Better Algorithms for Counting Triangles in Data Streams[*]

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

Sofya Vorotnikova
University of Massachusetts
svorotni@cs.umass.edu

Hoa T. Vu
University of Massachusetts
hvu@cs.umass.edu

ABSTRACT

We present space-efficient data stream algorithms for approximating the number of triangles in a graph up to a factor $1+\epsilon$. While it can be shown that determining whether a graph is triangle-free is not possible in sub-linear space, a large body of work has focused on minimizing the space required in terms of the number of triangles T (or a lower bound on this quantity) and other parameters including the number of nodes n and the number of edges m. Two models are important in the literature: the *arbitrary order* model in which the stream consists of the edges of the graph in arbitrary order and the *adjacency list order* model in which all edges incident to the same node appear consecutively. We improve over the state of the art results in both models. For the adjacency list order model, we show that $\tilde{O}(\epsilon^{-2}m/\sqrt{T})$ space is sufficient in one pass and $\tilde{O}(\epsilon^{-2}m^{3/2}/T)$ space is sufficient in two passes where the $\tilde{O}(\cdot)$ notation suppresses log factors. For the arbitrary order model, we show that $\tilde{O}(\epsilon^{-2}m/\sqrt{T})$ space suffices given two passes and that $\tilde{O}(\epsilon^{-2}m^{3/2}/T)$ space suffices given three passes and oracle access to the degrees. Finally, we show how to efficiently implement the "wedge sampling" approach to triangle estimation in the arbitrary order model. To do this, we develop the first algorithm for ℓ_p sampling such that multiple independent samples can be generated with $O(\text{polylog}\, n)$ update time; this primitive is widely applicable and this result may be of independent interest.

Categories and Subject Descriptors

F.2 [**Analysis of Algorithms & Problem Complexity**]

Keywords

data streams; triangles; clustering coefficients

1. INTRODUCTION

Estimating the number of triangles in a graph is a canonical problem in the data stream model of computation. The problem was first considered by Bar-Yossef et al. [6] nearly fifteen years ago and a significant body of work has since been devoted to designing more efficient and ingenious algorithms for the problem in both the single-pass [1, 2, 6, 9, 10, 22, 24, 29, 31, 37, 38, 41] and multi-pass models [8, 16, 28]. For a survey of existing graph stream algorithms, including triangle counting, see [32].

There appears to be two main reasons for the high level of interest in the problem. First, the number of triangles in a network and related quantities such as the *transitivity* or *global clustering coefficient* (the fraction of length two paths that are included in a triangle) play an important role in the analysis of real-world networks. Popular examples include motif detection in protein interaction networks [34], uncovering hidden thematic structure in the web graph [17], analysis of social networks [44], and the evaluation of large graph models [30]. Following Kutzkov et al. [9, 29], we direct the interested reader to Tsourakakis et al. [42] for an excellent overview of these and other applications. Second, the problem has a rich theory. The best exact algorithm in the RAM model runs in $O(m^{2\omega/(\omega+1)})$ time [2] where $\omega \leq 2.3728$ is the matrix multiplication exponent and m is the number of edges. This year, Eden et al. [18] designed the first sub-linear time algorithm. Finally, there are connections to a range of important problems in the field of fine-grained complexity [23]. Since many of the real world graphs of interest are massive, it is natural that the problem has also been studied in the appropriate computation models, e.g., the MapReduce model [40] and other parallel models [5, 7], external memory models [4], and the data stream model (see above for the long stream of references).

Two data stream models have been considered in the literature on triangle counting: the *arbitrary order* model in which the stream consists of the edges of the graph in arbitrary order and the *adjacency list order* model in which all edges incident to the same node appear consecutively.[1] Of the algorithms designed in both models, some are suitable when there are many triangles whereas others dominate if there are only a few triangles. We next discuss the state-of-the-art results and these trade-offs in the context of our new results.

1.1 Our Results and Related Work

In discussing our results and the related work we use n to denote the number of nodes in the input graph, m to be the number of edges, and T is the number of triangles in the graph.

Approximate Triangle Counting. We present four main algorithms that $(1+\epsilon)$-estimate the number of triangles: two algorithms for each data stream model (arbitrary order and adjacency list order)

[*]Work supported by NSF Awards CCF-0953754, IIS-1251110, CCF-1320719, and a Google Research Award.

PODS'16, June 26-July 01, 2016, San Francisco, CA, USA
© 2016 ACM. ISBN 978-1-4503-4191-2/16/06…$15.00
DOI: http://dx.doi.org/10.1145/2902251.2902283

[1]The adjacency list order model is closely related to the vertex arrival model that has been considered in the context of finding matchings in the data stream model [20, 26] and row-order arrival model consider in the context of linear algebra problems [12, 19].

where one is suitable for processing graphs with many triangles (in particular, when $T \geq m$) and the other is suitable for processing graphs with fewer triangles (i.e., $T \leq m$).[2] Specifically:

1. *Adjacency List Model:* We present a single-pass algorithm using $\tilde{O}(\epsilon^{-2}m/\sqrt{T})$ space and a two-pass algorithm using $\tilde{O}(\epsilon^{-2}m^{3/2}/T)$ space. We show that the space can be further reduced if we only need to distinguish triangle-free graphs from those with at least T triangles.

2. *Arbitrary Order Model:* We present a two-pass algorithm using $\tilde{O}(\epsilon^{-2}m/\sqrt{T})$ space and a three-pass algorithm using $\tilde{O}(\epsilon^{-2}m^{3/2}/T)$ space. However, the second algorithm assumes that we have oracle access to the degrees of the nodes.

It can be argued that using only $\approx m/\sqrt{T}$ space has become a natural goal in the context of estimating the number of triangles. In particular, Cormode and Jowhari [16] showed that any constant pass algorithm in the arbitrary order model required this amount of space when $m = \Theta(n\sqrt{T})$ and there is an existing two-pass algorithm that returns a 3-approximation using this amount of space. Furthermore, Jha et al. [22] showed that this space was sufficient for additively approximating T. Unfortunately, Braverman et al. [8] showed it was insufficient for achieving multiplicative approximation via a *single-pass* algorithm in the arbitrary order model. The significance of our results is showing that $\approx m/\sqrt{T}$ space is sufficient for $1+\epsilon$ approximation if we are given a single pass over a stream in adjacency list order or two passes over a stream in arbitrary order.

However, it is possible to improve upon $\approx m/\sqrt{T}$ space when T is large and our other two algorithms do just this. At a high level, the main difference between the two types of algorithms we present is as follows. The $m/\sqrt{T}$ dependence arises when we focus on distinguishing between edges that are involved in many triangles and those that are not, whereas the $m^{3/2}/T$ dependence arises when we distinguish between high and low degree nodes. The idea of distinguishing heavy and light edges or nodes is an important idea in the non-streaming work by Alon et al. [2], Eden et al. [18], Chiba and Nishizeki [11], among others. The main challenge in our work arises from the constraints of the data stream model. This necessitates new algorithms and new notions of heavy and light that may also depend on the ordering of the stream.

Wedge Sampling. An important technique developed in the context of triangle counting is that of *wedge sampling* [27, 39]. A wedge is a length-two path in a graph and the goal is to sample wedges uniformly from the graph. We use W to denote the number of wedges in a graph[3] and note that the global clustering coefficient equals $3T/W$. In the final section of this paper, we show the following:

3. *Fast Wedge Sampling via ℓ_p Sampling:* We show how to sample s independent wedges in the data stream model with $\tilde{O}(1)$ update time. We do this by first proving a new result for ℓ_p sampling, an important data stream primitive. See Section 4.2 for more details. Our second algorithm for the arbitrary order model is based on wedge sampling.

[2]For context, it can be shown that $T = O(m^{3/2})$ and there are graphs where $T = \Omega(m^{3/2})$ [2].

[3]For context, it can be shown that W is at least $m^2/(2n)$ (for $m \geq n$) and can be $\Omega(mn)$.

1.1.1 *Comparison to Previous Algorithms*

Adjacency List Model. Prior to our work, the state-of-the-art algorithms in the adjacency list model were a three-pass algorithm using $\tilde{O}(\sqrt{m} + \epsilon^{-2}m^{3/2}/T)$ space that was presented by Kolountzakis et al. [28] and a one-pass algorithm using $\tilde{O}(\epsilon^{-2}W/T)$ space that was presented by Buriol et al. [10]. Note that $\tilde{O}(\epsilon^{-2}W/T)$ is $\tilde{O}(\epsilon^{-2}mn/T)$ in the worst case and therefore the former algorithm may use significantly less space when $m \ll n^2$ but this improvement is limited by the additive $\sqrt{m}$ term. Our one-pass algorithm improves upon the Buriol et al. result for $T \leq n^2$ and improves upon the Kolountzakis et al. result for all values of T. This second observation follows because if $T \geq m$ then $m/\sqrt{T} \leq \sqrt{m}$ and if $T \leq m$ then $m/\sqrt{T} \leq m^{3/2}/T$. Our two-pass algorithm also dominates the algorithm of Kolountzakis et al. for all T since it uses only two passes rather than three passes and there is no additive $\sqrt{m}$ term in the space use.

Arbitrary Order Model. Prior to our work, the state-of-the-art results include another one-pass algorithm by Buriol et al. [10] that used $\tilde{O}(\epsilon^{-2}mn/T)$ space. Pavan et al. [38] showed that the dependence on n could be replaced by the maximum degree and Pagh and Tsourakakis [37] showed that this could be replaced by a dependence on the maximum number of triangles using a single edge. The other most relevant result is a one pass algorithm by Jha et al. [22] that returns an additive $\pm\epsilon W$ estimate (equivalently, an additive ϵ approximation of the transitivity coefficient) using $\tilde{O}(\epsilon^{-2}m/\sqrt{T})$ space. In establishing our first result for the arbitrary order model, we first revisit the Jha et al. algorithm to show that it can also be used to estimate the number of triangles multiplicatively using space that is almost identical to that required by Pagh and Tsourakakis' algorithm.

The most relevant previous work in the arbitrary order model is an $(1+\epsilon)$-approximation using $\tilde{O}(\epsilon^{-2}m/T^{1/3})$ space by Braverman et al. [8] and a $(3+\epsilon)$-approximation with $\tilde{O}(\epsilon^{-4.5}m/\sqrt{T})$ space by Cormode and Jowhari [16]. Note that Cormode and Jowhari initially claimed that their algorithm returned a $1+\epsilon$ but this claim is incorrect [13].[4] Subsequent to the submission of our paper, Cormode and Jowhari have independently designed a new algorithm [14]. This alternative algorithm, significantly different from their earlier algorithm and more complicated than our algorithm, uses $\tilde{O}(\epsilon^{-2.5}m/\sqrt{T})$ space.

1.2 Notation and Preliminaries

It will be convenient to assume the node set of the graph is $[n] = \{1, 2, \ldots, n\}$. Let $\Gamma(v)$ denote the neighbors of a node v and so $\deg(v) = |\Gamma(v)|$. We write (undirected) edges as sets of two nodes $\{u, v\}$ and write ordered pairs of nodes as uv. For example, $\{u, v\} = \{v, u\}$ but $uv \neq vu$. We use Δ to denote the set of triangles in the input graph and so $T = |\Delta|$. For a random variable X, we denote the expectation and variance as $\mathrm{E}[X]$ and $\mathrm{V}[X]$ respectively. $\mathbf{Bin}(n, p)$ denotes the binomial distribution with parameters n and p.

To simplify the presentation of our algorithms we adopt two main conventions that we explain here. Following Braverman et al. [8], we restrict our attention to bounding the expected space use of our randomized algorithm rather than bounding the space with high probability. Note that if the algorithm satisfies its accuracy guarantee with probability 99/100, for example, then it is straight-

[4]The error can be found in the proof of Theorem 3 [16]. Specifically, it is claimed that $Y \leq X$ and hence a lower bound on Y is a lower bound on X. However, all that can be shown is $Y \leq 3X$ and a factor of three is lost in the analysis.

forward to show that the algorithm satisfies its accuracy guarantee *and* doesn't exceed its expected space use by more than a factor 100 with probability at least $49/50$. Hence, by running a logarithmic number of copies of the algorithm in parallel, terminating any that exceed their space bound, and taking the median of the remaining estimates ensures an accurate answer with only a logarithmic space increase with $1-1/\operatorname{poly}(n)$ probability. Secondly, we parameterize our algorithms in terms of the actual number of triangles T in the graph and various quantities in the algorithm will depend on T. Obviously, we do not know T in advance (otherwise we wouldn't be trying to estimate it) but this convention is widely adopted in the literature. A natural way to formalize this is to phrase the problem as distinguishing between graphs with at most t triangles from those with at least $(1+\epsilon)t$ triangles where t is an input parameter. In practice, the quantities in the algorithm would be initialized based on a lower or upper bound (as appropriate) for the unknown quantities.

2. ADJACENCY LIST ALGORITHMS

In this model, we may assume that the stream consists of a sequence of ordered pairs xy. For each edge $\{x,y\}$, both xy and yx will be present in the stream. The promise on the ordering is that all tuples with the same first node appear consecutively in the stream. Aside from that constraint, the stream is ordered arbitrarily. For example, for the graph consisting of a cycle on three nodes $V=\{v_1,v_2,v_3\}$, a possible ordering of the stream could be

$$\langle v_3v_1, v_3v_2, v_1v_2, v_1v_3, v_2v_3, v_2v_1\rangle \ .$$

In this example, we say that the adjacency list for v_3 came first, then the adjacency list for v_1, and finally the adjacency list for v_2.

2.1 One Pass and $\tilde{O}(\epsilon^{-2}m/\sqrt{T})$ Space

Algorithm and Intuition. Define a total ordering on nodes $<_s$ based on stream ordering where $x <_s y$ if the adjacency list of x is specified before the adjacency list of y in the stream. Define

$$R_{xy} = \begin{cases} |\{z : \{x,y,z\} \in \Delta \text{ and } x <_s z <_s y\}| & \text{if } x <_s y \\ 0 & \text{if } y <_s x \end{cases}$$

and note that $\sum_{x,y} R_{xy} = T$.

The basic outline of the algorithm comprises of two interlocking parts. In the first part, we will sample each edge xy with probability p when it arrives and, until yx arrives, we count all nodes z such that $\{x,y,z\}$ forms a triangle. If we do not observe yx after xy was sampled (i.e., yx came before xy in the stream ordering) this counter will never be used. Otherwise, the counter equals R_{xy} when yx arrives. Hence, by summing these counters we get an estimator that equals $\sum_{xy} R_{xy} I[xy \text{ sampled}]$. In expectation it equals pT and has low variance if all R_{xy} are small.

The second part ensures that we estimate every R_{xy} if $R_{xy} \geq \sqrt{T}$ regardless of whether xy was sampled. This will allow us to restrict our attention to small R_{xy} in the first part of the algorithm (and hence get a good variance bound). The critical observation that allows us to estimate every large R_{xy} is as follows: when reading the neighbors of x, even if we did not sample xy, we will have probably sampled some of the edges in the set

$$\{xz : \{x,y,z\} \in \Delta \text{ and } x <_s z <_s y\}$$

if the number of edges in this set, i.e., R_{xy}, is large. Subsequently, each of these sampled edges form a triangle with the incident edges of y and the number of these triangles can be used to a) recognize R_{xy} is large and b) to estimate R_{xy}.

See Figure 1 for the detailed description of the algorithm with the appropriate bookkeeping.

Analysis. For the analysis, let $\mathcal{H}$ consist of all edges xy such that xy is defined as heavy by the algorithm. The final value of A can be written as $A = A_l + A_h$ where

$$A_l = \sum_{xy \notin \mathcal{H}} c_1(xy) \qquad \text{and} \qquad A_h = \sum_{xy \in \mathcal{H}} c_2(xy)$$

The next two lemmas establish that, with good probability, $A_l/p \approx \sum_{xy \notin \mathcal{H}} R_{xy}$ and $A_h/p \approx \sum_{xy \in \mathcal{H}} R_{xy}$.

LEMMA 1. *With probability at least* $99/100$*,*

$$A_h/p = \sum_{xy \in \mathcal{H}} R_{xy} \pm \epsilon T/2$$

and $R_{xy} \leq 2\sqrt{T}$ *for all* $xy \notin \mathcal{H}$*.*

PROOF. First note that $c_2(xy) \sim \mathbf{Bin}(R_{xy}, p)$. If $R_{xy} \geq \sqrt{T}/2$, then by an application of the Chernoff bound,

$$\Pr\left[c_2(xy) = (1 \pm \epsilon/2)pR_{xy}\right] \geq 1 - 2e^{-\epsilon^2 pR_{xy}/12} \geq 1 - 1/n^{10} \ .$$

Alternatively, if $R_{xy} \leq \sqrt{T}/2$ then $c_2(xy) < p\sqrt{T}$ with probability at least $1-1/n^{10}$. Taking the union bound over all xy establishes the lemma since

$$\sum_{xy : c_2(xy) \geq p\sqrt{T}} c_2(xy) = (1 \pm \epsilon/2)p \sum_{xy \in \mathcal{H}} R_{xy} \ .$$

□

LEMMA 2. *With probability at least* $99/100$*,*

$$A_l/p = \sum_{xy \notin \mathcal{H}} R_{xy} \pm \epsilon T/2 \ .$$

PROOF. First note that $c_1(xy) = R_{xy}$ with probability p and 0 otherwise. Furthermore, the $c_1(\cdot)$ values are independent because they each depend on whether a different tuple was sampled. Hence

$$\mathrm{E}\left[A_l\right] = p \sum_{xy \notin \mathcal{H}} R_{xy} \quad \text{and} \quad \mathrm{V}\left[A_l\right] \leq p \sum_{xy \notin \mathcal{H}} R_{xy}^2 \leq 4pT^{3/2} \ .$$

since $R_{xy} \leq 2\sqrt{T}$ for $xy \notin \mathcal{H}$. By an application of the Chebyshev bound,

$$\Pr\left[\left|A_l - \mathrm{E}\left[A_l\right]\right| \geq \epsilon pT/2\right] \leq \frac{4pT^{3/2}}{(\epsilon pT/2)^2} = \frac{16}{p\epsilon^2 T^{1/2}} \leq \frac{1}{100} \ .$$

□

We then use the above two lemmas to prove our first main result.

THEOREM 3. *There exists a* $\tilde{O}(\epsilon^{-2}m/\sqrt{T})$*-space algorithm using one pass in the adjacency list model that returns a* $(1+\epsilon)$*-approximation of* T *with probability* $49/50$*.*

PROOF. The accuracy guarantee follows from Lemmas 1 and 2. The expected space use is $\tilde{O}(pm) = \tilde{O}(\epsilon^{-2}m/\sqrt{T})$ since each edge is sampled with probability p and $\tilde{O}(1)$ bits of auxilary data is collected for each sampled node. □

2.2 Two Passes and $\tilde{O}(\epsilon^{-2}m^{3/2}/T)$ Space

Algorithm and Intuition. Define a total ordering on nodes $<_d$ based on degrees where $x <_d y$ if

$$\deg(x) < \deg(y) \ \text{ or } \ (\deg(x) = \deg(y) \text{ and } \mathrm{id}(x) < \mathrm{id}(x)) \ ,$$

```
**Algorithm** TRIANGLES1
  1. Initialize $A \leftarrow 0, S_1 \leftarrow \emptyset, S_2 \leftarrow \emptyset$ and $p \leftarrow \alpha \cdot \log n \cdot \epsilon^{-2}/T^{1/2}$ for some large constant $\alpha$.
  2. On seeing edges adjacent to $v$ in the adjacency stream:
     (a) Update auxiliary information about sampled edges:
         i. For all $ab \in S_1$: If $a, b \in \Gamma(v)$ then $c(ab) \leftarrow c(ab) + 1$
         ii. For all $av \in S_2$: Let $\mathrm{order}(av) = 1$
     (b) Sample additional edges and update estimator. For each incident edge $vu$:
         i. With probability $p$, $S_1 \leftarrow \{vu\} \cup S_1$ and set $c(vu) = 0$
         ii. With probability $p$, $S_2 \leftarrow \{vu\} \cup S_2$ and set $\mathrm{order}(vu) = 0$
         iii. Define
              $c_1(uv) := \begin{cases} c(uv) & \text{if } uv \in S_1 \\ 0 & \text{otherwise} \end{cases}$
              $c_2(uv) := |\{z : uz \in S_2, \mathrm{order}(uz) = 1, z \in \Gamma(v)\}|$
              and say $uv$ is *heavy* if $c_2(uv) \geq p\sqrt{T}$. Update the estimator as follows:
              $A \leftarrow A + \begin{cases} c_1(uv) & \text{if } uv \text{ is not heavy} \\ c_2(uv) & \text{if } uv \text{ heavy} \end{cases}$
  3. Return $A/p$
```

Figure 1: The TRIANGLES1 Algorithm. The algorithm maintains two sets of edges S_1 and S_2 where each is generated by sampling each element of the stream with probability p. For each $xy \in S_1$, we maintain a counter $c(xy)$ that counts the number of triangles $\{x, y, z\}$ where $x <_s z <_s y$. For each $xy \in S_2$, we maintain a boolean $\mathrm{order}(xy)$ that is initially 0 but is set to 1 when yx is observed; at this point we have deduced $x <_s y$. The elements in S_2 will be used to determine whether each R_{xy} is large and, if so, to estimate it. The elements of S_1 will be used to estimate the contribution of all R_{xy} that are not large.

i.e., $<_d$ is ordering the nodes by degree with ties broken by the id of the node (recall, we assume the nodes are labelled in $1, 2, \ldots, n$). For each edge $e = \{x, y\}$, define

$$R_{\{x,y\}} = |\{z : \{x, y, z\} \in \Delta\, , x <_d z, y <_d z\}|$$

and note that $\sum_{e \in E} R_e = T$.

The basic idea for the algorithm in this section is as follows: In the first pass, we generate a sample of edges S along with the degree of each endpoint of the sample edges. In the second pass, for each $e \in S$ we compute R_e and return $\sum_{e \in S} R_e$. This will equal pT in expectation and we will be able to bound the variance by first showing that $R_e \leq \sqrt{2m}$ for all $e \in E$. See Figure 2 for the detailed description of the algorithm with the appropriate bookkeeping.

Analysis. We first prove a bound on $\max R_e$ that will be required to bound the variance of our estimator.

LEMMA 4. $\max R_e \leq \sqrt{2m}$.

PROOF. Let $e = \{x, y\}$ and suppose $R_e = R_{\{x,y\}} > \sqrt{2m}$. Then $\deg(x) \geq \sqrt{2m}$. Furthermore, there exist at least $\sqrt{2m}$ nodes $z_1, z_2, \ldots$ such that $\{x, y, z_i\}$ is a triangle and $\deg(z_i) \geq \deg(x) > \sqrt{2m}$. But then

$$\deg(z_1) + \deg(z_2) + \ldots > \sqrt{2m} \cdot \sqrt{2m} = 2m$$

which is a contradiction since the sum of degrees of every node in the graph is $2m$. □

THEOREM 5. *There exists an $\tilde{O}(\epsilon^{-2} m^{3/2}/T)$-space algorithm using two passes in the adjacency list model that returns a $(1+\epsilon)$-approximation of T with probability $99/100$.*

PROOF. Consider the above algorithm and note that each edge e is contained in S with probability p and $A = \sum_{e \in S} R_e$. Hence, by appealing to Lemma 4,

$$\mathrm{E}[A] = Tp \qquad \text{and} \qquad \mathrm{V}[A] < \sum_{e \in E} R_e^2 p \leq \sqrt{2m} Tp$$

Then, by the Chebyshev bound,

$$\begin{aligned} \Pr[|A/p - T| \geq \epsilon T] \leq & (\sqrt{2m} T/p)/(\epsilon^2 T^2) \\ = & p^{-1} \epsilon^{-2} \sqrt{2m}/T \leq 1/100. \end{aligned}$$

Hence the algorithm has the desired accuracy. The expected space use is $\tilde{O}(pm) = \tilde{O}(\epsilon^{-2} m^{3/2}/T)$. □

Improved Algorithm for Testing Triangle-Freeness. We conclude this section by showing that if we are only trying to distinguish triangle-free graphs from those with at least T triangles, less space is sufficient.

THEOREM 6. *There exists an $\tilde{O}(m/T^{2/3})$-space algorithm using two passes in the adjacency list model that distinguishes triangle-free graphs from those with at least T triangles with probability* $99/100$.

The basic observation is that a graph with T triangles has at least $T^{2/3}$ edges involved in these triangles. This follows because any graph with m edges can have at most $O(m^{3/2})$ triangles (see, e.g., [2]). Hence, if each edge is sampled with probability $p = \alpha/T^{2/3}$ for some large constant $\alpha > 0$ at least one edge $\{u, v\}$ in some triangle $\{u, v, z\}$ will the sampled. We do this sampling in the first pass. Then, in the second pass of the algorithm when the

```
Algorithm TRIANGLES2
  1. First pass: Let $S = S' = \emptyset$ and $p = 200\epsilon^{-2}\sqrt{m}/T$. On seeing edges adjacent to $v$ in the stream:
     (a) For each $a \in \Gamma(v)$, with probability $p$ let $S' \leftarrow S' \cup \{va\}$ and store $\deg(v)$.
     (b) For each $bv \in S'$, let $S \leftarrow S \cup \{\{b, v\}\}$ and store $\deg(v)$.
  2. Second pass: Let $A = 0$. On seeing edges adjacent to $v$ in the stream:
     (a) For each edge $\{a, b\} \in S$ such that $a <_d v$, $b <_d v$, and $a, b \in \Gamma(v)$, $A \leftarrow A + 1$.
  3. Output: Return $A/p$.
```

Figure 2: The TRIANGLES2 Algorithm. After the first pass, S contains each edge in the graph sampled with probability p along with the degrees of the endpoints. In the second pass, we count the number of triangles $\{a, b, v\}$ where $\{a, b\} \in S$, $a <_d v$, and $b <_d v$.

neighbors of z are observed we will identify a triangle by tracking which of the sampled edges have two endpoints in $\Gamma(z)$.

3. ARBITRARY ORDER ALGORITHMS

Recall that in the arbitrary order model, the m edges of the graph may arrive in any order. It will be useful to start by briefly revisiting an algorithm by Jha, Seshadri and Pinar [22] in this model. They designed a beautifully simple algorithm for estimating the transitivity coefficient of a graph up to small additive error. We show that the algorithm can be used to estimate the number of triangles and that, if one makes a small change to their algorithm (essentially sampling edges independently rather than sampling a fixed number of edges), this facilitates a simple analysis of the algorithm that is similar to that used by Pagh and Tsourakakis [37]. While we think that simplifying the analysis and generalizing the result is valuable in its own right, our main purpose in revisiting this algorithm is that we will need to build upon it in the next section.

The single-pass algorithm is as follows:

```
1. Single Pass: $S \leftarrow \emptyset$, $A \leftarrow 0$. On the arrival of the edge $\{u, v\}$:
   (a) $S \leftarrow S \cup \{\{u, v\}\}$ with probability $p$ (to be determined).
   (b) $A \leftarrow A + |\{w : \{w, u\}$ and $\{w, v\} \in S\}|$
2. Output: Return $A/p^2$.
```

The following lemma bounds the probability that the output of the algorithm is far from T.

LEMMA 7. *Let $\tau = \sum_{e \in E} \binom{x_e}{2}$ and x_e is the number of triangles that include edge e. Then,*

$$\Pr\left[|A/p^2 - T| \geq B\right] \leq (T + \tau p)/(B^2 p^2) .$$

PROOF. Let $T_1, T_2, \ldots$ be the triangles in the graph and for each T_i let W_i be the length two path consisting of the first two edges in T_i that arrive in the stream. Let $A_i = 1$ if both edges in W_i are sampled and $A_i = 0$ otherwise. At the end of the stream $A = \sum A_i$. To analyze A, first note that $\mathrm{E}[A_i] = p^2$ and $\mathrm{V}[A_i] \leq p^2$. Furthermore, $\mathrm{Cov}[A_i, A_j] \leq \mathrm{E}[A_i A_j] = p^3$ if W_i and W_j share an edge (they can share at most one edge) and $\mathrm{Cov}[A_i, A_j] = 0$ otherwise. It follows that $\mathrm{E}[A] = Tp^2$ and

$$\begin{aligned} \mathrm{V}[A] &= \sum_{i \in T} \mathrm{V}[A_i] + \sum_{i \neq j} \mathrm{Cov}[A_i, A_j] \\ &\leq Tp^2 + \sum_{e \in E} \sum_{W_i \cap W_j = \{e\}} p^3 \\ &\leq Tp^2 + p^3 \sum_{e \in E} \binom{x_e}{2} = Tp^2 + \tau p^3 . \end{aligned}$$

The lemma follows from the Chebyshev's inequality. □

The parameter τ can be bounded as $O(Tx^*)$ where x^* is the maximum number of triangles that share the same edge. Hence, it follows that the variance of the estimate decreases with x^*. The following corollary follows by setting p appropriately. The first result is an algorithm with the same space bound as Pagh and Tsourakakis' algorithm [37] and the second result reproves the result of Jha et al. [22].

COROLLARY 8. *Setting p appropriately, the above single pass algorithm returns:*

- *$(1 \pm \epsilon)T$ with probability $99/100$ using $\tilde{O}(m(\epsilon^{-1}/\sqrt{T} + \epsilon^{-2}x^*/T))$ space.*
- *$T \pm \epsilon W$ with probability $99/100$ using $\tilde{O}(m\epsilon^{-2}/\sqrt{W})$ space.*

PROOF. First note that $\tau \leq \sum_{e \in E} x_e^2/2 \leq 1.5 \cdot x^* \cdot T$ and because $\binom{x_e}{2}$ is at most the square of the degree of an endpoint of e,

$$\begin{aligned} \tau = \sum_{e \in E} \binom{x_e}{2} &\leq \sum_{u : \deg(u) > 1} (\deg(u))^3 \\ &\leq \left(\sum_{u : \deg(u) > 1} (\deg(u))^2 \right)^{3/2} = O(W^{3/2}) . \end{aligned}$$

The expected space use of the algorithm is $\tilde{O}(pm)$. Hence, setting

$$p = \alpha \cdot \max(\epsilon^{-1}/\sqrt{T}, \epsilon^{-2}\tau/T^2)$$

in the algorithm for some sufficiently large constant α and appealing to Lemma 7 with $B = \epsilon T$ gives the first result. Similarly, setting $p = \alpha \cdot \epsilon^{-2}/\sqrt{W}$ in the algorithm and $B = \epsilon W$ gives the second result. □

3.1 Two Passes and $\tilde{O}(\epsilon^{-2}m/\sqrt{T})$ Space

From the analysis of the above one-pass algorithm, it is evident that the space required is very sensitive to the existence of edges that are involved in many triangles. In this section, we address this by considering two types of edges, *light* edges that are only involved in a small number of triangles and *heavy* that are involved in a large number of triangles. Using two passes, we estimate the number of triangles where every edge is light separately from the number of triangles with at least one heavy edge.

An oracle. Edges are characterized as heavy or light by an oracle defined by the following random process:

1. Sample each node z of the graph with probability
$$p = \beta\epsilon^{-2}\log n/\sqrt{T}$$
for some large constant $\beta > 0$. Let Z be the set of sampled nodes.

2. For any edge $e = \{u, v\}$, let $\tilde{x}_e = |\{z \in Z : u, v \in \Gamma(z)\}|$ and define
$$\text{oracle}(e) = \begin{cases} \mathtt{L} & \text{if } \tilde{x}_e < p\sqrt{T} \\ \mathtt{H} & \text{if } \tilde{x}_e \geq p\sqrt{T} \end{cases} .$$

Note that once Z is chosen, the value of $\text{oracle}(e)$ is fixed for all e, including edges used to define the oracle. The following lemma establishes that x_e is relatively small if $\text{oracle}(e) = \mathtt{L}$ and relatively large if $\text{oracle}(e) = \mathtt{H}$.

LEMMA 9. *With high probability for all* $e = \{u, v\}$*,* $\text{oracle}(e) = \mathtt{L}$ *implies* $x_e \leq 2\sqrt{T}$ *and* $\text{oracle}(e) = \mathtt{H}$ *implies* $x_e \geq \sqrt{T}/2$ *.*

PROOF. First, observe that for each e, $\tilde{x}(e) \sim \mathbf{Bin}(x_e, p)$. By an application of the Chernoff bound, if $x_e \geq 2\sqrt{T}$ then
$$\Pr\left[\tilde{x}(e) < p\sqrt{T}\right] \leq \exp(-2p\sqrt{T}/3) = n^{-10} .$$
Hence, by the union bound, with high probability $\text{oracle}(e) = \mathtt{H}$ if $x_e \geq 2\sqrt{T}$ for all edges e. The second case follows similarly. □

See Figure 3 for the two-pass algorithm. In the first pass, the algorithm instantiates the above oracle and samples some additional edges S_1. In the second pass, for each new edge that is light we increment a counter by a third of the number of triangles it forms with light edges from S_1. In expectation this counter will be p^2 times the total number of triangles involving three light edges. For each new edge that is heavy, we will use the oracle to estimate the number of triangles with i heavy edges that involve this edge for $i \in \{1, 2, 3\}$. If we incremented a counter by the sum of these estimates, we would count a triangle with i heavy edges i times. But by scaling appropriately we can ensure this counter is an estimate of the total number of triangles with at least one heavy edge.

THEOREM 10. *There exists a* $\tilde{O}(\epsilon^{-2}m/\sqrt{T})$*-space algorithm using two passes in the arbitrary order model that returns a* $(1+\epsilon)$*-approximation of* T *with probability at least* $49/50$*.*

PROOF. Let $T^{\mathtt{L}}$ be the number of triangles among the set of edges $E^{\mathtt{L}} = \{e \in E : \text{oracle}(e) = \mathtt{L}\}$. Let $W_1, W_2, \ldots,$ be the length two paths in $E^{\mathtt{L}}$ that are involved in triangles in $E^{\mathtt{L}}$. Notice, there are three length two paths involved in every triangle. Let $A_i = 1$ if both edges in W_i are in S_1^L and $A_i = 0$ otherwise. Then, following the analysis in Lemma 7 and appealing to Lemma 9, we can show that $A^{\mathtt{L}}/p^2 = \sum_i A_i/(3p^2)$ equals $T^{\mathtt{L}} \pm \epsilon T/2$ with probability at least $99/100$.

Let $T^{\mathtt{H}}$ be the number of triangles in the input graph that have at least one heavy edge. For each heavy e, define x_e^i to be the number of triangles that include edge e and have exactly i heavy edges. Then
$$T^{\mathtt{H}} = \sum_{e:\text{oracle}(e)=\mathtt{H}} (x_e^1 + x_e^2/2 + x_e^3/3)$$
because every heavy triangle with i heavy edges occurs in i different summands. Since each $\tilde{x}_e^i \sim \mathbf{Bin}(x_e^i, p)$ we can apply the Chernoff bound to prove that
$$\tilde{x}_e^1 + \tilde{x}_e^2/2 + \tilde{x}_e^3/3 = (1 \pm \epsilon)p(x_e^1 + x_e^2/2 + x_e^3/3)$$
with probability at least
$$1 - 2\exp(-\epsilon^2 p(\sqrt{T}/6)/3) = 1 - 1/\text{poly}(n) .$$
We then apply the union bound.

To bound the space used by the algorithm observe that
$$\mathrm{E}\left[|S_1| + |S_2|\right] = pm + \sum_v p\deg(v) = 3pm$$
so the expected space used by the algorithm is $\tilde{O}(\epsilon^{-2}m/\sqrt{T})$ as claimed. □

4. WEDGE SAMPLING VIA ℓ_P SAMPLING

In this section, we consider the "wedge sampling approach" proposed by Schank and Wagner [39] and Kolda et al. [27]. This approach is most relevant to estimating the number of triangles when the global clustering coefficient of the graph is large. The previous work did not consider the data stream model, but we show that it is relatively straightforward to design a streaming algorithm based on their ideas. Our main results in this section are a) designing an $\tilde{O}(\epsilon^{-2}m^{3/2}/T)$-space algorithm based on the wedge sampling approach given certain assumptions and b) improving the update time of the algorithm to $\tilde{O}(1)$. To achieve the fast update time our main contribution is a new result on ℓ_p sampling, an important primitive in data stream algorithms. Throughout this section we assume that $m \geq 2n$. This guarantees there are a $\Omega(n)$ wedges in the graph.

4.1 Wedge Sampling Algorithms

The basic wedge sampling idea is to sample length-two paths *uniformly* and compute the fraction of these that are contained in a triangle. If there are W length-two paths or "wedges", then this fraction is $3T/W$ because each triangle contains three length-two paths. Thus, by an application of the Chernoff bound, if we sample $O(\epsilon^{-2}W/T)$ wedges then we can estimate T/W up to a factor $1 \pm \epsilon$ with good probability. Hence, the challenge becomes how to uniformly sample from the set of wedges in the graph and check whether there is an edge that "completes" this wedge to a triangle.

To sample and test a single wedge in the arbitrary order model we use the following two-pass algorithm:

- *First pass:* Use an ℓ_2-sampling algorithm [25] to sample a node v with probability proportional to $(1 \pm \epsilon)\deg(v)^2$. In Section 4.2, we discuss ℓ_2 sampling and show that this can be done with fast update time even if we need to sample many wedges simultaneously.

- *Second pass:* Independently sample two edges e_1 and e_2 incident to v uniformly at random using reservoir sampling [43] or ℓ_0-sampling algorithm [25]. If $e_1 = e_2$, output FAIL. Else, check if one of the edges arriving after e_1 and e_2 completes a triangle with e_1 and e_2.

The following lemma shows that the above algorithm outputs fail with probability at most $1/4$ and if not, finds a triangle with probability $\approx T/W$.

```
Algorithm TRIANGLES3
  1. Initialization: Let $Z$ be a random subset of nodes where each node is in $Z$ with probability $p = \beta\epsilon^{-2}\log n/\sqrt{T}$.
  2. First pass:
     (a) Sample a random subset of edges $S_1$ where each edge is in $S_1$ with probability $p$.
     (b) Collect all edges $S_2$ that are incident to any node in $Z$.
  3. Second pass: Let $A_{\text{L}} = A_{\text{H}} = 0$. For each edge $e = \{u, v\}$ in the stream,
     (a) For $i \in \{1, 2\}$, define $S_i^{\text{L}} = \{e \in S_i : \text{oracle}(e) = \texttt{L}\}$ and $S_i^{\text{H}} = \{e \in S_i : \text{oracle}(e) = \texttt{H}\}$ where $Z$ and $S_2$
         define oracle as above.
     (b) If oracle$(e) = \texttt{L}$ then $A_{\text{L}} \leftarrow A_{\text{L}} + |\{w : \{u,w\}, \{v,w\} \in S_1^{\text{L}}\}|/3$
     (c) If oracle$(e) = \texttt{H}$ then $A_{\text{H}} \leftarrow A_{\text{H}} + \tilde{x}_e^1 + \tilde{x}_e^2/2 + \tilde{x}_e^3/3$ where
```

$$\begin{aligned}
\tilde{x}_e^1 &= |\{w \in Z : \{u,w\}, \{v,w\} \in S_2^{\text{L}}\}| \\
\tilde{x}_e^2 &= |\{w \in Z : \{u,w\} \in S_2^{\text{L}}, \{v,w\} \in S_2^{\text{H}}\}| \\
&\qquad + |\{w \in Z : \{u,w\} \in S_2^{\text{H}}, \{v,w\} \in S_2^{\text{L}}\}| \\
\tilde{x}_e^3 &= |\{w \in Z : \{u,w\}, \{v,w\} \in S_2^{\text{H}}\}|
\end{aligned}$$

```
         We will use $\tilde{x}_e^i/(ip)$ to estimate the number of triangles involving $e$ that contain $i$ heavy edges.
  4. Output: Return $A_{\text{L}}/p^2 + A_{\text{H}}/p$
```

Figure 3: The TRIANGLES3 Algorithm. In the first pass, we sample a set of edges S_2 that will be used to instantiate our oracle and a set of edges S_1. In the second pass, counter A_L will be used to estimate the number of triangles where all edges are light and counter A_H will be used to estimate the number of triangles that include at least one heavy edge.

LEMMA 11. *Let F be the event that the above algorithms fails. Then if $m \geq 2n$, $\Pr[F] \leq 1/2$ and the probability of finding a triangle is $(1 \pm \epsilon)T/W$ conditioned on $\neg F$.*

PROOF. We assume the ℓ_2 sampling is performed perfectly since the $(1 \pm \epsilon)$ error will only introduce a factor $(1 \pm \epsilon)$ to all the probabilities. First observe that

$$\begin{aligned}
\Pr[F] &= \sum_{v \in V} \frac{\deg(u)^2}{\sum_{u \in V} \deg(u)^2} \cdot \frac{1}{\deg(v)} \\
&= \frac{\sum_{v \in V} \deg(v)}{\sum_{u \in V} \deg(u)^2} \\
&\leq \frac{2m}{4m^2/n} = \frac{n}{2m} \leq 1/4 \,.
\end{aligned}$$

Let R_v be the event that we sample node v and that $e_1 \neq e_2$.

$$\begin{aligned}
&\Pr[R_v | \neg F] \\
&= \frac{\Pr[R_v \wedge \neg F]}{\Pr[\neg F]} \\
&= \frac{\deg(v)^2}{\sum_u \deg(u)^2} \left(\frac{2}{\deg(v)^2}\right) \left(\frac{1}{1 - \sum_v \deg(v)/\sum_u \deg(u)^2}\right) \\
&= \frac{2}{\sum_u \deg(u)(\deg(u) - 1)} = 1/W \,.
\end{aligned}$$

Therefore e_1 and e_2 form a wedge chosen uniformly at random. Hence the probability that we find a triangle with edges e_1, e_2, e_3 where e_3 arrives in the stream after e_1 and e_2 are sampled in the second pass equals $\frac{3T}{W} \cdot \frac{1}{3} = T/W$ as required. Note that the fact that e_3 comes last is not true if we condition on the node v that was chosen as the "center" of the wedge; but since we chose a wedge uniformly at random, there was probability of $1/3$ that e_1 and e_2 were the first two edges of the triangle. ☐

Appealing to the previous lemma and the above discussion we can multiplicatively estimate T/W in two passes. Using the fact we can also multiplicatively approximate W in parallel (see details below), we can also multiplicatively approximate T.

THEOREM 12. *There is a two pass, $\tilde{O}(\epsilon^{-2}W/T)$-space algorithm in the arbitrary order model that returns a $(1+\epsilon)$-approximation to T with probability at least $99/100$.*

Three pass algorithm using $\tilde{O}(\epsilon^{-2}m^{3/2}/T)$ space and a degree oracle. Note that the above algorithm could be implemented in a single pass if the degrees of the nodes in the graph were known a priori. If we can assume oracle access to the degrees of the nodes in the graph (this may be reasonable in various systems), we can further improve the space use given additional passes. Given a degree oracle we can evaluate the $<_d$ ordering (as defined in Section 2.2) between the two endpoints of an edge when this edge arrives. This will allow us to focus on a smaller set of wedges when following the wedge sampling procedure above. Specifically, let W' be the number of length two-paths x-y-z where $y <_d x$ and $y <_d z$. We call such wedges "$<_d$-consistent". Note that $W' \leq W$ and it can be significantly less; W may be $\Omega(mn)$ whereas $W' = O(m^{3/2})$.

LEMMA 13. $W' < 2m^{3/2}$.

PROOF. Associate every wedge with its internal node y. The number of $z \in \Gamma(y)$ such that $y <_d z$ is at most $\sqrt{2m}$ using the same argument as used in Lemma 4. Hence, every edge participates in at most $(\sqrt{2m} - 1)$ wedges in W'. Therefore $W' < 2m^{3/2}$. ☐

Our three pass algorithm for sampling $<_d$-consistent wedges and checking if they participate in triangles is described in Figure 4. To output the estimate of the number of triangles, rather than T/W', we note that W' can be estimated via second frequency moment estimation [9] as explained shortly.

The analysis of this algorithm is analogous to that of the previous algorithm. Note that it is not clear whether it is possible to collapse

```
Algorithm TRIANGLES4
  1. Initialization: $A = 0$, $B = 0$, $r = \alpha \log n \cdot \epsilon^{-2} m^{3/2}/T$ for a large constant $\alpha$.
  2. Repeat $r$ times in parallel:
     (a) First pass: Use an $\ell_2$-sampling algorithm to sample a node $v$ with probability proportional to $(1 \pm \epsilon)|\{u \in \Gamma(v) : v <_d u\}|$.
     (b) Second pass: Given the node $v$ chosen in the first pass, sample edges $e_1$ and $e_2$ from $\{\{u, v\} : u \in \Gamma(v), v <_d u\}$ via $\ell_0$ sampling.
     (c) Third pass: If $e_1 \neq e_2$:
         i. $B \leftarrow B + 1$.
         ii. If for some edge $e$ in the stream $\{e, e_1, e_2\}$ form a triangle, $A \leftarrow A + 1$.
  3. Output: $AW'/B$
```

Figure 4: The TRIANGLES4 Algorithm. The algorithm attempts to sample r wedges that are $<_d$-consistent in parallel. B stores the number of wedges found and A stores the number of these that participate in triangles.

the second and third pass in a single pass, since the stream may be ordered such that edges on high degree nodes arrive before edges on low degree nodes.

THEOREM 14. *Given access to a degree oracle, there is a three pass, $\tilde{O}(\epsilon^{-2} m^{3/2}/T)$-space algorithm in the arbitrary order model that returns a $(1 + \epsilon)$-approximation to T with probability at least $99/100$.*

Estimating W and W'. We start by rewriting W as follows:

$$W = 1/2 \cdot \sum_{v \in V} \deg(v)(\deg(v) - 1) = 1/2 \cdot \sum_{v \in V} \deg(v)^2 - m.$$

Assuming that $m \geq 2n$, then

$$1/2 \cdot \sum_{v \in V} \deg(v)^2 \geq 1/2 \cdot 4m^2/n \geq 4m.$$

Since m can be computed exactly, it follows that it is sufficient to $(1 + 3\epsilon/4)$-approximate $\sum_{v \in V} \deg(v)^2$ if we wish to approximate $(1 + \epsilon)$-approximate W. This can be done using existing algorithms for estimating the second frequency moment estimation [9].

The case of W' can be argued similarly. Let

$$\deg'(v) = |\{u \in \Gamma(v) : u >_d v\}| .$$

Then,

$$W' = 1/2 \cdot \sum_{v \in V} \deg'(v)(\deg'(v) - 1) = 1/2 \cdot \sum_{v \in V} \deg'(v)^2 - m/2.$$

We also have

$$1/2 \cdot \sum_{v \in V} \deg'(v)^2 \geq 1/2 \cdot m^2/n \geq m .$$

Thus, $(1 + \epsilon/2)$-approximating $\sum_{v \in V} \deg'(v)^2$ (which we can do given a degree oracle) is sufficient to $(1 + \epsilon)$-approximate W'.

4.2 Running Time and Fast ℓ_p-Sampling

The above two algorithms require running many parallel copies of the corresponding wedge sampler. It may initially appear that running s copies would necessitate $\Omega(s)$ update time and thus make the algorithms prohibitively slow. Fortunately, this can be avoided and we can ensure $O(\text{polylog}\, n)$ update time. Specifically, it is easy to ensure that the third pass can be performed with $O(\log n)$ update time; we store the wedges in a hash table and when a new edge $\{x, y\}$ is read, we increment a counter by the number of wedges with endpoints x and y that are present. The more challenging problem is ensuring the s copies of an ℓ_0 and ℓ_2 sampler can be performed in parallel with update time that is independent of s. We do this in the next section.

4.2.1 Fast ℓ_p Sampling

We now introduce a fast ℓ_p-sampling technique which completes the argument that $O(\text{polylog}\, n)$ update time in the above wedge sampling algorithms is possible. Since ℓ_p-sampling is an important primitive for numerous streaming applications such as cascaded norms, duplicate detection, and higher moment estimation [36], our technique is also expected to be of independent interest. The case of fast ℓ_0-sampling is significantly simpler and a result was previously known [33].

Preliminaries and Intuition. Most streaming problems can be modeled by the evolution of an underlying n-dimensional integer vector x. In the *turnstile model*, the stream consists of $\text{poly}(n)$ updates in the form $x_i \leftarrow x_i + \delta$. In the context of the above algorithms, x_i would correspond to the degree of node i or the number of neighbors of i that have higher degree or a boolean to indicate that node i is neighbor of node v. An ℓ_p-sampler (see, [3, 15, 25, 36]) takes one pass over the stream and with high probability returns $(j, \tilde{x}_j)$ where $\tilde{x}_j = (1 \pm \epsilon)x_j$ and

$$\Pr[j = i] = (1 \pm \epsilon)x_i^p / F_p(x)$$

where $F_p(x) = \sum_i x_i^p$ and $F_0 = |\{i : x_i \neq 0\}|$.

The space and update time of existing ℓ_p-sampling algorithms are $O(\text{poly}(\epsilon^{-1}, \log n))$ and $O(\text{polylog}\, n)$ respectively per sample. If we want to draw s independent ℓ_p-samples, the naive implementation that maintains s different ℓ_p-samplers in parallel would need $\Omega(s)$ update time. We will prove the following theorem.

THEOREM 15. *For $p \in [0, 2]$, there exists a one-pass algorithm that, with high probability, outputs s independent ℓ_p-samples using $O(s \cdot \text{poly}(\epsilon^{-1}, \log n))$ space and $O(\text{polylog}\, n)$ update time.*

We achieve $\text{polylog}\, n$ update time regardless of s and our result is most significant when $s = o(n)$ and $s = \omega(\text{polylog}\, n)$. The main idea behind our algorithm is as follows. We hash the coordinates of x into w groups and for each of these groups we maintain a small number of local ℓ_p-samplers restricting to the corresponding coordinates. To draw an ℓ_p-sample, we randomly pick a group with probability proportional to its mass contribution to $F_p(x)$ and pick an ℓ_p-sample from that group using a local ℓ_p-sampler. The main challenge is ensuring that each group's contribution is small so that we only need to maintain a small number of samplers in each group, hence, the fast update time. At a very high level, we achieve this by separating the heavy coordinates into one group using Heavy-Hitters algorithm.

Detailed Description. We restrict our attention to the case $p \in (0, 2]$. We make use of the following Heavy-Hitters result (see [25], Lemma 1 and Section 4.4) .

LEMMA 16. *For $p \in (0, 2]$, there exists a one-pass, $O(\epsilon^{-p}\phi^{-1} \cdot \operatorname{polylog} n)$-space and $O(\log n)$-update time algorithm that with high probability returns $A \subseteq [n]$ and $B = \{\tilde{x}_i : \tilde{x}_i = (1 \pm \epsilon)x_i$ and $i \in A\}$ such that if $x_i^p \geq \phi F_p(x)$ then $i \in A$ and if $x_i^p < (\phi/8)F_p(x)$ then $i \notin A$.*

We next introduce some definitions. We consider a set of pairwise independent functions $\{h_i\}_{i\in[d]} : [n] \to [w]$ where $d = c\log n$ and $w = cs$ for some sufficiently large constant c. We let

$$A_{i,j} = \{k \in [n] : h_i(k) = j\} .$$

We use $a^{(i,j)}$ to denote the characteristic vector of the set $A_{i,j}$. Finally, we define the set of heavy coordinates as

$$H = \{j \in [n] : x_j^p \geq F_p(x)/s\}$$

and let H' be any superset of H. We consider the algorithm in Figure 5.

By running the Heavy-Hitters algorithm with $\phi = 1/s$, at the end of the stream, we have a set $H' \supseteq H$ as required. Moreover, for each $k \in H'$, the algorithm also returns $\tilde{x}_k^p = (1 \pm O(\epsilon))x_k^p$. One can use frequency moment approximation algorithm such as [21] that supports $O(\operatorname{polylog} n)$ update time to maintain r and $\alpha^{(i,j)}$.

The update time during the stream is $O(\operatorname{polylog} n)$. To see this, a stream update to coordinate k involves the following steps. First, the algorithm needs to compute $h_i(k)$ for each $i \in [d] = [c\log n]$. Then, it updates $O(\log n)$ data structures that are used to maintain $\{\alpha^{(i,j)}\}_{h_i(k)=j}$, r, and the ℓ_p-samplers on the vectors $\{a^{(i,j)}\}_{h_i(k)=j}$. Finally, the algorithm updates the Heavy-Hitters subroutine. Each of these updates can be done in $O(\operatorname{polylog} n)$ time.

Based on the algorithm, we say group $A_{i,j}$ is *good* if $(i, j) \in G$. Let $I_{k\in H'}$ be the indicator variable for the event $k \in H'$. We define

$$g(k) = |\{(i, j) \in G : k \in A_{i,j}\}| + I_{k\in H'} .$$

Intuitively, we use $g(k)$ in the rejection probability to avoid bias toward the coordinates that appear in many groups. It is important to note that the rejection probability is valid. Since each coordinate is in at most d good groups, for sufficiently small ϵ,

$$\alpha \leq (1 + \epsilon) \sum_{(i,j)\in G} F_p(a^{(ij)}) \leq (1 + \epsilon)d \cdot F_p(x) \leq 2d \cdot r .$$

Furthermore, it is obvious that

$$\beta \leq (1 + \epsilon) \sum_{k\in H'} x_k^p \leq (1 + \epsilon)F_p(x) \leq 2d \cdot r .$$

LEMMA 17. *For all $k \notin H'$, k belongs to at least one good group with high probability.*

PROOF. Observe that if $k \notin H'$, then $x_k^p \leq F_p(x)/s$. Fix i and suppose $h_i(k) = j$. By pairwise independence,

$$\begin{aligned} \mathrm{E}\left[F_p(a^{(i,j)})\right] &\leq x_k^p + \sum_{z\neq k} x_z^p/w \\ &\leq x_k^p + F_p(x)/(c \cdot s) \leq 2F_p(x)/s. \end{aligned}$$

Applying Markov bound, $\Pr\left[F_p(a^{(i,j)}) > 8F_p(x)/s\right] \leq 1/4$. Hence,

$$\begin{aligned} \Pr[A_{i,j} \text{ is good}] &\geq \Pr\left[(1+\epsilon)F_p(a^{(i,j)}) < 10(1-\epsilon)F_p(x)/s\right] \\ &\geq \Pr\left[F_p(a^{(i,j)}) < 8F_p(x)/s\right] \geq 3/4. \end{aligned}$$

for sufficiently large c and small ϵ. Thus, the probability that there is no good group for k is at most $(1/4)^d = (1/4)^{c\log n} \leq n^{-c}$. The lemma follows by taking the union bound over all $k \in [n]$. □

Let $S_p(x_k)$ be the event of outputting $(k, \tilde{x}_k)$ as a sample and let $S_p(\text{success}) = \cup_{k\in[n]} S_p(x_k)$. The probability of successfully retrieving a sample is

$$\begin{aligned} \Pr[S_p(\text{success})] &= \sum_{k\in H'} \Pr[S_p(x_k) \mid \text{HEAD}] \Pr[\text{HEAD}] + \\ &\quad \sum_{\substack{(i,j)\in G \\ k\in A_{i,j}}} \Pr[S_p(x_k) \mid \text{TAIL}] \Pr[\text{TAIL}] \\ &= \sum_{k\in H'} \frac{1}{2} \cdot \frac{(1\pm\epsilon)x_k^p}{\beta} \cdot \frac{\beta}{2dr \cdot g(k)} \\ &\quad + \sum_{\substack{(i,j)\in G \\ k\in A_{i,j}}} \frac{1}{2} \cdot \frac{\alpha^{(i,j)}}{\alpha} \cdot \frac{\alpha}{2dr} \cdot \frac{(1\pm\epsilon)x_k^p}{F_p(a^{(i,j)})} \cdot \frac{1}{g(k)} \\ &= \sum_{k\in[n]} \frac{(1\pm O(\epsilon))x_k^p}{4cF_p(x)\log n} = \Theta(\frac{1}{\log n}). \end{aligned}$$

The third step follows from the fact that $r = (1 \pm \epsilon)F_p(x)$ and $\alpha^{(i,j)} = (1 \pm \epsilon)F_p(a^{(i,j)})$. It is straight forward to verify that

$$\Pr[S_p(x_k) \mid S(\text{success})] = (1 \pm O(\epsilon))x_k^p/F_p(x)$$

as required.

Finally, we show that it suffices to maintain $O(\log n)$ ℓ_p-samplers on each vector $a^{(i,j)}$.

LEMMA 18. *With high probability, repeating $O(s\log n)$ times procedure S_p, we obtain s independent ℓ_p-samples. Furthermore, we need $O(\log n)$ different ℓ_p-samples from each good group.*

PROOF. As shown above, $\Pr[S_p(\text{success})] = \Omega(1/\log n)$. The first claim follows from Chernoff bound. On the other hand, $\alpha^{(i,j)} \leq 10r/s$ for all $(i, j) \in G$. Therefore, the probability that we require an ℓ_p-sample from a good group $A_{i,j}$ is

$$\alpha^{(i,j)}/(4dr) = O(1/(s\log n)) .$$

Therefore, for appropriate choice of constants, the probability that a good group needs more than $O(\log n)$ ℓ_p-samples is less than $1/\operatorname{poly}(n)$. Finally, appealing to the union bound over all $O(s\log n)$ good groups concludes the second claim. □

5. CONCLUSIONS

We gave four main algorithms for estimating the number of triangles in the arbitrary order and adjacency list data stream models.

```
Algorithm FAST $\ell_p$ SAMPLING
  1. During the stream,
     (a) Maintain $r = (1 \pm \epsilon)F_p(x)$ and $\alpha^{(i,j)} = (1 \pm \epsilon)F_p(a^{(i,j)})$ for each vector $a^{(i,j)}$.
     (b) Maintain $O(\log n)$ $\ell_p$-samplers on each vector $a^{(i,j)}$.
     (c) Run the Heavy-Hitters algorithm with $\phi = 1/s$ on the vector $x$.
  2. In post-processing:
     (a) $H' \leftarrow$ the indices returned by the Heavy-Hitters algorithm and compute $\beta := \sum_{k \in H'} \tilde{x}_k^p$.
     (b) Compute the set $G := \{(i,j) : \alpha^{(i,j)} < 10r/s\}$ and compute $\alpha := \sum_{(i,j) \in G} \alpha^{(i,j)}$.
     (c) Repeat the following procedure $S_p$ until $s$ samples are retrieved:
         i. Toss a fair coin.
         ii. If HEAD:
             A. Randomly pick $k \in H'$ where $k = k'$ with probability $\tilde{x}_{k'}^p/\beta$.
             B. Reject with probability $1 - \beta/(2d \cdot r \cdot g(k))$. Otherwise, output $(k, \tilde{x}_k)$ as a sample.
         iii. If TAIL:
             A. Randomly pick $(i,j) \in G$ where $(i,j) = (i',j')$ with probability $\alpha^{(i'j')}/\alpha$.
             B. Reject the current procedure with probability $1 - \alpha/(2 \cdot d \cdot r)$.
             C. Otherwise, use the next $\ell_p$-sampler on $a^{(i,j)}$ to get $(k, \tilde{x}_k)$.
             D. Reject with probability $1 - 1/g(k)$. Otherwise, output $(k, \tilde{x}_k)$ as a sample.
```

Figure 5: The FAST ℓ_p-SAMPLING ALGORITHM

These algorithms use less space than the existing state-of-the-art algorithms for either all parameter settings (the number of nodes, edges, triangles etc.) or a large range of parameters depending on whether one or two passes over stream are permitted.

6. REFERENCES

[1] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 5–14, 2012.

[2] N. Alon, R. Yuster, and U. Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.

[3] A. Andoni, R. Krauthgamer, and K. Onak. Streaming algorithms via precision sampling. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 363–372. IEEE, 2011.

[4] L. Arge, M. T. Goodrich, and N. Sitchinava. Parallel external memory graph algorithms. In *24th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2010, Atlanta, Georgia, USA, 19-23 April 2010 - Conference Proceedings*, pages 1–11, 2010.

[5] S. Arifuzzaman, M. Khan, and M. V. Marathe. PATRIC: a parallel algorithm for counting triangles in massive networks. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 529–538, 2013.

[6] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA.*, pages 623–632, 2002.

[7] J. W. Berry, B. Hendrickson, S. Kahan, and P. Konecny. Software and algorithms for graph queries on multithreaded architectures. In *21th International Parallel and Distributed Processing Symposium (IPDPS 2007), Proceedings, 26-30 March 2007, Long Beach, California, USA*, pages 1–14, 2007.

[8] V. Braverman, R. Ostrovsky, and D. Vilenchik. How hard is counting triangles in the streaming model? In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 244–254, 2013.

[9] L. Bulteau, V. Froese, K. Kutzkov, and R. Pagh. Triangle counting in dynamic graph streams. *CoRR*, abs/1404.4696, 2014.

[10] L. S. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, and C. Sohler. Counting triangles in data streams. In *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*, pages 253–262, 2006.

[11] N. Chiba and T. Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. Comput.*, 14(1):210–223, 1985.

[12] K. L. Clarkson and D. P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 205–214, 2009.

[13] G. Cormode. Personal communication 8/23/15. 2015.

[14] G. Cormode. Personal communication 1/23/16. 2016.

[15] G. Cormode and D. Firmani. A unifying framework for ℓ_0-sampling algorithms. *Distributed and Parallel Databases*, 32(3):315–335, 2014.

[16] G. Cormode and H. Jowhari. A second look at counting triangles in graph streams. *Theor. Comput. Sci.*, 552:44–51, 2014.

[17] J.-P. Eckmann and E. Moses. Curvature of co-links uncovers hidden thematic layers in the world wide web. *Proceedings of the National Academy of Sciences*, 99(9):5825–5829, 2002.

[18] T. Eden, A. Levi, D. Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. In *FOCS*, 2015.

[19] M. Ghashami, E. Liberty, J. M. Phillips, and D. P. Woodruff. Frequent directions : Simple and deterministic matrix sketching. *CoRR*, abs/1501.01711, 2015.

[20] A. Goel, M. Kapralov, and S. Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 468–485, 2012.

[21] P. Indyk and D. Woodruff. Optimal approximations of the frequency moments of data streams. In *thirty-seventh annual ACM symposium on Theory of computing*, pages 202–208. ACM New York, NY, USA, 2005.

[22] M. Jha, C. Seshadhri, and A. Pinar. A space-efficient streaming algorithm for estimating transitivity and triangle counts using the birthday paradox. *TKDD*, 9(3):15:1–15:21, 2015.

[23] A. G. Jørgensen and S. Pettie. Threesomes, degenerates, and love triangles. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 621–630, 2014.

[24] 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 16-29, 2005, Proceedings*, pages 710–716, 2005.

[25] H. Jowhari, M. Saglam, and G. Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *PODS*, pages 49–58, 2011.

[26] M. Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1679–1697, 2013.

[27] T. G. Kolda, A. Pinar, and C. Seshadhri. Triadic measures on graphs: The power of wedge sampling. In *Proceedings of the 13th SIAM International Conference on Data Mining, May 2-4, 2013. Austin, Texas, USA.*, pages 10–18, 2013.

[28] M. N. Kolountzakis, G. L. Miller, R. Peng, and C. E. Tsourakakis. Efficient triangle counting in large graphs via degree-based vertex partitioning. *Internet Mathematics*, 8(1-2):161–185, 2012.

[29] K. Kutzkov and R. Pagh. Triangle counting in dynamic graph streams. In *Algorithm Theory - SWAT 2014 - 14th Scandinavian Symposium and Workshops, Copenhagen, Denmark, July 2-4, 2014. Proceedings*, pages 306–318, 2014.

[30] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins. Microscopic evolution of social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 462–470, 2008.

[31] M. Manjunath, K. Mehlhorn, K. Panagiotou, and H. Sun. Approximate counting of cycles in streams. In *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, pages 677–688, 2011.

[32] A. McGregor. Graph stream algorithms: a survey. *SIGMOD Record*, 43(1):9–20, 2014.

[33] A. McGregor, D. Tench, S. Vorotnikova, and H. T. Vu. Densest subgraph in dynamic graph streams. In *Mathematical Foundations of Computer Science 2015*, pages 472–482. Springer Berlin Heidelberg, 2015.

[34] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. In *Science*, pages 824–827, 2002.

[35] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[36] M. Monemizadeh and D. P. Woodruff. 1-pass relative-error ℓ_p-sampling with applications. In *SODA*, pages 1143–1160, 2010.

[37] R. Pagh and C. E. Tsourakakis. Colorful triangle counting and a mapreduce implementation. *Inf. Process. Lett.*, 112(7):277–281, 2012.

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

[39] T. Schank and D. Wagner. Approximating clustering coefficient and transitivity. *J. Graph Algorithms Appl.*, 9(2):265–275, 2005.

[40] S. Suri and S. Vassilvitskii. Counting triangles and the curse of the last reducer. In *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011*, pages 607–614, 2011.

[41] K. Tangwongsan, A. Pavan, and S. Tirthapura. Parallel triangle counting in massive streaming graphs. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 781–786, 2013.

[42] C. E. Tsourakakis, M. N. Kolountzakis, and G. L. Miller. Triangle sparsifiers. *J. Graph Algorithms Appl.*, 15(6):703–726, 2011.

[43] J. S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, 1985.

[44] B. F. Welles, A. V. Devender, and N. S. Contractor. Is a friend a friend?: investigating the structure of friendship networks in virtual worlds. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010, Extended Abstracts Volume, Atlanta, Georgia, USA, April 10-15, 2010*, pages 4027–4032, 2010.