The First International Conference on
Machine Learning and Knowledge Discovery (MLKD 2024)
Amirkabir University of Technology, December 18-19, 2024

On the Space Complexity of Counting Triangles Using Oracles

Hossein Jowhari* Arash Rahmati†

Abstract

In this paper we study data stream algorithms for approximating the number of triangles under the assumption that the algorithm has unlimited access to an oracle that answers certain queries about the input graph. We present both upper bounds (algorithms) and space lower bounds for this problem. More specifically, our bounds apply to algorithms that use a degree oracle (given a vertex, the oracle provides the degree of the queried vertex) and an edge-triangle oracle. Given the query edge $\{u, v\}$, an edge-triangle oracle answers whether the edge $\{u, v\}$ participates in a triangle or not. In addition, we implement two single-pass algorithms (and the associated oracles) in both the edge-arrival and the vertex-arrival models to evaluate their performance on real-world datasets. Despite the inaccuracies of the oracles used in our experiments, our study shows that they can improve the performance of state-of-the-art triangle counting algorithms on some real-world graphs.

Keywords: Triangle Counting, Learning Augmented Algorithms, Data Stream Algorithms

1 Introduction

Graphs are useful structures that are used to model real-world problems by representing relationships between entities. Computing the structural properties of a graph that models a real-world problem provides insight and facilitates the analysis of the problem. An important structural property of a graph, with many applications, is its number of triangles, i.e. the number of triplets of vertices in which each vertex is connected to the other two vertices.

The problem of counting the number of triangles in a graph, represented as a stream of edges, was first introduced in 2002 [2] and has ever since been widely studied in the streaming model due to its numerous applications in spam detection, community mining, link prediction, etc [1]. The problem has been studied under the assumption of both single-pass [8] and multi-pass algorithms [3]. Some researchers have focused on the insertion-only model where edges cannot be deleted once they are inserted [16, 14], whereas others have worked on the dynamic setting [19, 20], which handles edge deletions as well. The order of edges is divided into three main categories: arbitrary order, random order, and the vertex arrival. The most challenging case is the arbitrary order where edges may arrive in any order that is decided by an adversary. In the random order model, as the name suggests, the edges randomly appear in the stream. Finally, in the vertex-arrival model, all edges incident to a certain vertex appear together and thus every edge is seen twice in the stream. Lastly, while most of the research in the field focuses on approximation with multiplicative error, there are papers that propose algorithms with additive error [17].

Recently, the problem of counting triangles has been explored within the framework of learning-augmented algorithms. In this context, it is assumed that a (noisy) predictor or oracle is available which is capable of answering specific queries about the input, while the data stream is processed. In practice, the predictor is constructed by training a machine learning model on previous instances of the problem or by utilizing other features of the input. In some cases, a previously stored instance of the problem can directly fulfill the role of the predictor. Although the predictor may give incorrect or approximate information about the input, assuming its error is bounded, it has been practically demonstrated that the additional information can enhance the efficiency of the algorithms.

In this direction, Chen et al. [6] has initiated the study of learning-augmented algorithms for estimating the number of triangles in the data stream model. They have given one-pass algorithms that use a heavy-edge oracle (the oracle predicts whether an edge is involved in many triangles or not). Specifically, they have shown that using a heavy-edge oracle one can achieve space bounds that is not possible without the oracle assumption. They have also presented experimental results that show the practicality of their approach over some real-world datasets.

Inspired by the Chen et al's work, in this paper we study a related oracle called an "edge-triangle oracle" which decides whether an edge participates in at least one triangle or not. We also consider the "degree oracle" which provides the degree of a vertex once asked (the degree oracle has been considered before in a work by McGregor et al. [14]). We revisit previous algorithms and observe that in some cases the space usage is improved (or the number of passes is reduced) when an

*Faculty of Applied Mathematics, K. N. Toosi University of Technology, jowhari@kntu.ac.ir
†Faculty of Applied Mathematics, K. N. Toosi University of Technology, arashrahmati@email.kntu.ac.ir

edge-triangle or a degree oracle is available. We also show space lower bounds which indicate that even with unlimited access to a perfect oracle one needs a certain amount of space to get an estimate of the number of triangles. Our theoretical bounds are presented in details in Section 1.3. On the experimental side, we show the edge-triangle oracle can be helpful in practice. More specifically, our experiments show that, in both edge-arrival and vertex-arrival models, using an edge-triangle oracle in combination with the approach used in [6], we can achieve better estimates in low memory settings.

1.1 Preliminaries

Given a graph $G(V, E)$ with m edges and n vertices, we focus on (ϵ, δ)-estimation of $T(G)$, the number of triangles in the graph using randomized algorithms with high probability. In other words, we need to guarantee $\mathbb{P}[|\hat{T} - T(G)| > \epsilon T(G)] < \delta$ where $\hat{T}$ is our estimate. We only work on the insertion-only model in this paper. Let $\Delta(G)$, Δ_E, and Δ_V denote the maximum degree of G, the maximum number of triangles on any given edge, and the maximum number of triangles on any given vertex respectively. Formal definitions of the oracles in previous works and the one we introduce for the first time are as follows.

Definition 1 (Degree Oracle) *Given a vertex u, a degree oracle, Deg-Oracle(.), returns the degree of the vertex, i.e. Deg-Oracle$(u) = d_u$.*

Definition 2 (Heavy-Edge Oracle) *If t_e denotes the number of triangles incident to the edge $e = \{u, v\}$, a heavy-edge oracle, Heavy-Oracle(e), returns TRUE if $t_e > \theta$ and FALSE otherwise, where θ is the heaviness threshold.*

Definition 3 (Edge-Triangle Oracle) *Given an edge e, an edge-triangle oracle, Edge-Tri-Oracle(e), returns TRUE if $t_e > 0$ and FALSE otherwise.*

Due to the randomized nature of our algorithms, we repeatedly need to use Chebyshev's inequality and Chernoff bounds to complete our proofs of correctness. For brevity and avoiding repetitions, we will use Median-of-Means Improvement from [5] which is stated as follows.

Lemma 1 (Median-of-Means Improvement [5]) *There is a universal positive constant c such that the following holds. Let the random variable X be an unbiased estimator for a real quantity Q. Let $\{X_{ij}\}_{i\in[t],j\in[k]}$ be a collection of independent random variables with each X_{ij} distributed identically to X, i.e. $\mathbb{E}[X_{ij}] = Q$, where*

$$t = c \log \frac{1}{\delta}, \quad \text{and} \quad k = \frac{3\,\mathrm{Var}[X]}{\epsilon^2(\mathbb{E}[X])^2}.$$

Let $Z = median_{i\in[t]} \left(\frac{1}{k}\sum_{j=1}^{k} X_{ij}\right)$. Then, we have

$$\mathbb{P}\left(|Z - Q| \geq \epsilon Q\right) \leq \delta,$$

i.e., Z is an (ϵ, δ)-estimate for Q.

Thus, if an algorithm can produce X using s bits of space, then there is an (ϵ, δ)-estimation algorithm using

$$O\left(s \cdot \frac{\mathrm{Var}[X]}{(\mathbb{E}[X])^2} \cdot \frac{1}{\epsilon^2} \log \frac{1}{\delta}\right)$$

bits of space.

1.2 Previous Work

We have summarized the space bound of the existing algorithms in the arbitrary order for one-pass and multi-pass settings in Table 1 and Table 2. We use $\tilde{O}$ to suppress logarithmic factors that often appear in randomized algorithms due to independent repetitions. We use T rather than $T(G)$ as it is obvious what we are referring to.

Ref.	Space	Oracle
[15]	$\tilde{O}(m(\epsilon^{-2}\Delta_E/T + \varepsilon^{-1}/\sqrt{T}))$	-
[9]	$\tilde{O}(\epsilon^{-2}m\Delta^2/T)$	-
[16]	$\tilde{O}(\epsilon^{-2}m\Delta/T)$	-
[10]	$\tilde{O}(\epsilon^{-1}m\sqrt{\Delta_E/T})$	-
[10]	$\tilde{O}(\epsilon^{-1}m\sqrt{\Delta_V/T})$	-
[8]	$\tilde{O}(\epsilon^{-2}(m/T)(\Delta_E + \sqrt{\Delta_V}))$	-
[6]	$\tilde{O}(\epsilon^{-1}(m/\sqrt{T} + \sqrt{m}))$	heavy-edge

Table 1: Previous *one-pass, arbitrary order* triangle counting algorithms

Apart from dependency on T and m, we observe that all of the previous algorithms, except for [6], are dependent on Δ_E (Note that $\Delta_E \leq \Delta$ and $\Delta_E \leq \Delta_V$.) Even the last work took advantage of a heavy-edge oracle to remove dependency on Δ_E. A heavy edge in their work is one that is involved in at least θ triangles.

Ref.	Space	Pass	Oracle
[7]	$\tilde{O}(\epsilon^{-2.5}m/\sqrt{T})$	2	-
[14]	$\tilde{O}(\epsilon^{-2}m/\sqrt{T})$	2	-
[14]	$\tilde{O}(\epsilon^{-2}m^{3/2}/T)$	3	degree
[3]	$\tilde{O}(\epsilon^{-2}m^{3/2}/T)$	4	-

Table 2: Space bound of the previous *multi-pass, arbitrary order* triangle counting algorithms

From table 2, the algorithm of [3] can also run in 3 passes if given access to a degree oracle. It is important to note that depending on whether $T > m$ or not, algorithms with space complexity of either $m^{3/2}/T$ or $m/\sqrt{T}$ can outperform the other one.

1.3 Our Results

Our results are of both theoretical and practical interests. In theory, our upper bounds (algorithms) are summarized in table 3. In practice, we run a modified version of the algorithms of [6] where we have added an oracle that detects the *unimportant* edges (edges that are not involved in any triangle.) The oracle works by either directly using the previous instances of the graph or training machine learning models as a guide to guess the unimportant edges. Our experimental results show that edge-triangle oracles that directly use previous instances of the graph to guess the unimportant edges are accurate enough in terms of both relative error and the variance compared to the one-pass streaming algorithm of Chen et al. However, when we use machine learning models to train the oracles on Reddit Hyperlinks dataset, the trained oracles are not accurate enough to improve our estimates of T.

Space	Pass	Order	Oracle
$\tilde{O}(\epsilon^{-2}\Delta)$	1	arbitrary	edge-triangle
$\tilde{O}(\epsilon^{-2}m^{3/2}/T)$	2	random	degree
$\tilde{O}(\epsilon^{-2}\sqrt{m})$	3	arbitrary	edge-triangle degree

Table 3: Our triangle counting algorithms in theory

We also prove the following lower bounds.

1. Any one pass streaming algorithm that approximates the number of triangles in a graph within $1+\epsilon$ factor and has unlimited access to an edge-triangle oracle needs $\Omega(\epsilon^{-1}\Delta)$ bits of space, where $\epsilon \in (0, 1/4]$.

2. Any one pass randomized streaming algorithm that distinguishes triangle-free graphs from graphs with at least $\Omega(n)$ triangles, and $\Omega(n^2)$ edges, and has unlimited access to a degree oracle requires $\Omega(n^2)$ bits of space.

2 Algorithms

In this section we present our algorithms in theory for the triangle counting problem that assume the existence of an oracle.

2.1 Degree oracle

With a degree oracle, we build up on the algorithm of [3]. Let T_e be the number of triangles on edge e where the maximum degree of the triangle is not on this edge. More precisely,

$$T_{e=\{u,v\}} = \{s \in V(G) \mid \{s,u\} \in E(G) \wedge \{s,v\} \in E(G) \wedge d_s > \max\{d_u, d_v\}\}.$$

The comparison of vertices are based on their degrees, and if degrees are equal, the names of vertices are compared lexicographically. Thus, $\sum_{e\in E(G)} T_e = T$ and it has been proved [3] that $T_e \leq \sqrt{2m}$. At this point, we propose algorithm 1.

Algorithm 1 $\tilde{O}(\varepsilon^{-2}m^{3/2}/T)$-Space, 2-Pass Algorithm

Pass 1:
Select one edge $e_1 = \{u, v\}$ using reservoir sampling.
Pass 2:
WLOG, we assume $d_u > d_v$ using the degree oracle
$(Y, {d_u}', flag, e_2) \leftarrow (0, 0, 0, \emptyset)$
for edges in the form $e_i = \{u, w_i\}$ **do**
  if Deg-Oracle$(w_i) > d_u$ **then**
    ${d_u}' \leftarrow {d_u}' + 1$
    if coin$(1/{d_u}')$= *"head"* **then**
      $(e_2, flag) \leftarrow (e_i, 0)$
    else
      if e_i completes the wedge e_1e_2 **then**
        $flag \leftarrow 1$
if $flag = 1$ **then** $Y \leftarrow 2{d_u}'$
$X \leftarrow mY$
return X

In algorithm 1, the outcome of coin(p) is "*head*" with probability p. We use ${d_u}'$ to denote the number of neighbors of u with degrees higher than d_u. In the first pass, the algorithm samples one edge using reservoir sampling uniformly at random [21]. In the second pass, the algorithm samples only 1 neighbor of u. However, the degree oracle helps us sample what we call a "good" neighbor, one that has a higher degree than those of u and v. Note that in [3], they had to sample multiple neighbors since they could not detect "good" neighbors and thus the space complexity of one independent execution of their algorithm was not $O(1)$. Finally, the algorithm waits for the arrival of the third, completing edge of the triangle after w_i (the "good" neighbor) has been sampled. We will now calculate the expectation and the variance of the output random variable of algorithm 1 and prove theorem 2.

Theorem 2 *Given graph G as a stream of edges in the random order model, there is a 2-pass algorithm that benefits from a degree oracle and approximates $T(G)$ within $1+\epsilon$ factor using $\tilde{O}(\frac{m^{3/2}}{\epsilon^2 T})$ space.*

Proof. If we assume that the algorithm has sampled e_i in the first pass, we can calculate $\mathbb{E}[Y \mid e_i]$. This is the part we need to assume the random order since the third edge may not appear after w_i is sampled if an adversary has ordered the edges. So, the fraction $1/2$ assumes the order is random and the last edge may appear after the second edge with probability 0.5.

$$\mathbb{E}[Y \mid e_i] = \frac{T_{e_i}}{{d_u}'}\frac{1}{2}2{d_u}' = T_{e_i}$$

So,

$$\begin{aligned}\mathbb{E}[X] = m\mathbb{E}[Y] &= m(\frac{1}{m}\mathbb{E}[Y \mid e_1] + \cdots + \frac{1}{m}\mathbb{E}[Y \mid e_m]) \\ &= m\frac{1}{m}\sum_{e\in E} T_e = T\end{aligned}$$

We now proceed to calculate an upper bound on the variance.

$$\begin{aligned}\mathrm{Var}[X] = \mathrm{Var}[mY] &= m^2\mathrm{Var}[Y] \le m^2\mathbb{E}[Y^2] \\ &= m^2\frac{1}{m}\sum_{e\in E}\mathbb{E}[Y^2|e] \\ &= m\sum_{e_i\in E}\frac{T_{e_i}}{{d_u}'}\frac{1}{2}2{d_u}'2{d_u}' \\ &\le 2m\max_{u\in V}\{{d_u}'\}\sum_{e_i\in E} T_{e_i} \\ &= 2m\sqrt{2m}T = O(m^{3/2}T)\end{aligned}$$

Now that we have the upper bound on the variance of our unbiased estimator and that we know each execution requires $O(1)$ space, we use lemma 1 to calculate the number of repetitions required as follows.

$$\begin{aligned}\text{Repetitions} = \frac{3\mathrm{Var}[X]}{\varepsilon^2\mathbb{E}[X]^2}c\log(\frac{1}{\delta}) &= O(\frac{m^{3/2}T}{\varepsilon^2T^2}c\log(\frac{1}{\delta})) \\ &= \tilde{O}(\frac{m^{3/2}}{\varepsilon^2T})\end{aligned}$$

□

2.2 Edge-triangle oracle

The work of [16] proposed a single-pass streaming algorithm in the arbitrary order model that approximates T in $\tilde{O}(\epsilon^{-2}m\Delta/T)$ space with multiplicative error. Having access to an edge-triangle oracle as described in definition 3, it is easy to see how we can prove the following theorem. In fact, all that is required is that the first edge r_1 in their work should be sampled from what we call "good" edges that participate in at least one triangle. Then, m would be replaced by m', which is the number of edges that are involved in at least one triangle. Knowing $m' \le 3T$ would then lead to the following theorem.

Theorem 3 *Given graph G as a stream of edges, there is a 1-pass algorithm that benefits from an edge-triangle oracle and approximates $T(G)$ within $1+\epsilon$ factor using $\tilde{O}(\frac{\Delta(G)}{\epsilon^2})$ space.*

Proof. Follows from the algorithm in Pavan et. al. [16]. □

2.3 Degree & Edge-triangle oracles

In this section, we explore the power of having access to both oracles of edge-triangle and degree. Similar to what we proposed in section 2.1, we use the degree oracle to sample "good" neighbors. Furthermore, we sample a "good" edge in the first pass similar to section 2.2. In order to make it work in the arbitrary order model, we require an additional pass over the stream. The pseudocode of this algorithm is illustrated in algorithm 2 where we assume $d_u > d_v$.

Algorithm 2 $\tilde{O}(\varepsilon^{-2}\sqrt{m})$-Space, 3-Pass Algorithm

Pass 1:
$(m', e_1) \leftarrow (0, \emptyset)$
for edges e_i in the stream **do**
  if Edge-Tri-Oracle(e_i) = *"TRUE"* **then**
    ▷ e_i is in a triangle.
    $m' \leftarrow m' + 1$
    $e_1 = \{u, v\} \leftarrow e_i$ w.p. $(1/m')$
Pass 2:
$(Y, {d_u}', e_2) \leftarrow (0, 0, \emptyset)$
for edges in the form $e_i = \{u, w_i\}$ **do**
  if Deg-Oracle(w_i) $> d_u$ **then**
    ${d_u}' \leftarrow {d_u}' + 1$
    $e_2 \leftarrow e_i$ w.p. $(1/{d_u}')$
Pass 3:
if $\{w_i, v\} \in E$ **then** ▷ the wedge is in a triangle.
  $Y \leftarrow {d_u}'$
return $X = m'Y$

Theorem 4 *Given graph G as a stream of edges, there is a 3-pass algorithm that benefits from a degree oracle and an edge-triangle oracle, and approximates $T(G)$ within $1+\epsilon$ factor using $\tilde{O}(\epsilon^{-2}\sqrt{m})$ space.*

Proof. Following the same procedure as in theorem 2 we see that:

$$\mathbb{E}[X] = T, \qquad \mathrm{Var}[X] = O(m'\sqrt{m}T) = O(\sqrt{m}T^2)$$

Since the space complexity of one execution of the algorithm is $O(1)$, the space complexity is the number of repetitions required to achieve $(1+\epsilon)$ approximation,

which is:

$$\text{Repetitions} = \frac{3\text{Var}[X]}{\varepsilon^2 \mathbb{E}[X]^2} c \log(\frac{1}{\delta}) = O(\frac{\sqrt{m}T^2}{\varepsilon^2 T^2} c \log(\frac{1}{\delta})) = \tilde{O}(\frac{\sqrt{m}}{\varepsilon^2})$$

□

3 Lower bounds

Our lower bounds are based on reductions from the well-known indexing problem. In the $INDEX(n)$ problem, two parties Alice and Bob communicate to answer a question about their input. Alice holds a bit vector $x \in \{0,1\}^n$ and Bob holds a query index $i \in \{1, \cdots, n\}$. Bob wants to know the value of $x[i]$. We have the following fact regarding the communication complexity of the Indexing problem.

Fact 5 *[12] Any one-way randomized communication protocol for $INDEX(n)$ problem that succeeds with probability at least 3/4 requires $\Omega(n)$ bit of communication.*

We first show a lower bound for streaming algorithms that use an edge-triangle oracle.

Theorem 6 *Let $\epsilon \in (0, \frac{1}{4}]$. Any 1-pass streaming algorithm that approximates the number of triangles in a graph within $1+\epsilon$ factor and has unlimited access to an edge-triangle oracle needs $\Omega(\frac{\Delta}{\epsilon})$ bits of space.*

Proof. We use a reduction from a variant of the indexing problem. In the $INDEX(n,k)$ problem, Alice holds an n-bit length string A (with exactly $n/2$ number of 1's in it) and Bob holds a query subset $Q \subseteq [n] = \{1, \cdots, n\}$ with $|Q| = k$. It is promised that A is fixed on Q (i.e. for all $i \in Q$ and $j \in Q$ we have $A[i] = A[j]$.) Starting with Alice, the players exchange messages and at the end Bob should answer whether $A[Q] = 1$ or $A[Q] = 0$. By a straightforward reduction from the regular Indexing problem, it can be shown that any 1-way communication protocol for $INDEX(n,k)$ requires $\Omega(n/k)$ bits of communication.

We show a one-pass streaming algorithm for approximating the number of triangles can be used as a 1-way protocol for the $INDEX(n,k)$ problem. Let d be a positive integer. Consider an instance of $INDEX(n,k)$ problem where k is a positive integer whose value will be determined later. Given their input, Alice and Bob define the edge set of a graph G with vertex set $V(G) = X \cup Y \cup Z$ where $|X| = |Y| = n$ and $|Z| = dk$. First we define the edge set of Alice. For each $i \in [n]$, if $A[i] = 1$, Alice adds the edge (x_i, y_i). At the other side, Bob having the query subset $Q \subseteq [n]$, for each $i \in Q$, he connects both x_i and y_i to d new vertices in Z.

Note that G has $2n + dk$ vertices and since A has $\frac{n}{2}$ number of 1's, the graph G has $\frac{n}{2} + 2dk$ edges. From the construction, we get that if $A[Q] = 1$ then G will have exactly dk triangles otherwise it is triangle-free.

Now, given graph G, we construct an augmented graph G' in the following way. For each edge (u,v) in G, we add a new vertex uv and add the edges (uv,u) and (uv,v) to G. The main observation here is that every edge in G' necessarily participates in a triangle and thus one cannot gain any extra information if an edge-trianlge oracle is called on G'. Also note that $\Delta(G') = 2\Delta(G) \leq 2(d+1)$, and by the definition of G', we have $T(G') = |E(G)| + T(G)$. Consequently if $A[Q] = 1$ we have $T(G') = \frac{n}{2} + 3dk$ otherwise $T(G') = \frac{n}{2} + 2dk$.

Setting k and d such that $kd = \epsilon n$, we get that a $1+O(\epsilon)$ approximation algorithm for counting triangles can distinguish between the case where $T(G') = \frac{n}{2} + 3\epsilon n$ and $T(G') = \frac{n}{2} + 2\epsilon n$. Consequently, it can solve the $INDEX(n,k)$ instance. This proves a space lower bound of $\Omega(n/k) = \Omega(\frac{d}{\epsilon})$ for approximating the number of triangles as stated in the theorem. □

It is known that one-pass streaming algorithms for distinguishing triangle-free graphs from graphs with at least $T = \Omega(n)$ triangles and m edges require $\Omega(m)$ bits of space [4]. Here we extend this hardness result to streaming algorithms that have unlimited access to a degree oracle for the case where $m = \Omega(n^2)$.

Theorem 7 *Any 1-pass randomized streaming algorithm that distinguishes triangle-free graphs from graphs with at least $\Omega(n)$ triangles and $m = \Omega(n^2)$ edges, and has unlimited access to a degree oracle requires $\Omega(n^2)$ bits of space.*

Proof. For the sake of the proof, we define the 2-player communication game problem $RBI(n,d)$. In this game, Alice holds a regular bipartite graph $G = (A \cup B, E)$ where $|A| = |B| = n$ and every vertex in G has degree d. In the other side, Bob holds the pair $a \in A$ and $b \in B$. Bob wants to know if the edge (a,b) exists in G or not. Using a reduction from $INDEX(nd)$, we can show any randomized one-way protocol for $RBI(n,d)$ requires $\Omega(nd)$ bits of communication. To see this, let $x \in \{0,1\}^{nd/2}, i \in \{1, \cdots, nd/2\}$ be an instance of the Indexing problem where n is an even number. Alice converts her bit-vector x into a regular bipartite graph as follows. First Alice divides x into d parts $x_1 \cdots x_d$. Each part has length $n/2$. For each $j \in \{1, \cdots, d\}$, given x_j Alice creates a (non-overlapping) perfect matching M_j between the sets $A = \{a_1, \cdots, a_n\}$ and $B = \{b_1, \cdots, b_n\}$ as follows. This ensures that, at the end, each vertex in the graph has degree d. First we describe the perfect matching associated with x_1. The other matchings are described similarly as shown below.

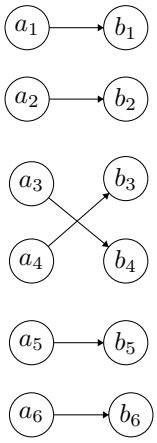

Figure 1: An example illustrating the perfect matching for $x_1 = 101$ in the proof of Theorem 7.

Suppose $x_1[1] = 1$. In this case we put the edges (a_1, b_1) and (a_2, b_2). Otherwise, if $x_1[1] = 0$, we put the edges (a_1, b_2) and (a_2, b_1). Following this pattern, if $x_1[k] = 1$ we put the edges (a_{2k-1}, b_{2k-1}) and (a_{2k}, b_{2k}). Otherwise, when $x_1[k] = 0$, we put the edges (a_{2k-1}, b_{2k}) and (a_{2k}, b_{2k-1}). Figure 1 shows an example for the bit string $x_1 = 101$

Now for the perfect matching M_2, we first perform circular shifts of the vertices in B (we do it twice) and then use x_2 to create the matching M_2 between A and the shifted B in the same way we used x_1 to create M_1. Generally, to create M_j, first we do $2j-2$ circular shifts of the vertices in B and then put the matching between A and the shifted B in a similar manner. Putting the perfect matchings $M_1, \cdots, M_d$, together, we get a bipartite d-regular graph $G = (A \cup B, E)$. It is easy to see that if Bob wants to know the value of $x[i]$, by querying an edge in the graph G, he can find out the answer.

Now we establish a reduction from $RBI(n, d)$ to our problem of interest. Given an instance of $RBI(n, d)$, Bob adds a series of vertices and edges to the underlying graph $G = (A \cup B, E)$ as follows. Having the query (a, b), Bob adds the set of vertices $U = \{u_1, \cdots, u_n\}$. For each u_i, he connects u_i to both $a \in A$ and $b \in B$. Note that if the edge (a, b) exists, this will create n triangles. At the same time this raises the degrees of a and b to $d + n$. To make these vertices indistinguishable from the rest of the vertices in $A \cup B$ (in the terms of the degree), Bob also adds two sets of vertices $A' = \{a'_1, \cdots, a'_n\}$ and $B' = \{b'_1, \cdots, b'_n\}$. For each i, Bob connects a'_i to all the vertices in $A/\{a\}$. In the same manner, for each i, Bob connects b'_i to all the vertices in $B/\{b\}$. Now each vertex in $A \cup B$ has degree $d+n$. The vertices in $A' \cup B'$ have degree $n-1$ and the vertices in U are of degree 2. Let G' be the resulting graph. Note that G' has $5n$ vertices and has $m = \Theta(nd+n^2)$ number of edges. More important, the degree of each vertex in G' is known by Alice and Bob. Therefore, using a degree oracle does not reveal any new information about the other party's input. As said above, if the edge (a, b) exists in G, the resulting graph G' will have exactly n triangles otherwise it will be triangle-free. The statement of the theorem follows from the lower bound for $RBI(n, d)$ and setting $d = O(n)$. □

4 Experimental Results

4.1 Datasets

We use three datasets described as follows.

- Oregon[1]: This dataset consists of 9 graphs $\{\#1, ..., \#9\}$ of Autonomous Systems (AS) peering information inferred from Oregon route-views between March 31 2001 and May 26 2001 on the internet [13].
- As-Caida[2]: The dataset contains 122 CAIDA Autonomous System (AS) graphs, from January 2004 to November 2007. Each file contains a full AS graph derived from a set of RouteViews BGP table snapshots. We will use the data from 2006 and 2007 in our experiments. There are 52 instances of the graph of 2006 $\{\#1, ..., \#52\}$, and there are 46 instances of the graph of 2007 $\{\#1, ..., \#46\}$.
- Reddit Hyperlinks[3]: The hyperlink network [11] represents the directed connections between two subreddits (a subreddit is a community on Reddit, a social network). Subreddit 300-dimensional embeddings are also available for most of the nodes. The network is extracted from publicly available Reddit data of 2.5 years from Jan 2014 to April 2017. The subreddit-to-subreddit hyperlink network is extracted from the posts that create hyperlinks from one subreddit to another. We say a hyperlink originates from a post in the source community and links to a post in the target community.

The statistics of the first instances of Oregon and As-Caida along with the statistics of Reddit Hyperlinks are summarized in table 4.

4.2 How to make oracles

We build oracles in two ways similar to [6].

- In Oregon and As-Caida, we directly use previous data. In fact, we look at the first instance of each graph and memorize those important (heavy) edges. We also memorize unimportant edges (edges not involved in any triangle).

[1] https://snap.stanford.edu/data/Oregon-1.html
[2] https://snap.stanford.edu/data/as-Caida.html
[3] https://snap.stanford.edu/data/soc-RedditHyperlinks.html

- In Reddit Hyperlinks, since we have node representations, $f(u)$, we train machine learning models. More specifically, we train a linear regression model as our heavy-edge oracle, and a logistic regression as our edge-triangle oracle that outputs yes/no. Similar to [6], the labels are calculated using the exact count, and the features of edges are made using node embeddings as follows.

$$f(e = \{u, v\}) = \underbrace{(f(u), f(v), \|f(u) - f(v)\|_1, \|f(u) - f(v)\|_2)}_{\text{602-dimensional vector}}$$

par	O#1	C2006#1	C2007#1	Reddit
n	10670	21202	24013	35775
m	22002	42925	49332	124330
T	17144	30433	40475	406391
Δ	2312	2381	2377	2336
Δ_E	526	578	602	725
Δ_V	3431	3530	4590	31967

Table 4: Statistics of the graph datasets (O:Oregon, C:Caida)

4.3 One-Pass Algorithms of Interest

We compare our algorithms with the work of Chen et al., algorithms 1 and 4 in [6]. This is because they have already shown that over most of the datasets (Oregon, Caida-2006, and Caida-2007) their algorithms (in both edge-arrival and vertex-arrival models) work more accurately than previous best one pass streaming algorithms (ThinkD [19] and WRS [18].) So at least on these datasets, the algorithms of Chen et al. are now the state of the art. Furthermore, we emphasize that the proposed algorithms in this work follow Chen et al's idea of using heavy-edge oracles except that here we also add the oracles for the edge-triangle predictions. This idea, in addition to their idea of keeping important (heavy) edges, helps improve the accuracy and more importantly lowers the variance of our estimates over multiple executions.

Note that these practical algorithms have not been discussed in the theoretical side of our paper; thus, we refer the reader to [6] for details. However, the main idea is to use an oracle to keep heavy edges separately and sample light edges. Then, triangles, based on their heavy and light edges, are divided into different groups; each of them will have its unique counter. Each counter is finally divided by the probability of the triangle being sampled and counted. At the end, all the final values are summed to estimate the total number of triangles in the graph.

4.4 Comparison of Algorithms

We use Oregon#1, Caida-2006#1, and Caida-2007#1 as the oracles to detect the heavy and the unimportant edges. The input of the algorithms are Oregon#4, Caida-2006#30, and Caida-2007#25. The error is $\left|(1 - \hat{T}/T) \times 100\right|$. We run each algorithm 50 times and calculate the median relative error. The colors around each line is $\pm$ 1 STD of the relative errors in 50 independent executions.

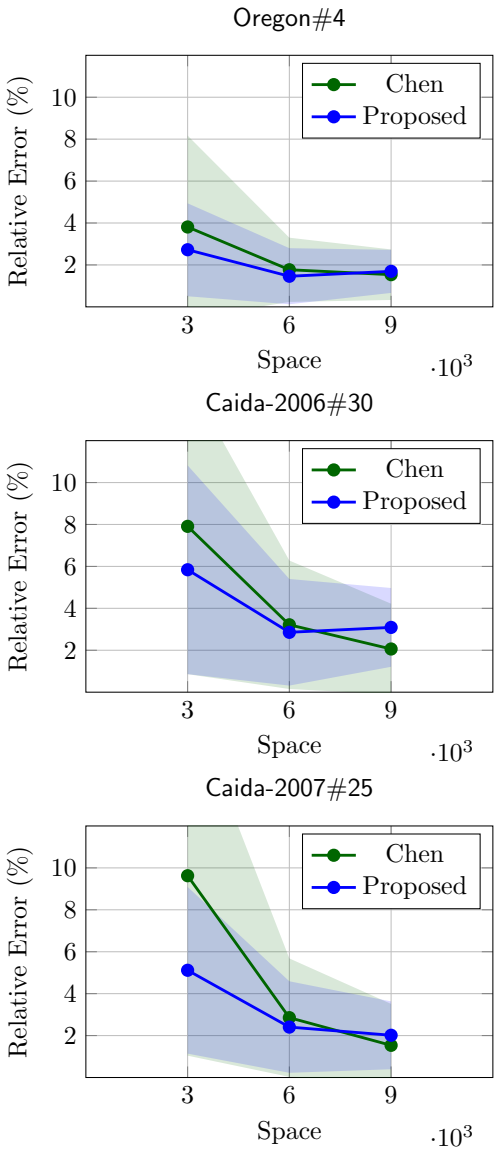

Figure 2: Comparison (edge-arrival) Oregon and CAIDA datasets

We observe from Figure 2 that the proposed algorithm in all cases has lower variance especially when the space is very limited. The median relative errors of the proposed algorithm, except for the space of 9000 edges,

have been lower compared to the work of Chen et al. [6]. It is reasonable that the proposed algorithm cannot outperform the work of Chen et al when we are allowed to sample many edges since the oracles are not perfect, and we lose some triangles by mistakenly removing some edges involved in triangles. Despite a higher error at 9000 sampled edges, the variance of the proposed algorithm is still lower.

For the Oregon dataset, we evaluate the proposed algorithm's performance over other instances as well in figure 3. In this experiment the space allowed for the two algorithms is 3000 edges. Thus, both algorithms have been given the same resources. The vertical lines in figure 3 show ± 1 STD of the relative errors in 50 independent executions.

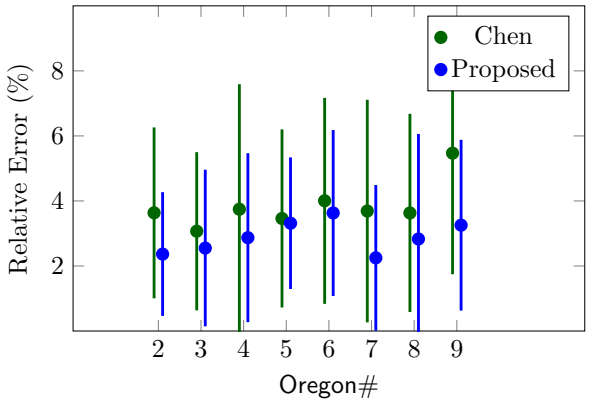

Figure 3: Comparison of algorithms on other instances of Oregon (edge-arrival)

To further evaluate our idea of removing unimportant edges, we implemented the first algorithm of [6] which is in the vertex arrival model. Vertex-arrival model is a simpler setting in which vertices arrive with their neighbors one at a time. The errors become smaller in this setting and using our idea of removing unnecessary edges makes the errors even smaller as seen in Figure 4 for datasets Oregon#4, CAIDA2006#30 and CAIDA2007#25.

Since each algorithm is executed 50 times to calculate the median error, the variance of the output values are illustrated in figure 5. The values of variance is calculated for both algorithms at 6000 edges, accompanied by those in the edge-arrival model for the Oregon#4 dataset. This also confirms the reduction in variance by removing the unnecessary edges. Similar reductions will be observed at other space capacities, which we have excluded for brevity.

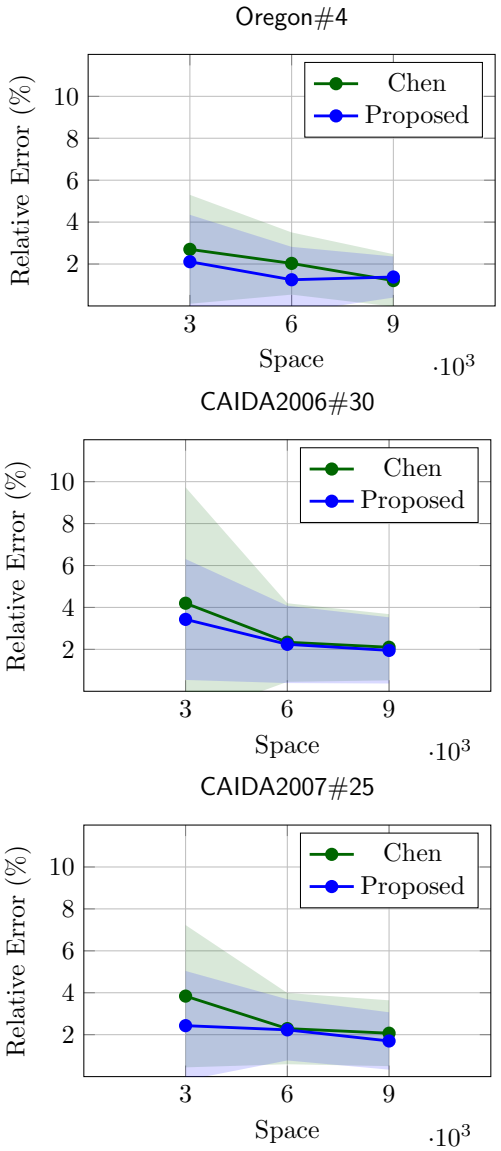

Figure 4: Comparison (vertex-arrival) OREGON and CAIDA datasets

We finally present our results on Reddit Hyperlinks based on the oracles we train. As mentioned earlier, we combine features of the nodes to create features for the edges. However, out of the 124K edges, we can only make features for 103K edges due to lack of features for some nodes. Next, a logistic regression is trained to predict whether or not an edge is involved in at least one triangle. A linear regression model is also trained to predict which edges are heavy (involved in many triangles). The first half of the edges serve as the training set for the models. The results are shown in figure 6.

As we observe from figure 6, the proposed algorithm does not reduce the variance, nor does it result in better median errors in most cases. In the following section, we will discuss why this has happened.

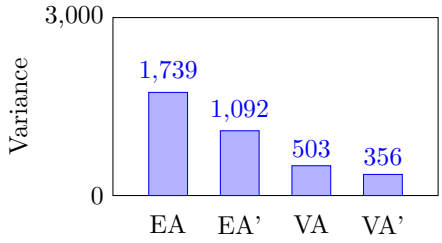

Figure 5: OREGON#4, Variance of outputs for Space = 6000 edges. (EA: Chen's Edge-Arrival Alg, VA: Chen's Vertex-Arrival Alg, ': removing unnecessary edges)

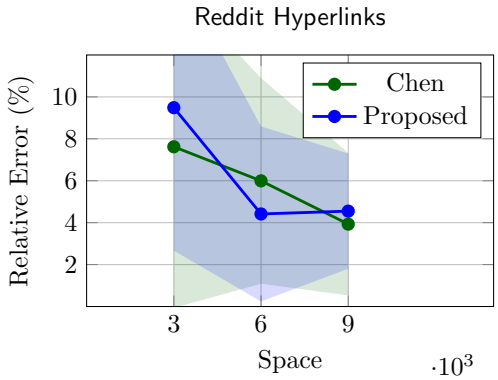

Figure 6: Comparison (ML-based oracles)

4.5 Discussion

In the previous section we showed how using an edge-triangle oracle can improve the accuracy of the learning-augmented algorithms in [6]. In practice, our main idea is to remove the unnecessary edges that do not participate in any triangle. One might argue that such removals require extra space to memorize the unimportant edges; however, this is merely a demonstration of how removal of the unnecessary edges can lead to improvements of state-of-the art algorithms. In fact, oracles can be machine learning models that do not necessarily occupy much space but are capable of deciding whether or not an edge is involved in a triangle with high confidence. Such models are only trained on the first few instances of the graph and are subsequently saved and benefited from in the future instances to detect the unimportant edges.

Additionally, we note that the improvement of the proposed method relies highly on detecting edges that do not participate in any triangles. When we directly use previous data, the edge-triangle oracle, although not perfect, is accurate enough to help us outperform the algorithm of Chen et al. However, when we use logistic regression on the first half of the edges in Reddit Hyperlinks, our approach does not lead to a better estimate. This happens since the model does not predict the unimportant edges accurately enough. When directly using the previous data, the oracles successfully remove over 8000 unnecessary edges, whereas when using logistic regression, the oracle only removes 3000 unnecessary edges. Considering the 120K edges of the graph of Reddit Hyperlinks, removing 3000 edges is negligible and hence does not not lead to better estimates. We present the accuracy of the oracles in Tables 5 and 6. There, Positive means the oracle has declared an edge to be part of a triangle while Negative means the oracle has declared an edge unnecessary. NI means the oracle cannot make a prediction on that edge because it either had not appeared in previous instances or does not have a feature vector (representation) for the regression model.

Oracle	TP	TN	FP	FN	NI	Acc
O#1	11061	8534	522	517	2113	%94
C6#1	16299	16836	1734	1589	9627	%90
C7#1	19102	19906	1528	1540	9434	%92

Table 5: Oracle Accuracy (Direct Use of Previous Data)

Oracle	TP	TN	FP	FN	NI	Acc
LR	81431	3148	16039	3147	20565	%81

Table 6: Logistic Regression Accuracy

5 Conclusion

In this paper, we studied the problem of Triangle Counting in the streaming model with the assumption that the algorithm has access to oracles that provide information about the input. Here we have only considered the insertion-only model where the edges are inserted but not deleted. We have presented both theoretical bounds and some experimental results.

In theory, we proposed three algorithms using one, two, and three passes respectively. The first algorithm takes advantage of an edge-triangle oracle, a new oracle introduced in this paper. The second algorithm assumes access to a degree oracle, and the third one is given access to both oracles. In terms of the order of edges, except for the second algorithm which is in the random order model, the other two algorithms are in arbitrary order. The space complexity of the algorithms are $\tilde{O}(\epsilon^{-2}\Delta)$, $\tilde{O}(\epsilon^{-2}m^{3/2}/T)$, and $\tilde{O}(\epsilon^{-2}\sqrt{m})$ respectively. We also proved space lower bounds for 1-pass triangle counting algorithms that benefit from edge-triangle and the degree oracles.

In our experiments, we have implemented the idea of removing edges that do not participate in any triangles by calling oracles. These oracles can directly use previous instances of the graph, or they can be built by training machine learning models. Our experiments show that in the autonomous systems' datasets, due to

the high accuracy of edge-triangle oracles that directly use previous data, the proposed algorithms can lead to improvements over state-of-the-art (the algorithms of Chen et al [6]). This happens specifically when space usage is low both in the edge-arrival and the vertex-arrival model. However, when we use machine learning models to train oracles, the accuracy is not high enough to outperform the work of Chen et al.

Finally we want to highlight two theoretical problems that are left open in this area.

1. Can we show an improved algorithm or space lower bound for multi-pass algorithms that use an edge-triangle oracle?

2. What is the space complexity of algorithms for approximating the number of triangles that use a degree oracle?

References

[1] M. Al Hasan and V. S. Dave. Triangle counting in large networks: a review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(2):e1226, 2018.

[2] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *SODA*, volume 2, pages 623–632, 2002.

[3] S. K. Bera and A. Chakrabarti. Towards tighter space bounds for counting triangles and other substructures in graph streams. In *34th Symposium on Theoretical Aspects of Computer Science*, 2017.

[4] V. Braverman, R. Ostrovsky, and D. Vilenchik. How hard is counting triangles in the streaming model? In F. V. Fomin, R. Freivalds, M. Z. Kwiatkowska, and D. Peleg, editors, *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, volume 7965 of *Lecture Notes in Computer Science*, pages 244–254. Springer, 2013.

[5] A. Chakrabarti. Data stream algorithms lecture notes, 2020.

[6] J. Y. Chen, T. Eden, P. Indyk, H. Lin, S. Narayanan, R. Rubinfeld, S. Silwal, T. Wagner, D. P. Woodruff, and M. Zhang. Triangle and four cycle counting with predictions in graph streams. *arXiv preprint arXiv:2203.09572*, 2022.

[7] G. Cormode and H. Jowhari. A second look at counting triangles in graph streams (corrected). *Theoretical Computer Science*, 683:22–30, 2017.

[8] R. Jayaram and J. Kallaugher. An optimal algorithm for triangle counting in the stream. *arXiv preprint arXiv:2105.01785*, 2021.

[9] H. Jowhari and M. Ghodsi. New streaming algorithms for counting triangles in graphs. In *Computing and Combinatorics: 11th Annual International Conference, COCOON 2005 Kunming, China, August 16–19, 2005 Proceedings 11*, pages 710–716. Springer, 2005.

[10] N. Kavassery-Parakkat, K. M. Hanjani, and A. Pavan. Improved triangle counting in graph streams: power of multi-sampling. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 33–40. IEEE, 2018.

[11] S. Kumar, W. L. Hamilton, J. Leskovec, and D. Jurafsky. Community interaction and conflict on the web. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 933–943. International World Wide Web Conferences Steering Committee, 2018.

[12] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, 1997.

[13] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187, 2005.

[14] A. McGregor, S. Vorotnikova, and H. T. Vu. Better algorithms for counting triangles in data streams. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 401–411, 2016.

[15] R. Pagh and C. E. Tsourakakis. Colorful triangle counting and a mapreduce implementation. *Information Processing Letters*, 112(7):277–281, 2012.

[16] A. Pavan, K. Tangwongsan, S. Tirthapura, and K. Wu. Counting and sampling triangles from a graph stream. *Proc. VLDB Endow.*, 6(14):1870–1881, 2013.

[17] C. Seshadhri, A. Pinar, and T. G. Kolda. Fast triangle counting through wedge sampling. In *Proceedings of the SIAM Conference on Data Mining*, volume 4, page 5, 2013.

[18] K. Shin. Wrs: Waiting room sampling for accurate triangle counting in real graph streams. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 1087–1092. IEEE, 2017.

[19] K. Shin, J. Kim, B. Hooi, and C. Faloutsos. Think before you discard: Accurate triangle counting in graph streams with deletions. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 141–157. Springer, 2018.

[20] L. D. Stefani, A. Epasto, M. Riondato, and E. Upfal. Triest: Counting local and global triangles in fully dynamic streams with fixed memory size. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(4):1–50, 2017.

[21] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.