(t), u \neq v} \alpha_u \beta_v \right)^2 \right]
\]
\[
\mathbb{E} \left[ \sum_{t, u, v} \alpha_u \beta_v \right]^2
\]
\[
\sum_{t, u, v} \mathbb{E}[X_{u, v}]^2
\]
Hence, \( \mathbb{E}[X] = F_2(x) \) and the \( \mathbb{V}[X] = O(F_2(x)) \) where the variance follows from [15]. Repeating the process \( O(1/\epsilon^2 \log 1/\delta) \) times in parallel yields a \( 1 + \gamma \) approximation via the median of means analysis.

4.2.3 \( F_1(x) \) Algorithm and Analysis.

1. Initialization: Let \( S \) be a random sample of vertex pairs where each pair is sampled independently with probability \( p = 6\epsilon^{-4} \eta^2 n^2 T^2 \log n \).

2. During a single pass: For each \( \{u, v\} \in S \), compute \( z_{u, v} \).

3. Postprocessing: Return \( Z = 1/p \sum_{\{u, v\} \in S} z_{u, v} \).

Note that \( \mathbb{E}[Z] = F_1(x) \) and since max \( z_{u, v} \leq 1/\epsilon \), by an application of the Chernoff bound,
\[
\mathbb{P}[|Z - \mathbb{E}[Z]| \geq \epsilon T] \leq 2 \exp(-\epsilon^2 T/F_1(z)^2 p F_1(z)/3) = 2 \exp(-6\epsilon^2 T^2 p (F_1(z)^2)) \leq 2/n^2
\]
The space used is \( O(p n^4 \log n) = O(\epsilon^{-4} \eta^2 n^2 T^2 \log^2 n) \) bits.

4.2.4 Alternative Approach via \( \ell_2 \) Sampling. To prove the second part of Theorem 4.3, we analyze \( x \) via \( \ell_2 \) sampling rather than norm estimation. The basic approach is as follows: define a random variable \( X \) by sampling \( u, v \) with probability \( x_u^2/F_2(x) \) and letting \( X = 1 \) with probability \( (x_u - 1)/(4x_u) \) and 0 otherwise. Then \( \mathbb{E}[X] = T/F_2(x) \). Averaging \( r = O(\epsilon^{-2} F_2(x)/T \log n) \) times gives a \((1 + \epsilon)\) approximation of \( T/F_2(x) \) with high probability. Note that
\[
F_2(x) \leq \sum_{u, v} \min(x_u, 1) + 6T
\]
and hence \( F_2(x) \leq n^2 + 6T \). Therefore, \( r = O(\epsilon^{-2} (n^2 + T)/T \log n) \).

It remains to argue that we can estimate \( F_2(x) \) (so we can rescale \( X \) to give \( T \)) and perform the necessary sampling. Given \( O(\lambda) \) space, it is possible to store each adjacency list and transform this adjacency list into a sequence of updates to the vector \( x \). Then we can \( 1 + \epsilon \) approximate \( F_2(x) \) in polylogarithmic space using existing frequency moment algorithms [2, 20]. Similarly, we can sample edges with probability proportional to \( x_u^2/F_2(x) \) using existing \( \ell_2 \) sampling algorithms [17, 23, 28].

5 4-CYCLES IN ARBITRARY ORDER MODEL

In this section, we describe two algorithms and a lower bound related to counting 4-cycles in the graph. The first algorithm takes three passes over the stream, uses space \( O(m/T^{1/3}) \), where \( T \) denotes the number of 4-cycles in the graph, and returns a \((1 + \epsilon)\) approximation. The second one takes two passes, uses space \( O(m^{3/2} T^{3/4}) \), and distinguishes between graphs with 0 and \( T \) 4-cycles. For the lower bound, we show that distinguishing between graphs with 0 and \( T \) 4-cycles in a constant number of passes requires \( O(m/T^{1/2}) \) space.

Our two algorithms rely on the following structural result that is proved in the appendix:

Lemma 5.1. We call an edge \( e \in E(G) \) "bad" if it is contained in at least \( \eta \sqrt{T} \) 4-cycles, and "good" otherwise. There are at least \( T(1 - 82/\eta) \) cycles containing no more than one bad edge.

5.1 Three-pass \( \widetilde{O}(m/T^{1/3}) \)-space Algorithm

5.1.1 Basic Idea. This result relies on two simple ideas:
We then approximate the number of 4-cycles with 3 or 4 light edges.

Sampling edges uniformly at a certain rate allows us to obtain some 3-paths which are involved in 4-cycles.

Sampling vertices uniformly and storing all incident edges allows us to build an oracle classifying edges as heavy or light (involved in a large or small number of 4-cycles).

We then approximate the number of 4-cycles with 3 or 4 light edges and use that as our estimate.

### 5.1.2 Algorithm

The outline of the algorithm is as follows (the oracle is defined later):

**Pass 1:**
- Let $p = \frac{c \log n}{\epsilon T T}$.
- Sample edges with probability $p$, call set $S_0$.
- Sample vertices with probability $p$, call set of sampled vertices $Q_1$; collect incident edges, call set $S_1$.
- Sample vertices with probability $p$, call set of sampled vertices $Q_2$; collect incident edges, call set $S_2$.

**Pass 2:** For every edge $e$ in the stream:
- Check if $e$ completes any 3 edges from $S_0$ to a 4-cycle. For each such cycle $\tau$, store $(e, \tau)$.

**Pass 3:**
- Use the oracle based on $Q_1$, $Q_2$, $S_1$, and $S_2$ to classify all edges involved in cycles stored in pass 2.
- Let $A_0$ be the number of $(e, \tau)$ pairs s.t. $\tau$ has no heavy edges.
- Let $A_1$ be the number of $(e, \tau)$ pairs s.t. $e$ is heavy and the other 3 edges in $\tau$ are light.

**Return:** $A_0/(4p^3) + A_1/p^3$.

We now describe the oracle. Assume that we are trying to classify edge $e$. We define a graph $H_e = (V_e, E_e)$ and run the Useful Algorithm from Section 3 on it. The graph is defined as follows. Let $V_e$ be the set of edges sharing an endpoint with $e$ and let $E_e$ be pairs $(e_i, e_j)$, such that there exists an edge $e_k$, such that $(e_i, e_k, e_j)$ form a 4-cycle. In other words, "edges" in $H_e$ correspond to 4-cycles in $G$ which involve edge $e$. Since $H_e$ is an unweighted graph, the Useful Algorithm returns an estimate of $|E_e|$, which corresponds to the heaviness of $e$.

Recall, that the Useful Algorithm requires a uniform independent sample of vertices in the graph. There is a small caveat in obtaining a sample from $H_e$, since we are not sampling edges of $G$ independently, but rather sample vertices and collect all incident edges. Note, that a vertex in $G$ can have zero, one, or two edges incident to it which share an endpoint with $e$ (if we sample a vertex involved in $e$ itself, simply ignore it), and thus can correspond to zero, one, or two "vertices" in $H_e$. We address this problem by further subsampling. Consider two hash functions $f : V \times E \to \{0, 1, 2, 3\}$ and $g : V \times E \to \{0, 1, 2, 3\}$ defined as follows:

$$f(v, e) = \begin{cases} 0 \text{ with probability 0.4} \\ 1 \text{ with probability 0.4} \\ 2 \text{ with probability } q \\ 3 \text{ with probability } 0.2 - q \end{cases}$$

$$g(v, e) = \begin{cases} 0 \text{ with probability 0.4 + } q \\ 1 \text{ with probability 0.6 - } q \end{cases}$$

The value for $q$ is picked such that it satisfies the following two equations: $(p(0.4 + q))^2 = pq$ and $q \leq 0.2$. Note, that it is always possible for $p < 0.1$. If $v$ has two incident edges sharing an endpoint with $e$, compute $f(v, e)$, pick the first edge if the output is 0, second if it is 1, both if it is 2, and neither if it is 3. If $v$ has one incident edge sharing an endpoint with $e$, compute $g(v, e)$, pick the edge if the output is 0 and don’t pick the edge if it is 1. It is easy to see, that this way we restore the independence of the sample for each classifier. Note, that we are still keeping the entirety of sets $S_1$ and $S_2$ and perform this additional subsampling on the fly.

Let $I_e$ be the size of $|E_e|$ returned by the Useful Algorithm when it is run on $H_e$. Define $\text{oracle}(e) = L$ if $I_e < q \sqrt{T}$ (edge is called light) and $\text{oracle}(e) = H$ if $I_e \geq q \sqrt{T}$ (edge is called heavy).

It follows from Lemma 3.1, that $\text{oracle}(e) = L$ implies $t(e) \leq 2q \sqrt{T}$ and $\text{oracle}(e) = H$ implies $t(e) \geq q \sqrt{T}/2$ with high probability.

**5.1.3 Correctness**

Let $\overline{T}$ be the estimate returned by the algorithm. Below we show that

$$(1 - 164/\eta - \epsilon)T \leq \overline{T} \leq (1 + \epsilon)T$$

with constant probability. Let $T_i$ be the number of 4-cycles with $i$ heavy edges. From Lemma 5.1, $(1 - 164/\eta)T \leq T_0 + T_1 \leq T$, since edges with $t(e) \leq qT/2$ are classified as light w.h.p.

Let $\overline{T}_0 = A_0/(4p^3)$ and $\overline{T}_1 = A_1/p^3$. Note that $\mathbb{E}[\overline{T}_0] = T_0$ and $\mathbb{E}[\overline{T}_1] = T_1$.

**Lemma 5.2.** With constant probability, $\overline{T}_0 = T_0 \pm \epsilon T/2$ and $\overline{T}_1 = T_1 \pm \epsilon T/2$.

**Proof.** By an application of the Chebyshev bound,

$$\mathbb{P}[|\overline{T}_0 - T_0| \leq \epsilon T/2] \leq 1/16$$

as long as $\sqrt{\mathbb{E}\left[\overline{T}_0^2\right]} \leq \epsilon^2 T^2/64$. We now give a bound on the variance of $\overline{T}_0$. Let $\mathcal{H}_0$ be the set of 3-paths which are involved in 4-cycles in $T_0$. Let $X_q$ be 1 if all 3 edges of path $q \in \mathcal{H}_0$ were sampled and 0
where equation 1 follows from the fact that any light path \( q \in \mathcal{H}_0 \) intersects at most \( 6\eta \sqrt{\mathcal{H}} \) other paths in \( \mathcal{H}_0 \) and equation 2 follows since \( p \geq c \sqrt{m/(e^2T^2/4)} \) for sufficiently large constant \( c \).

Proving \( \mathbb{P} \left[ |T_1 - T_2| \leq \varepsilon T/2 \right] \leq 1/16 \) follows along the same lines.

5.1.4 Space Complexity. The expected size of sets \( S_0, S_1 \), and \( S_2 \) collected in pass 1 is \( mp = O(m \log n / \sqrt{Tm}) \). The expected number of cycles stored in pass 2 is \( 4T/p^3 = \Theta(T^{1/4}) \). Note that \( T^{1/4} = O(m/T^{1/4}) \) since a pair of disjoint edges can be involved in at most 2 distinct cycles and thus \( T \leq 2m^2 \). We run four instances of the oracle (one per edge) for each 4-cycle stored. Note, that all oracles use \( S_1 \cup S_2 \) without creating individual copies of this data. However, each instance also keeps track of \( O(W/M) \) counters (where \( W \) and \( M \) are defined in Section 3). Note, that \( W = |E_1| < m \), since the number of 4-cycles on \( e \) is at most \( m \), and \( M \leq (1/p)^2 = \Theta(\sqrt{T}) \). Thus, the extra space per classifier is \( O(m/\sqrt{T}) \) and the total space used by the algorithm is \( \tilde{O}(m/T^{1/4}) \).

We now state our final result.

**Theorem 5.3.** There exists an \( \tilde{O}(m/T^{1/4}) \) space algorithm that takes three passes over an arbitrary order stream and returns a \((1 + \epsilon)\) multiplicative approximation to the number of 4-cycles in the graph with probability at least \( 3/4 - 1/\text{poly} \ n \).

By running \( \Theta(\log 1/\delta) \) copies of the algorithm in parallel and taking the median of their outputs, we can increase the success probability to \( 1 - \delta \), where \( \delta \in (0, 1) \).

5.2 Two-pass \( \tilde{O}(m^{3/2}/T^{3/4}) \)-space Algorithm

In this section, we describe an algorithm which distinguishes between graphs with 0 and \( T \) 4-cycles using \( \tilde{O}(m^{3/2}/T^{3/4}) \) space and two passes over the stream. Our algorithm relies on the following result from extremal graph theory [25, 31]:

**Lemma 5.4.** Let \( G = (V, E) \) be graph. If \( |E| \geq 2|V|^{3/2} \), then \( G \) contains a 4-cycle.

5.2.1 Algorithm. In the first pass, we sample edges with probability \( p = c/T^{1/2} \), where \( c \) is a sufficiently large constant. Call the set of sampled edges \( S \). We show that if the input graph has \( T \) 4-cycles, then with constant probability \( S \) will contain a pair of vertex-disjoint edges \( e \) and \( e' \) which belong to the same 4-cycle. Let \( V_S \) be the set of vertices involved in the edges we sampled and \( G_S = (V_S, E_S) \) be the subgraph induced by \( V_S \). Then \( G_S \) contains a 4-cycle involving \( e \) and \( e' \). However, we do not know which two edges in \( S \) form this special pair. Therefore, in the second pass we keep collecting edges in \( G_S \) until we find a 4-cycle.

Thus, our algorithm is as follows:

**Pass 1:** Sample edges with probability \( p = c/T^{1/2} \), call set of sampled edges \( S \).

**Pass 2:** Collect edges with both endpoints in \( V_S \), until you find a 4-cycle or reach the end of the stream.

5.2.2 Space Complexity. By Chernoff bound, \( |S| = \tilde{O}(m/\sqrt{T}) \) with high probability and thus \( |V_S| = \tilde{O}(m/\sqrt{T}) \). If there are no 4-cycles in \( G_S \), it follows from Lemma 5.4 that the number of edges collected in two passes is \( |E_S| < 2|V_S|^{3/2} = \tilde{O}(m^{3/2}/T^{3/4}) \). Otherwise, after collecting at most 2\(|V_S|^{3/2} = \tilde{O}(m^{3/2}/T^{3/4}) \) edges in \( G_S \) we find a 4-cycle.

5.2.3 Correctness. Below, we prove that if the input graph contains \( T \) 4-cycles, then \( G_S \) contains a 4-cycle with constant probability.

Let an edge be heavy if the number of 4-cycles containing that edge is at least \( 100 \sqrt{T} \). We call a pair of disjoint edges heavy if it contains a heavy edge and light otherwise. Let \( D_L \) be the set of light pairs involved in 4-cycles and \( D_L = |D_L| \). Lemma 5.1 states that at least \( T/4 \) cycles contain no more than one heavy edge. Hence, \( D_L \geq T/4 \).

**Lemma 5.5.** If \( G \) contains \( T \) 4-cycles, in pass 1 we sample at least one pair in \( D_L \) with constant probability.

Proof. Let \( X_q \) be 1 if both edges of pair \( q \in D_L \) were sampled and 0 otherwise. Then \( X = \sum_{q \in D_L} X_q \) is the number of light pairs that are sampled. Note that \( \mathbb{E}[X] = p^2 |D_L| = \Omega(1) \) when constant \( c \) in the sampling probability is sufficiently large.

\[
\mathbb{P}[X \leq 0] = \mathbb{P}[\mathbb{E}[X] - X \geq \mathbb{E}[X]] \leq \frac{1}{10}
\]
where the last inequality follows from Chebyshev’s inequality if $\forall [X] \leq \mathbb{E}[X]^2 / 10$. We now prove this bound on the variance.

$$
\mathbb{V}[X] = \sum_{q \in \mathbb{D}_L} \mathbb{V}[X_q] + \sum_{q \in \mathbb{D}_L, q' \neq q} \text{cov}[X_q, X_{q'}] \\
\leq \sum_{q \in \mathbb{D}_L} \mathbb{E}[X_q^2] + \sum_{q \in \mathbb{D}_L, q' \neq q} \mathbb{E}[X_q X_{q'}] \\
\leq \sum_{q \in \mathbb{D}_L} p^2 + \sum_{q \in \mathbb{D}_L, t \in \mathbb{D}_L} \sum_{q' \neq q} p^3 \\
\leq D_L p^2 + \sum_{q \in \mathbb{D}_L} 220 \sqrt{T} p^3 \\
= D_L p^2 + 220 D_L \sqrt{T} p^3 \\
= (p^2 D_L)^2 / 10
$$

where equation 3 follows from the fact that any light pair $q \in \mathbb{D}_L$ intersects at most $2 \cdot 110 \sqrt{T}$ other pairs in $\mathbb{D}_L$ and equation 4 from the fact that $D_L \geq T/4$ and from constant $c$ in the sampling probability being sufficiently large. The theorem then follows from Chebyshev’s inequality.

We now state our final result.

**Theorem 5.6.** There exists an $\tilde{O}(m^{3/2}/T^{3/4})$ space algorithm that takes two passes over an arbitrary order stream and distinguishes between graphs with 0 and $T$ 4-cycles with probability at least $2/3$.

By running $\Theta(\log 1/\delta)$ copies of the algorithm in parallel and taking the majority answer, we can increase the success probability to $1 - \delta$, where $\delta \in (0, 1)$.

### 5.3 One-pass algorithm when $T$ is large

We now revisit the approach presented in Section 4.2. There we proved it yields a polylogarithmic space algorithm in the adjacency list model if $T$ is sufficiently large. Here we show that it can be implemented in the arbitrary order model if we use more space.

**Theorem 5.7.** If $T = \Omega(n^2/k^2)$, there exists a single pass algorithm for $1 + \epsilon$ approximating $T$ using $\tilde{O}(\epsilon^{-2} n)$ space in the arbitrary order model.

The only deviation in the analysis of Section 4.2 is that in the arbitrary order model, we need to maintain $3n$ counters when computing the basic estimator, one for each of $A_1, B_1, C_1$ for each vertex $t$. When edge $(u, v)$ arrives we increment $A_t, B_t, C_t, A_{uv}, B_{uv}, C_{uv}$ by adding the appropriate combination of entries of $\alpha$ and $\beta$. Note that this algorithm would also work in the dynamic graph setting where edges are both inserted and deleted.

### 5.4 Lower Bound

**Theorem 5.8.** For any $m$ and $T \leq m^2$, there exists $m' = \Theta(m)$ and $T' = \Theta(T)$ such that any arbitrary order 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 $\Omega(m/T^{1/2})$ space.

**Proof.** For the proof we use a reduction from Disjointness communication complexity problem: Alice holds a binary string $s_l$ of length $r$ and Bob holds a string $s^l$ of the same length. Using multi-way communication, they must determine whether there exists an index $x$ such that $s_{l,x} = s^l_{x,1} = 1$, answering 1 if there is and 0 otherwise. The communication complexity of this problem is $\Omega(r) [19, 30]$. We embed an instance of the disjointess problem in a graph which has no 4-cycles if the output is 0, and $T$ cycles if it is 1. Therefore, any algorithm that can distinguish between 0 and $T$ 4-cycles (in particular, any algorithm for counting 4-cycles) would provide a protocol for the communication problem, with communication complexity equal to the space cost of the algorithm.

Let the vertices of the graph be

- Two special vertices $u$ and $w$.
- $n/k$ groups of $k$ vertices $V_1, V_2, \ldots, V_n/k$.

Alice and Bob hold strings of length $n/k$. For every $s_{1}^{l} = 1$ in Alice’s string, she inserts $k$ edges between $u$ and $V_i$. For every $s_{2}^{l} = 1$ in Bob’s string, he inserts $k$ edges between $V_j$ and $w$. This graph has $\Theta(n)$ edges. If the strings are disjoint, the graph consists of two disjoint stars and thus has no 4-cycles. If there is an index $x$, such that $s_{1}^{l,x} = s_{2}^{l,x} = 1$, then we have set $V_k$ with edges to both $u$ and $w$. Thus, it contains $\binom{k}{2} = \Theta(k^2)$ 4-cycles. The bound is then $\Omega(n/k) = \Omega(m/T^{1/2})$.

ACKNOWLEDGMENTS

This work was partially supported by NSF grants CCF-1934846, CCF-1908849, and CCF-1637536.

REFERENCES


A OMITTED PROOFS

Proof of Lemma 5.1. First, note that at the end of the algorithm, 

\[ A = \sum_{v \in R_2} w^i(v) \]

\[ A_H = \sum_{v \in V_H} w^i(v) \]

\[ V_H^t \] is the set of heavy vertices in \( R_2 \)

\[ a(v) = w^i(v) \] for each \( v \in V_H^t \)

\[ A_L = \sum_{v \in V_L \cap R_2} w^i(v) \]

By an application of the Chernoff bound, it is easy to show that for all \( v \in V_H \), \( w^i(v) \geq \sqrt{M}/2 \) and for all \( v \in V_L \), \( w^i(v) \leq 2\sqrt{M} \) with high probability. Thus, we condition on this event.

Consider \( v \in V_H \) and observe that \( \mathbb{E}_{w} \left[ w^i(v) \right] = w^i(v)p \). Furthermore, by the Chernoff bound,

\[ \mathbb{P} \left[ |w^i(v) - w^i(v)p| \geq (\varepsilon/2)w^i(v)p \right] \leq 2\exp(-\varepsilon^2 w^i(v)p/(12\lambda)) \leq 1/poly(n) \]

where the last inequality follows from the assumption \( w^i(v) \geq \sqrt{M}/2 \). Hence, with high probability,

\[ A_H/p = \frac{1}{p} \sum_{v \in V_H} w^i(v) = (1 \pm \varepsilon/2) \sum_{v \in V_H} w^i(v) \]

Now consider \( V_L \) and observe that \( \mathbb{E}_{w} \left[ A_L \right] = p \sum_{v \in V_L} w^i(v) \). If \( W \leq M \) then \( \mathbb{E}_{w} \left[ A_L \right] \leq M \) and by the Chernoff bound,

\[ \mathbb{P} \left[ |A_L - \mathbb{E}_{w} \left[ A_L \right]| \geq epM/2 \right] \leq 2\exp(-\varepsilon^2 Mp/(24\sqrt{M})) = 1/poly(n) \]

where the inequality follows from the assumption that \( w^i(v) \leq 2\sqrt{M} \). Similarly, it follows that if \( W > 2M \),

\[ \mathbb{P} \left[ |A_L - \mathbb{E}_{w} \left[ A_L \right]| \geq epW/2 \right] \leq 1/poly(n) \]

and if \( W < M/2 \),

\[ \mathbb{P} \left[ |A_L - \mathbb{E}_{w} \left[ A_L \right]| \geq epM/2 \right] \leq 1/poly(n) \]

We now prove the three statements of the lemma.

a. From eq. 5 and 6 it follows that if \( W \leq M \), then with high probability \( \tilde{W} = W \pm \varepsilon M/2 = \varepsilon M/2 \), where we set \( \varepsilon = 1/2 \).

b. From eq. 5 and 7 it follows that if \( W > 2M \), then with high probability \( \tilde{W} = W - \varepsilon M/2 - \varepsilon W/2 > 2M(1 - \varepsilon/2) - \varepsilon M/2 > M \) assuming \( \varepsilon < 2/3 \). Thus, by contrapositive, \( \tilde{W} < M \) implies \( W \leq 2M \).

c. From eq. 5 and 8 it follows that if \( W < M/2 \), then with high probability \( \tilde{W} = W + \varepsilon M/2 + \varepsilon M/2 < M \) assuming \( \varepsilon \leq 1/2 \). Thus, by contrapositive, \( \tilde{W} > M \) implies \( W \geq M/2 \).

\[ \square \]

Proof of Lemma 1.1. Let \( T_1 \) be the number of cycles with \( i \) heavy edges. Let \( T_{2,a} \) be the number of cycles with \( 2 \) heavy edges that are vertex disjoint and let \( T_{2,b} = T_2 - T_{2,a} \). We note that there are at most \( 4V^2/\eta \) heavy edges, as each cycle contains 4 edges. Therefore, there are at most \( 16T/\eta^2 \) different pairs of heavy edges. In particular, there are at most \( 32T/\eta^2 \) cycles containing a pair of vertex disjoint heavy edges, as each such pair may participate in at most two distinct cycles. Therefore,

\[ T_{2,a} + T_3 + T_4 \leq 32T/\eta^2 \]

(9)
For each pair of vertices \( uv \), let \( g_{uv} \) be the number of wedges with end points \( u \) and \( v \) with no heavy edges, and \( b_{uv} \) be the number of wedges of that form with two heavy edges. Note that

\[
\sum_{uv \in V(G)^2} b_{uv} \leq 16T/\eta^2.
\]

Then we have:

\[
T_{2,b} = \sum_{uv \in V(G)^2} g_{uv}b_{uv}
\]

\[
\leq \sum_{uv \in V(G)^2} b_{uv} + \sum_{uv \in V(G)^2: g_{uv} \geq 2} g_{uv}b_{uv}
\]

\[
\leq 16T/\eta^2 + \left( \sum_{uv \in V(G)^2: g_{uv} \geq 2} g_{uv}^2 \right)^{1/2} \left( \sum_{uv \in V(G)^2: g_{uv} \geq 2} b_{uv}^2 \right)^{1/2}
\]

where the last line follows from Hölder’s inequality. Then,

\[
\sum_{uv \in V(G)^2: g_{uv} \geq 2} g_{uv}^2 \leq 4 \sum_{uv \in V(G)^2: g_{uv} \geq 2} \left( g_{uv} \right)^2 \leq 8T
\]

and

\[
\sum_{uv \in V(G)^2} b_{uv}^2 = \sum_{uv \in V(G)^2: g_{uv} \geq 2} 2b_{uv} + \sum_{uv \in V(G)^2} b_{uv}
\]

\[
\leq 4T_4 + 16T/\eta^2
\]

\[
\leq 128T/\eta^2 + 16T/\eta^2,
\]

where the last line follows from Eq. 9. Hence,

\[
T_{2,a} + T_{2,b} + T_3 + T_4
\]

\[
\leq 32T/\eta^2 + 16/\eta^2 + \sqrt{8T} \cdot \sqrt{128T/\eta^2 + 16T/\eta^2}
\]

\[
= T\left(48/\eta^2 + 24\sqrt{2}/\eta\right)
\]

\[
< 82T/\eta. \quad \square
\]