**Noname manuscript No.**
(will be inserted by the editor)

# Temporal Locality-Aware Sampling for Accurate Triangle Counting in Real Graph Streams

**Dongjin Lee · Kijung Shin · Christos Faloutsos**

Received: date / Accepted: date

**Abstract** If we cannot store all edges in a dynamic graph, which edges should we store to estimate the triangle count accurately?

Counting triangles (i.e., cliques of size three) is a fundamental graph problem with many applications in social network analysis, web mining, anomaly detection, etc. Recently, much effort has been made to accurately estimate the counts of global triangles (i.e., all triangles) and local triangles (i.e., all triangle incident to each node) in large dynamic graphs, especially with limited space. Although existing algorithms use sampling techniques without considering temporal dependencies in edges, we observe *temporal locality* in the formation of triangles in real dynamic graphs. That is, future edges are more likely to form triangles with recent edges than with older edges.

In this work, we propose a family of single-pass streaming algorithms called *Waiting-Room Sampling* (WRS) for estimating the counts of global and local triangles in a fully dynamic graph, where edges are inserted and deleted over time, within a fixed memory budget. WRS exploits the temporal locality by always storing the most recent edges, which future edges are more likely to form triangles with, in the *waiting room*, while it uses reservoir sampling and its variant for the remaining edges. Our theoretical and empirical analyses show that WRS is: **(a) Fast and 'any time':** runs in linear time, always maintaining and updating estimates while the input graph evolves, **(b) Effective**: yields up to *47% smaller estimation error* than its best competitors, and **(c) Theoretically sound**: gives unbiased estimates with small variances under the temporal locality.

**Keywords** Triangle Counting · Graph Stream · Waiting-Room Sampling · Temporal Locality

D. Lee
School of Electrical Engineering, KAIST, Daejeon, South Korea, 34141.
E-mail: dongjin.lee@kaist.ac.kr

K. Shin (Corresponding Author)
Graduate School of AI & School of Electrical Engineering, KAIST, Daejeon, South Korea, 34141.
E-mail: kijungs@kaist.ac.kr

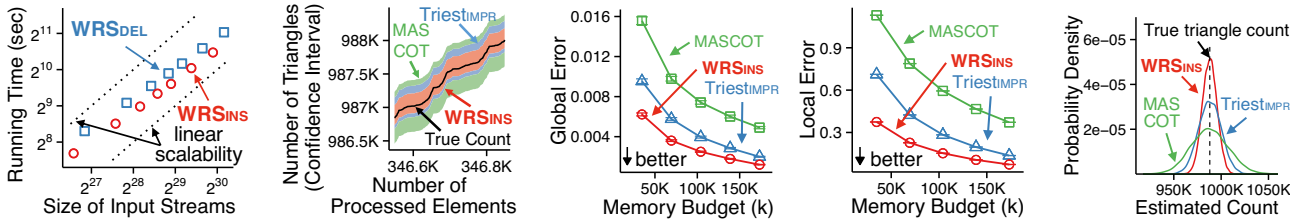
C. Faloutsos
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 15213.
E-mail: christos@cs.cmu.edu

## 1 Introduction

Consider a large dynamic graph where edges are inserted and deleted over time. If we cannot store every edge in memory, which edges should we store to estimate the count of triangles accurately?

Counting the triangles (i.e., cliques of size three) in a graph is a fundamental problem with many applications. For example, triangles in social networks have received much attention as an evidence of homophily (i.e., people choose friends similar to themselves) [28] and transitivity (i.e., people with common friends become friends) [51]. Thus, many concepts in social network analysis, such as social balance [51], the global/local clustering coefficient [52], and the transitivity ratio [30], are based on the triangle count. Moreover, the count of triangles has been used for spam and anomaly detection [6, 27, 40], web-structure analysis [9], degeneracy estimation [36], and query optimization [4].

Due to the importance of triangle counting, numerous algorithms have been developed in many different settings, including multi-core [41, 20], external-memory [20, 16], distributed-memory [3], and MapReduce [43, 32, 31] settings. The algorithms aim to accurately and




(a) WRS is fast (b) WRS is 'any time' (c) WRS is effective (d) WRS is unbiased

Fig. 1: Strengths of WRS. (a) WRS scales linearly with the size of the input stream. (b) WRS always maintains the estimates of the triangle counts while the input graph evolves. (c) WRS is more accurate than state-of-the-art streaming algorithms in both global and local triangle counting. (d) WRS gives unbiased estimates (Theorems 1 and 2) with small variances (Lemma 5).

rapidly count *global triangles* (i.e., all triangles in a graph) and/or *local triangles* (i.e., triangles that each node is involved with).

Especially, as many real graphs, including social media and web, evolve over time, recent work has focused largely on streaming settings where graphs are given as a sequence of edge insertions and deletions. To accurately estimate the count of the triangles in large graph streams not fitting in memory, a number of sampling-based algorithms have been developed, as summarized in Table 1.

Notably, many real-world graphs exhibit *temporal locality* in triangle formation, i.e., the tendency that future edges are more likely to form triangles with recent edges than with older edges (see Section 4 for details). However, existing streaming algorithms sample edges without considering temporal dependencies between edges, and thus they cannot exploit the temporal locality in triangle formation.

How can we exploit temporal locality for accurately estimating global and local triangle counts? As an answer to this question, we propose *Waiting-Room Sampling* (WRS)[1], a family of single-pass streaming algorithms for estimating the counts of global and local triangles in a fully dynamic graph stream, within a fixed memory budget. WRS always stores the most recent edges in the *waiting room*, while using standard reservoir sampling [49] and its variant, namely random pairing [13], for the remaining edges. Due to the temporal locality, when a new edge arrives, recent edges, which the new edge is likely to from triangles with, are always stored in the waiting room. Thus, the waiting room enables WRS to discover more triangles, which are useful for estimating the triangle counts more accurately, while reservoir sampling and random pairing enable WRS to give unbiased estimates.

[1] A preliminary version of WRS for insertion-only graph streams was presented in [35]. This work is an extended version of [35] with (a) a new algorithm for fully dynamic graph streams, (b) theoretical analyses on its accuracy and complexity, and (c) additional experiments with more datasets, competitors, and evaluation metrics.

Table 1: Comparison of state-of-the-art streaming algorithms for triangle counting. By exploiting temporal patterns, WRS achieves the best accuracy. WRS also satisfies all other considered criteria.

| | **WRS** (Proposed) | ThinkD [38,40] | TrièstFD [8] | ESD [15] | TrièstIMPR [8] | Mascot [27] | Others [1,2,33] |
|---|---|---|---|---|---|---|---|
| Count Global Triangles | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Count Local Triangles | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ |
| Handling Large Graphs* | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ |
| Handle Edge Additions | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Handle Edge Deletions | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ |
| Exploit Temporal Patterns | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |

*Graphs that are too large to fit in memory

Our theoretical and empirical analyses show that WRS has the following strengths:

- **Fast and 'any time'**: WRS scales linearly with the size of the input stream, and it gives the estimates of triangle counts at any time, not only at the end of the input stream (Figures 1(a)-(b)).
- **Effective**: WRS produces up to *47% smaller* estimation error than its best competitors (Figure 1(c)).
- **Theoretically sound**: we prove the unbiasedness of the estimates given by WRS and their small variances under temporal locality (Theorems 1 and 2; Figure 1(d)).

**Reproducibility**: The code and datasets used in the paper are available at `http://dmlab.kaist.ac.kr/wrs/`.

This paper is organized as follows. In Section 2, we review related work. In Section 3, we introduce notations and the problem definition. In Section 4, we discuss temporal locality in real graph streams. In Section 5, we propose our algorithm WRS and analyze its theoretical properties. After sharing experimental results in Section 6, we conclude in Section 7.

## 2 Related Work

We discuss previous work on counting global and local triangles in a graph. After we briefly discuss counting triangles in a static graph, we focus on counting triangles with limited space in a graph stream, where edges arrive as a data stream. See Table 1 for a summary of the state-of-the-art streaming algorithms.

**Triangle counting in static graphs:** There has been considerable interest in counting triangles in a static graph, and there have been numerous algorithms working in multi-core [41,20], external-memory [20,16], distributed-memory [3], and MapReduce [43,32,31] settings. While most of them are exact, several of them are approximate based on eigenvalues [44], sampled edges [3,11,19,41], and sampled wedges (i.e., paths of length 2) [34,47,46]. Notably a state-of-the-art method [46] estimates the global triangle count by sampling wedges after eliminating those that are less likely to participate in triangles. These approaches for static graphs are not directly applicable to graph streams, where edges should be processed in the order that they arrive.

**Global triangle counting in insertion-only graph streams:** For estimating the count of global triangles (i.e., all triangles in a graph), Tsourakakis et al. [45] proposed sampling each edge i.i.d. with probability $p$. Then, the global triangle count is estimated simply by multiplying that in the sampled graph by $p^{-3}$. Jha et al. [17] and Pavan et al. [33] proposed sampling wedges (i.e., paths of length 2) instead of edges for better space efficiency. Ahmed et al. [1] proposed an edge-sampling method where edges are sampled with different probabilities depending on whether an adjacency (i.e. an edge sharing a node) of them has been sampled or not. Ahmed et al. [2] proposed a priority-based edge sampling method where the priority is proportional to the number of the adjacencies of each edge or the number of triangles created by the edge.

**Global triangle counting in fully-dynamic graph streams:** Kutzkov and Pagh [25] combined edge and wedge sampling methods for global triangle counting in fully-dynamic graph streams. For the same problem, Han and Sethu [15] proposed an incremental algorithm, which, however, requires the entire graph to be maintained in memory.

**Local triangle counting in insertion-only graph streams:** For estimating the count of local triangles (i.e., triangles with each node), Lim and Kang [27] proposed MASCOT which samples each edge i.i.d with a fixed probability $p$ but updating global and local counts whenever an edge arrives, even when the edge is not sampled. To properly set $p$, however, the number of edges in input streams should be known in advance. Likewise, randomly coloring nodes to sample the edges connecting nodes of the same color, as suggested by Kutzkov and Pagh [24], requires the number of nodes in advance to decide the number of colors. De Stefani et al. [8] proposed TRIÈST$_{\text{IMPR}}$ to address this problem using reservoir sampling [49], which fully utilizes given memory space, without requiring any prior knowledge of the input stream. Recently, Jung et al. [18] and Wang et al. [50] proposed streaming algorithms for counting global and local triangles, respectively, in a multigraph stream, which may have duplicated edges. Shin et al. [37,39] proposed distributed algorithms for triangle counting in a graph stream where edges are streamed from multiple sources. The streamed edges are processed and sampled across multiple workers.

**Local triangle counting in fully-dynamic graph streams:** De Stefani et al. [8] proposed TRIÈST$_{\text{FD}}$ for global and local triangle counting in fully-dynamic graph streams. TRIÈST$_{\text{FD}}$ samples edges using random pairing [13], a variant of reservoir sampling for fully-dynamic streams, without requiring any prior knowledge of the input stream. While TRIÈST$_{\text{FD}}$ simply discards unsampled edges, THINKD, proposed by Shin et al. [38,40] for the same problem, fully utilizes unsampled edges to update estimates before discarding them. Specifically, in response to each arrived change in the input stream, THINKD first updates corresponding estimates, which are shown to be unbiased, and then update the sampled graph (i.e., sampled edges). When updating the sampled graph, THINKD$_{\text{FAST}}$, which is a simple and fast version, samples each edge i.i.d. with a fixed probability, while THINKD$_{\text{ACC}}$, which is an accurate version, employs random pairing [13].

**Semi-streaming algorithms:** In addition to single-pass streaming algorithms, semi-streaming algorithms that require multiple passes over a graph were also explored [6,23].

**Comparison with our proposed algorithm:** Our single-pass algorithm WRS estimates both global and local triangle counts in a fully dynamic graph stream without any prior knowledge of the input graph stream (see Section 3.3 for the detailed settings). Different from the existing approaches above, WRS exploits the temporal locality in real-world graph streams (see Section 4), which leads to higher accuracy.

## 3 Preliminaries and Problem Definition

In this section, we first introduce notations and concepts used in the paper. Then, we review two uniform sampling schemes that our proposed algorithms are based on. Lastly, we define the problem of global and local triangle counting in a real graph stream.

Table 2: Table of symbols.

| Symbol | Definition |
|---|---|
| Notations for Graph Streams | |
| $\mathcal{G}^{(t)} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)})$ | graph $\mathcal{G}$ at time $t$ |
| $\{u, v\}$ | edge between nodes $u$ and $v$ |
| $\{u, v, w\}$ | triangle with nodes $u$, $v$, and $w$ |
| $\Delta^{(t)} = (e^{(t)}, \delta^{(t)})$ | edge insertion or deletion at time $t$ |
| $e^{(t)} = \{u, v\}$ | edge changed at time $t$ |
| $t_{uv}$ | arrival time of edge $\{u, v\}$ |
| $e^{(i)}_{uvw}$ | edge arrived $i$-th among $\{\{u, v\}, \{v, w\}, \{w, u\}\}$ |
| $t^{(i)}_{uvw}$ | arrival time of $e^{(i)}_{uvw}$ |
| $\mathcal{T}^{(t)}$ | set of triangles in $\mathcal{G}^{(t)}$ |
| $\mathcal{T}^{(t)}_u$ | set of triangles with a node $u$ in $\mathcal{G}^{(t)}$ |
| Notations for Algorithms (defined in Section 5) | |
| $\mathcal{S}$ | given memory space |
| $\mathcal{W}$ | waiting room |
| $\mathcal{R}$ | reservoir |
| $\mathcal{E}_{\mathcal{R}}$ | set of edges flowing into $\mathcal{R}$ from $\mathcal{W}$ |
| $k$ | maximum number of edges stored in $\mathcal{S}$ |
| $\alpha$ | relative size of the waiting room (i.e., $\lvert\mathcal{W}\rvert/\lvert\mathcal{S}\rvert$) |
| $c$ | estimated global triangle count |
| $c_u$ | estimated local triangle count of node $u$ |
| $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$ | graph composed of the edges in $\mathcal{S}$ |
| $\hat{\mathcal{N}}_u$ | set of neighbors of a node $u$ in $\hat{\mathcal{G}}$ |
| Notations for Analyses (defined in Section 5) | |
| $\mathcal{A}^{(t)}$ | set of triangles added at time $t$ or earlier |
| $\mathcal{D}^{(t)}$ | set of triangles deleted at time $t$ or earlier |
| $\{u, v, w\}^{(t)}$ | triangle $\{u, v, w\}$ added or deleted at time $t$ |

### 3.1 Notations and Concepts

Symbols frequently used in the paper are listed in Table 2. Consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with the set of nodes $\mathcal{V}$ and the set of edges $\mathcal{E}$. We use the unordered pair $\{u, v\} \in \mathcal{E}$ to indicate the edge between two distinct nodes $u \neq v \in \mathcal{V}$, and the unordered triple $\{u, v, w\}$ to represent the triangle (i.e., clique of size three) with nodes $u$, $v$, and $w$.

Consider a dynamic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ that evolves over time from an empty graph as edges (and adjacent nodes) are inserted and deleted. The changes in $\mathcal{G}$ can be represented as a *fully-dynamic graph stream* $(\Delta^{(1)}, \Delta^{(2)}, \dots)$, which is the sequence of edge insertions and deletions. For each $t \in \{1, 2, ...\}$, we use $e^{(t)}$ to denote the edge added or deleted at time $t$ and use the pair $\Delta^{(t)} = (e^{(t)}, \delta^{(t)})$, where $\delta^{(t)} \in \{+, -\}$, to denote the change in $\mathcal{G}$ at time $t$. That is, $\Delta^{(t)} = (\{u, v\}, +)$ indicates the arrival of a new edge $\{u, v\} \notin \mathcal{E}$, and $\Delta^{(t)} = (\{u, v\}, -)$ indicates the removal of an existing edge $\{u, v\} \in \mathcal{E}$ at time $t$. We denote $\mathcal{G}$ after the $t$-th change $\Delta^{(t)}$ is applied by $\mathcal{G}^{(t)} = (\mathcal{V}^{(t)}, \mathcal{E}^{(t)})$. We denote the set of triangles in $\mathcal{G}^{(t)}$ by $\mathcal{T}^{(t)}$ and the set of triangles with a node $u$ by $\mathcal{T}^{(t)}_u \subset \mathcal{T}^{(t)}$. We call $\mathcal{T}^{(t)}$ *global triangles* and $\mathcal{T}^{(t)}_u$ *local triangles* of each node $u$.

We use $t_{uv} \in \{1, 2, ...\}$ to indicate the last arrival time of each edge $\{u, v\}$. For example, in cases where the edge $\{u, v\}$ is deleted and reinserted in a fully dynamic graph stream (i.e., $\Delta^{(t_1)} = (\{u, v\}, +)$, $\Delta^{(t_2)} = (\{u, v\}, -)$, and $\Delta^{(t_3)} = (\{u, v\}, +)$, where $t_1 < t_2 < t_3$), $t_{uv}$ is the arrival time of the reinserted edge (i.e., $t_{uv} = t_3$). For each triangle $\{u, v, w\} \in \mathcal{T}^{(t)}$, we let $e^{(1)}_{uvw}$, $e^{(2)}_{uvw}$, and $e^{(3)}_{uvw}$ be the edge arriving first, second, and last among $\{u, v\}$, $\{v, w\}$, and $\{w, u\}$, which together form $\{u, v, w\}$. We use $t^{(i)}_{uvw}$ to indicate the arrival time of $e^{(i)}_{uvw}$. That is, $t^{(1)}_{uvw} := \min\{t_{uv}, t_{vw}, t_{wu}\}$, $t^{(2)}_{uvw} := \text{median}\{t_{uv}, t_{vw}, t_{wu}\}$, and $t^{(3)}_{uvw} := \max\{t_{uv}, t_{vw}, t_{wu}\}$. These notations are used to account for the concept of temporal locality in Section 4.

### 3.2 Uniform Sampling Schemes for Dynamic Datasets

In this subsection, we give an overview of two uniform sampling schemes, namely reservoir sampling and random pairing, for sampling as many items as possible from a dynamic dataset. Assume that a dataset $\mathcal{D}$ is initially empty, and it evolves over time as items are inserted and deleted. A sample $\mathcal{S} \subseteq \mathcal{D}$ is a set of items selected from $\mathcal{D}$ by a specific sampling rule. Following [7], a sampling scheme is *uniform* if all equal-sized subsets of $\mathcal{D}$ are equally likely to be $\mathcal{S}$, i.e., if

$$\mathbb{P}[\mathcal{S} = \mathcal{A}] = \mathbb{P}[\mathcal{S} = \mathcal{B}],\ \forall \mathcal{A} \neq \mathcal{B} \subseteq \mathcal{D}\ s.t.\ |\mathcal{A}| = |\mathcal{B}|. \quad (1)$$

The two sampling schemes described below satisfy this *uniformity*.

#### *3.2.1 Reservoir Sampling*

Given a sequence of item insertions into the dataset $\mathcal{D}$, reservoir sampling [49], which is described in Algorithm 1, maintains a bounded-size uniform sample $\mathcal{S}$. We use $k$ to denote the maximum size of the maintained sample $\mathcal{S}$. Until the sample size $|\mathcal{S}|$ reaches $k$, each item inserted into $\mathcal{D}$ is stored in $\mathcal{S}$. Then, for each item inserted into $\mathcal{D}$, reservoir sampling replaces the newly inserted item with a randomly selected item from $\mathcal{S}$ with probability $k/|\mathcal{D}|$. Reservoir sampling guarantees that each item in $\mathcal{D}$ has the same probability $k/|\mathcal{D}|$ of being stored in $\mathcal{S}$, and thus it guarantees uniformity (i.e., Eq. (1)). However, it cannot handle item deletions, and thus it can be used only for insertion-only datasets.

#### *3.2.2 Random Pairing (RP)*

Random Pairing (RP) [13] is a uniform sampling method for fully dynamic datasets, where items are both added and deleted. The main idea behind RP is to make use of newly inserted items to "compensate" for previous item deletions.

**Algorithm 1:** Reservoir Sampling

**Input :** (1) Insertion-only dataset: $\mathcal{D}$,
(2) Memory budget: $k$,
(3) Current sample: $\mathcal{S}$,
(4) Item addition: $a$
**Output:** Updated sample: $\mathcal{S}$

$\mathcal{S} \leftarrow \emptyset$, $|\mathcal{D}| \leftarrow 0$
**Procedure** INSERT($a$)
  $|\mathcal{D}| \leftarrow |\mathcal{D}| + 1$
  **if** $|\mathcal{S}| < k$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{a\}$
  **else if** $Bernoulli(k/|\mathcal{D}|) = 1$ **then**
    replace a randomly chosen item in $\mathcal{S}$ with $a$

**Algorithm 2:** Random Pairing (RP)

**Input :** (1) Fully-dynamic dataset: $\mathcal{D}$,
(2) Memory budget: $k$,
(3) Current sample: $\mathcal{S}$,
(4) Item addition or deletion: $a$
**Output:** Updated sample: $\mathcal{S}$

$\mathcal{S} \leftarrow \emptyset$, $|\mathcal{D}| \leftarrow 0$, $n_b \leftarrow 0$, $n_g \leftarrow 0$
**Procedure** INSERT($a$)
  $|\mathcal{D}| \leftarrow |\mathcal{D}| + 1$
  **if** $n_b + n_g = 0$ **then**
    **if** $|\mathcal{S}| < k$ **then** $\mathcal{S} \leftarrow \mathcal{S} \cup \{a\}$
    **else if** $Bernoulli(k/|\mathcal{D}|) = 1$ **then**
      replace a randomly chosen item in $\mathcal{S}$ with $a$
  **else**
    **if** $Bernoulli(n_b/(n_b + n_g)) = 1$ **then**
      $\mathcal{S} \leftarrow \mathcal{S} \cup \{a\}$, $n_b \leftarrow n_b - 1$
    **else** $n_g \leftarrow n_g - 1$
**Procedure** DELETE($a$)
  $|\mathcal{D}| \leftarrow |\mathcal{D}| - 1$
  **if** $a \in \mathcal{S}$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \setminus \{a\}$, $n_b \leftarrow n_b + 1$
  **else** $n_g \leftarrow n_g + 1$

RP classifies item deletions into two categories: "bad" uncompensated deletions and "good" uncompensated deletions. An uncompensated deletion is "bad" if the deleted item is in the sample, and thus the deletion decreases the sample size by 1. On the other hand, an uncompensated deletion "good" if the deleted item is not in the sample, and thus the deletion does not affect the sample size. RP maintains counters $n_b$ and $n_g$ to record the number of "bad" and "good" uncompensated deletions, respectively. Undoubtedly, $n_b + n_g$ is equal to the total number of uncompensated deletions.

RP is described in Algorithm 2, where we use $k$ to denote the maximum size of the maintained sample $\mathcal{S}$. For each item deletion from the fully-dynamic dataset $\mathcal{D}$, RP checks whether the deleted item is included in the maintained sample $\mathcal{S}$ or not. If the deleted item is in $\mathcal{S}$, RP removes the item from $\mathcal{S}$ and increases $n_b$ by 1; otherwise, RP simply increases $n_g$ by 1. For each item inserted into $\mathcal{D}$, RP checks whether there are deletions that it needs to compensate for.

- **Case 1:** If there is no uncompensated deletion (i.e., if $n_b + n_g = 0$), then the insertion is proceeded as in reservoir sampling. Specifically, if $\mathcal{S}$ has less than $k$ items, RP adds the inserted item to $\mathcal{S}$. Otherwise, RP replaces a uniformly random item in $\mathcal{S}$ with the newly inserted item with a certain probability.
- **Case 2:** On the other hand, if there are uncompensated deletions (i.e., if $n_b + n_g > 0$), then RP flips a coin. If the coin shows head, whose probability is $n_b/(n_b + n_g)$, RP adds the new item to the sample $\mathcal{S}$ and decreases $n_b$ by 1. Otherwise, RP simply discards the new item and decreases $n_g$ by 1.

Following the above procedure, RP guarantees uniformity (i.e., Eq. (1)) in the presence of arbitrary item insertions and deletions [13]. Unlike reservoir sampling, due to item deletions, RP does not guarantee that $k$ samples are surely stored even if more than $k$ items are processed.

### 3.3 Problem Definition

In this work, we consider the problem of counting the global and local triangles in a graph stream assuming the following realistic conditions:

C1 **No Knowledge:** no information about the input stream (e.g., the node count, the edge count, etc) is available in advance.
C2 **Real Dynamic:** in the input stream, edge insertions and edge deletions arrive in the order by which edges are created and removed in the input graph
C3 **Limited Memory Budget:** we store at most $k$ edges in memory.
C4 **Single Pass:** edge additions and deletions are processed one by one in their order of arrival. Past changes cannot be accessed unless they are stored in memory (within the budget stated in C3).

Based on these conditions, we define the problem of global and local triangle counting in a real graph stream in Problem 1.

**Problem 1 (Global and Local Triangle Counting in a Real Graph Stream)**

(1) **Given:**
- a real graph stream $\{\Delta^{(1)}, \Delta^{(2)}, ...\}$
- a memory budget $k$,

(2) **Estimate:**
- the global triangle count $|\mathcal{T}^{(t)}|$
- the local triangle counts $\{(u, |\mathcal{T}_u^{(t)}|)\}_{u \in \mathcal{V}^{(t)}}$

at current time $t \in \{1, 2, ...\}$

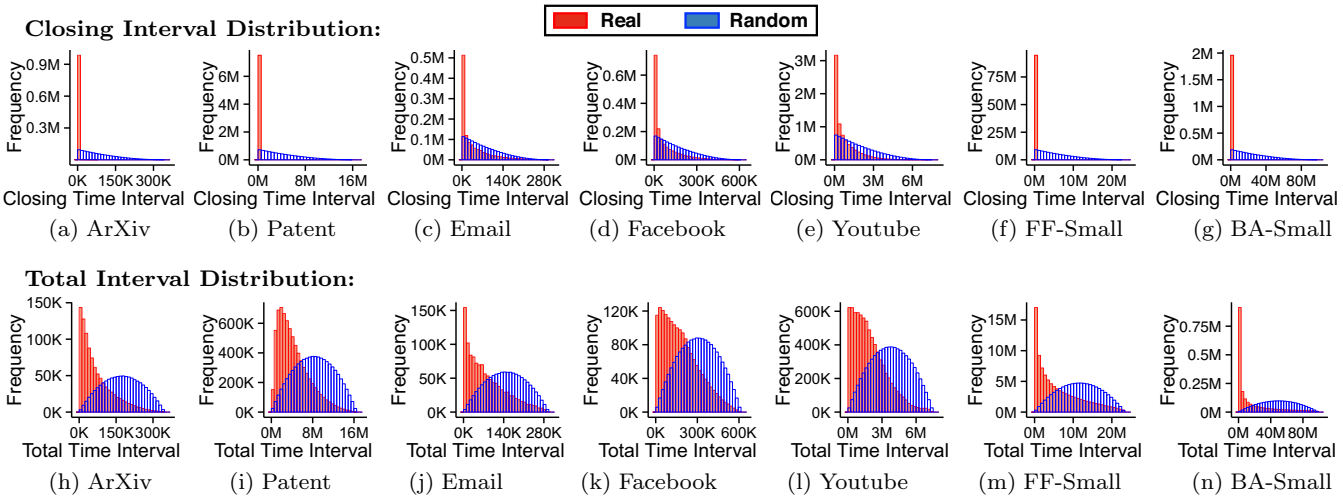



Fig. 2: Temporal locality in the formation of triangles in real graph streams. Closing and total intervals tend to be shorter in real graph streams than in randomly ordered ones. That is, future edges are more likely to form triangles with recent edges than with older edges.

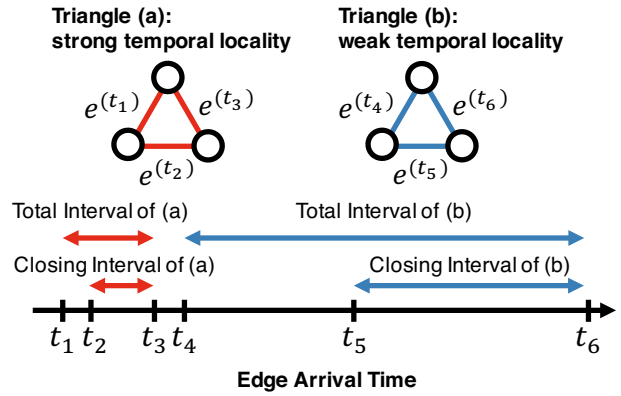



Fig. 3: Pictorial description of total intervals, closing intervals, and temporal locality.

(3) **to Minimize:** the estimation errors.

If $\delta^{(t)} = +$ for every $t \in \{1, 2, ..., \}$, we call Problem 1 *triangle counting in an insertion-only graph stream*. Otherwise, we call Problem 1 *triangle counting in a fully dynamic graph stream*. In Problem 1, instead of minimizing a specific measure of estimation error, we follow a general approach of reducing both bias and variance. This approach is robust to many measures of estimation error, as we show in the experiment section.

## 4 Empirical Pattern: Temporal Locality

In this section, we discuss *temporal locality* (i.e., the tendency that future edges are more likely to form triangles with recent edges than with older edges) in real graph streams. To show the temporal locality, we investigate the distribution of *closing intervals* (see Definition 1) and *total intervals* (see Definition 2) in real graph streams. Figure 3 shows examples of closing and total intervals.

**Definition 1 (Closing Interval)** The *closing interval* of a triangle is defined as the time interval between the arrivals of the second and last edges. That is,

$$closing\_interval(\{u, v, w\}) := t^{(3)}_{uvw} - t^{(2)}_{uvw}.$$

**Definition 2 (Total Interval)** The *total interval* of a triangle is defined as the time interval between the arrivals of the first and last edges. That is,

$$total\_interval(\{u, v, w\}) := t^{(3)}_{uvw} - t^{(1)}_{uvw}.$$

Figure 2 shows the distributions of the closing and total intervals in real graph streams (see Section 6.1 for descriptions of the streams) and random ones obtained by randomly shuffling the orders of the edges in the corresponding real streams. In every dataset, both intervals tend to be much shorter in the real stream than in the randomly ordered one. That is, future edges do not form triangles with all previous edges with equal probability. They are more likely to form triangles with recent edges than with older edges.

Then, why does the temporal locality exist? It is related to *transitivity* [51], i.e., the tendency that people with common friends become friends. When an edge $\{u, v\}$ arrives, we can expect that edges connecting $u$ and other neighbors of $v$ or connecting $v$ and other neighbors of $u$ will arrive soon. These future edges form triangles with the edge $\{u, v\}$. The temporal locality is also related to new nodes. For example, in citation networks, when a new node arrives (i.e., a paper is published), many edges incident to the node (i.e., citations of the paper), which are likely to form triangles with each other, are created almost instantly. Likewise, in

social media, new users make many connections within a short time by importing their friends from other social media or their address books during 'on-boarding' processes.

## 5 Proposed Method: WRS

In this section, we propose *Waiting-Room Sampling* (WRS), a family of two single-pass streaming algorithms that exploit the temporal locality, presented in the previous section, for accurate global and local triangle counting in real graph streams. The two algorithms, namely $\text{WRS}_{\text{INS}}$ and $\text{WRS}_{\text{DEL}}$, are designed for insertion-only graph streams and fully-dynamic streams with edge deletions, respectively. Throughout the paper, we refer to WRS as the set of $\text{WRS}_{\text{INS}}$ and $\text{WRS}_{\text{DEL}}$. We first discuss the intuition behind WRS in Section 5.1. Then, we describe the details of $\text{WRS}_{\text{INS}}$ and $\text{WRS}_{\text{DEL}}$ in Sections 5.2 and 5.3, respectively. After that, we theoretically analyze their accuracy and complexity in Sections 5.4 and 5.5, respectively.

### 5.1 Intuition behind WRS

For accurate estimation, WRS minimizes both the bias and variance of estimates. Reducing the variance is related to finding more triangles because, intuitively speaking, knowing more triangles is helpful to accurately estimate their count. This relation is more formally analyzed in Section 5.4. Thus, the following two goals should be considered when deciding which edges to store in memory:

- **Goal 1.** unbiased estimates of triangle counts should be able to be computed from the stored edges.
- **Goal 2.** when a new edge arrives, it should form many triangles with the stored edges.

Uniform random sampling, such as reservoir sampling and random pairing, achieves Goal 1 but fails to achieve Goal 2, ignoring the temporal locality, described in Section 4. Storing the latest edges, while discarding the older ones, can be helpful to achieve Goal 2, as suggested by the temporal locality. However, simply discarding old edges makes unbiased estimation nontrivial.

To achieve both goals, WRS combines the two policies above. Specifically, it divides the memory space into the *waiting room* and the *reservoir*. The most recent edges are always stored in the waiting room, while the remaining edges are uniformly sampled in the reservoir using reservoir sampling or random pairing. The waiting room enables achieving Goal 2 since it exploits the temporal locality by storing the latest edges, which future edges are more likely to form triangles with. On the other hand, the reservoir enables achieving Goal 1, as described in detail in the following sections.

### 5.2 Details of $\text{WRS}_{\text{INS}}$

In this subsection, we describe $\text{WRS}_{\text{INS}}$, a simple version of WRS for insertion-only graph streams. $\text{WRS}_{\text{INS}}$ provides unbiased estimates of the global and local triangle counts in insertion-only graph streams. We first describe the sampling policy of $\text{WRS}_{\text{INS}}$, which is presented in Figure 4. Then, we describe how to estimate the triangle counts from sampled edges. A pseudo code of $\text{WRS}_{\text{INS}}$ is given in Algorithm 3.

#### *5.2.1 Sampling Policy (Lines 6-13 of Algorithm 3)*

We use $\mathcal{S}$ to denote the given memory space, where at most $k$ edges are stored. WRS divides $\mathcal{S}$ into the *waiting room* $\mathcal{W}$ and the *reservoir* $\mathcal{R}$, and we use $\alpha$ to denote the relative size of $\mathcal{W}$ (i.e., $|\mathcal{W}| = k\alpha$). For simplicity, we assume $k\alpha$ and $k(1-\alpha)$ are integers. In $\mathcal{W}$, the most recent $k\alpha$ edges in the input stream are stored, and some among the edges flowing from $\mathcal{W}$ are stored in the reservoir $\mathcal{R}$. Reservoir sampling, presented in Section 3.2.1, is used to decide which edges to be stored in $\mathcal{R}$. If we let $e^{(t)} = \{u, v\}$ be the edge arriving at time $t \in \{1, 2, \dots\}$ (i.e., $\Delta^{(t)} = (\{u, v\}, +)$), then the sampling scheme of $\text{WRS}_{\text{INS}}$ is described as follows:

- **(Case 1).** If $\mathcal{W}$ is not full, add $e^{(t)}$ to $\mathcal{W}$ (line 6).
- **(Case 2).** If $\mathcal{W}$ is full and $\mathcal{R}$ is not full, $\text{WRS}_{\text{INS}}$ first removes $e^{(t-k\alpha)}$ from $\mathcal{W}$, which is the oldest edge in $\mathcal{W}$, and then stores $e^{(t)}$ in $\mathcal{W}$, which is a queue with the 'first in first out' (FIFO) mechanism (lines 9-10). After that, $\text{WRS}_{\text{INS}}$ stores $e^{(t-k\alpha)}$ in $\mathcal{R}$ (line 11).
- **(Case 3).** If both $\mathcal{W}$ and $\mathcal{R}$ are full, $\text{WRS}_{\text{INS}}$ removes the oldest edge $e^{(t-k\alpha)}$ from $\mathcal{W}$ and inserts the new edge $e^{(t)}$ to $\mathcal{W}$ (lines 9-10). Then, $\text{WRS}_{\text{INS}}$ replaces a uniformly random edge in $\mathcal{R}$ with $e^{(t-k\alpha)}$ with probability $p^{(t)}$, defined as

$$p^{(t)} := \frac{k(1-\alpha)}{|\mathcal{E}_{\mathcal{R}}^{(t)}|}, \tag{2}$$

where $\mathcal{E}_{\mathcal{R}}^{(t)} := \{e^{(1)}, ..., e^{(t-k\alpha)}\}$ is the set of edges flowing from $\mathcal{W}$ to $\mathcal{R}$ at time $t$ or earlier (lines 12-13). That is, reservoir sampling, presented in Section 3.2.1, is used for $\mathcal{R}$, and thus each edge in $\mathcal{E}_{\mathcal{R}}^{(t)}$ is stored in $\mathcal{R}$ with equal probability $p^{(t)}$.

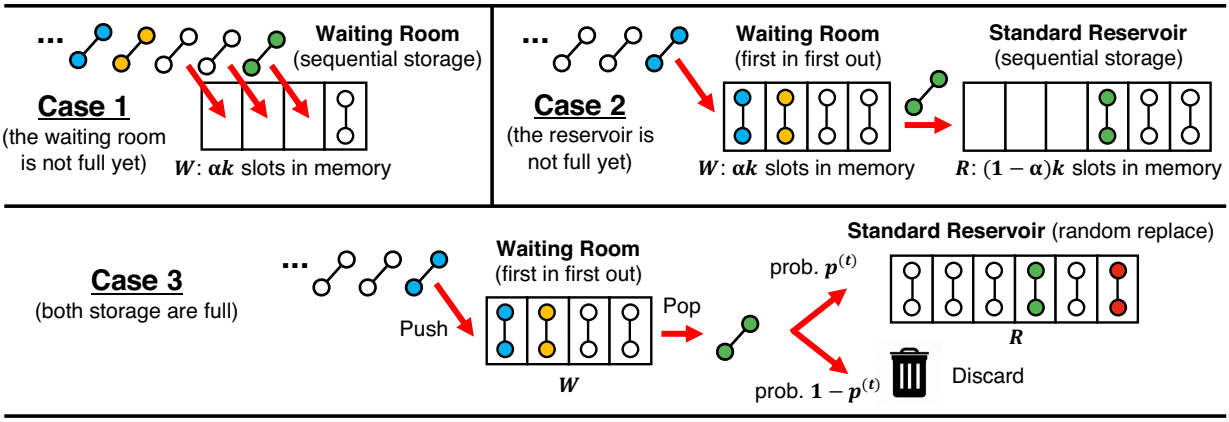



Fig. 4: Pictorial description of the sampling process in WRS$_{\text{INS}}$. Assume a new edge arrives. **Case 1:** If the waiting room is not full, then the new edge is added to the waiting room. **Case 2:** If the waiting room is full while the reservoir is not full, the oldest edge in the waiting room is moved from the waiting room to the reservoir, and the new edge is added to the waiting room. **Case 3:** If both the waiting room and the reservoir are full, the oldest edge in the waiting room is evicted from the waiting room, and the new edge is added to the waiting room. Then, with probability $p^{(t)}$, the evicted edge replaces a uniformly random edge in the reservoir. Note that the latest $|\mathcal{W}|$ edges are stored in the waiting room, while the remaining older edges are uniformly sampled in the reservoir.

**Algorithm 3:** WRS$_{\text{INS}}$: proposed algorithm for insertion-only graph streams

**Input :** (1) insertion-only graph stream: $\{\Delta^{(1)}, \Delta^{(2)}, \dots\}$,
(2) memory budget: $k$,
(3) relative size of the waiting room: $\alpha$
**Output:** (1) estimated global triangle count: $c$,
(2) estimated local triangle counts: $c_u$ for each node $u$

1 $|\mathcal{E}_{\mathcal{R}}| \leftarrow 0$
2 **foreach** $\Delta^{(t)} = (\{u, v\}, +)$ **do**
  /* Update estimates */
3  **foreach** *node* $w$ *in* $\hat{\mathcal{N}}_u \cap \hat{\mathcal{N}}_v$ **do**
    /* Discover the triangle $\{u, v, w\}$ */
4  initialize $c$, $c_u$, $c_v$, and $c_w$ to 0 if they have not been set
5  increase $c$, $c_u$, $c_v$, and $c_w$ by $1/p^{(t)}_{uvw}$
  /* Sample the inserted edge $\{u, v\}$ */
6  **if** $|\mathcal{W}| < k\alpha$ **then** $\mathcal{W} \leftarrow \mathcal{W} \cup \{\{u, v\}\}$
7  **else**
8  $|\mathcal{E}_{\mathcal{R}}| \leftarrow |\mathcal{E}_{\mathcal{R}}| + 1$
9  remove the oldest edge $\{x, y\}$ from $\mathcal{W}$
10  add $\{u, v\}$ to $\mathcal{W}$
11  **if** $|\mathcal{R}| < k(1-\alpha)$ **then** $\mathcal{R} \leftarrow \mathcal{R} \cup \{\{x, y\}\}$
12  **else if** Bernoulli$(k(1-\alpha)/|\mathcal{E}_{\mathcal{R}}|) = 1$ **then**
13  replace a uniformly random edge in $\mathcal{R}$ with $\{x, y\}$

In summary, when $\Delta^{(t)}$ arrives (or after $\Delta^{(t-1)}$ is processed), if $t \leq k+1$, then each edge in $\{e^{(1)}, ..., e^{(t-1)}\}$ is always stored in $\mathcal{W}$ or $\mathcal{R}$. If $t > k+1$, then each edge in $\{e^{(t-k\alpha)}, ..., e^{(t-1)}\}$ is always stored in $\mathcal{W}$, while each edge in $\{e^{(1)}, ..., e^{(t-k\alpha-1)}\}$ is stored in $\mathcal{R}$ with probability $p^{(t-1)}$.

*5.2.2 Estimating Triangle Counts (Lines 3-5 of Algorithm 3)*

We use $\hat{\mathcal{G}} = (\hat{\mathcal{V}}, \hat{\mathcal{E}})$ to denote the sampled graph composed of the edges in $\mathcal{S}$ (i.e., in $\mathcal{W}$ or $\mathcal{R}$), and we use $\hat{\mathcal{N}}_u$ to denote the set of neighbors of each node $u \in \hat{\mathcal{V}}$ in $\hat{\mathcal{G}}$. We use $c$ and $c_u$ to denote the estimates of the global triangle count and the local triangle count of each node $u$, respectively, in the stream so far. That is, if we let $c^{(t)}$ and $c^{(t)}_u$ be $c$ and $c_u$ after processing $\Delta^{(t)}$, they are estimates of $|\mathcal{T}^{(t)}|$ and $|\mathcal{T}^{(t)}_u|$, respectively.

When each edge $e^{(t)} = \{u, v\}$ arrives, WRS$_{\text{INS}}$ first finds the triangles composed of $\{u, v\}$ and two other edges in $\hat{\mathcal{G}}$. The set of such triangles is $\{\{u, v, w\} : w \in \hat{\mathcal{N}}_u \cap \hat{\mathcal{N}}_v\}$. For each such triangle $\{u, v, w\}$, WRS$_{\text{INS}}$ increases $c$, $c_u$, $c_v$, and $c_w$ by $1/p^{(t)}_{uvw}$, where $p^{(t)}_{uvw}$ is the probability that WRS$_{\text{INS}}$ discovers $\{u, v, w\}$ at time $t$, i.e., the probability that $\{v, w\}$ and $\{w, u\}$ are in $\mathcal{S}$ when $\{u, v\}$ arrives (see Lemma 1). Then, the expected increase of the counters by each triangle $\{u, v, w\}$ becomes 1 and the counters become unbiased estimates, as formalized in Theorem 1 in Section 5.4.

The remaining task is to compute $p^{(t)}_{uvw}$, the probability that WRS$_{\text{INS}}$ discovers triangle $\{u, v, w\}$. To this end, we classify triangles into 3 types depending on where the first and second edges are stored when the third edge arrives, as in Definition 3. Recall that $e^{(i)}_{uvw}$ indicates the edge arriving $i$-th among the edges in $\{u, v, w\}$.

**Definition 3 (Types of Triangles)** We define the type of each triangle $\{u, v, w\}$, depending on where the first and the second edges in $\{u, v, w\}$ are stored when

the third edge arrives, as follows:

$$type_{uvw} := \begin{cases} 1 & \text{if } e^{(1)}_{uvw} \in \mathcal{W} \text{ and } e^{(2)}_{uvw} \in \mathcal{W} \\ 2 & \text{else if } e^{(2)}_{uvw} \in \mathcal{W} \\ 3 & \text{otherwise.} \end{cases}$$

Note that a triangle is of Type 1 if its total interval is short. A triangle is of Type 2 if its closing interval is short while its total interval is not. If neither of its intervals is short, a triangle is of Type 3.

The probability that each triangle is discovered by $\text{WRS}_{\text{INS}}$ depends on its type, as formalized in Lemma 1, where the superscript $(t)$ is used to denote the value of each variable after the $t$-th element $\Delta^{(t)}$ is processed.

**Lemma 1 (Triangle Discovering Probability in $\text{WRS}_{\text{INS}}$.)** *Given the maximum size $k$ of sample and the relative size $\alpha$ of the waiting room, the probability $p^{(t)}_{uvw}$ that each new triangle $\{u,v,w\} \in \mathcal{T}^{(t)} - \mathcal{T}^{(t-1)}$ is discovered in line 3 of Algorithm 3 at time $t$ is as follows:*

$$p^{(t)}_{uvw} = \begin{cases} 1 & \text{if } type_{uvw} = 1 \\ \frac{z^{(t-1)}_{\mathcal{R}}}{|\mathcal{E}^{(t-1)}_{\mathcal{R}}|} & \text{if } type_{uvw} = 2 \\ \frac{z^{(t-1)}_{\mathcal{R}}}{|\mathcal{E}^{(t-1)}_{\mathcal{R}}|} \cdot \frac{z^{(t-1)}_{\mathcal{R}}-1}{|\mathcal{E}^{(t-1)}_{\mathcal{R}}|-1} & \text{if } type_{uvw} = 3, \end{cases} \tag{3}$$

*where $z^{(t-1)}_{\mathcal{R}} = min(k(1-\alpha), |\mathcal{E}^{(t-1)}_{\mathcal{R}}|)$.*

*Proof.* Without loss of generality, we assume $e^{(1)}_{uvw} = \{v,w\}$, $e^{(2)}_{uvw} = \{w,u\}$, and $e^{(3)}_{uvw} = e^{(t)} = \{u,v\}$. That is, $\{v,w\}$ arrives earlier than $\{w,u\}$, and $\{w,u\}$ arrives earlier than $\{u,v\}$. When $e^{(t)} = \{u,v\}$ arrives at time $t$, the triangle $\{u,v,w\}$ is discovered if and only if $\{v,w\}$ and $\{w,u\}$ are in $\mathcal{S}^{(t-1)}$.

Note that, since $\text{WRS}_{\text{INS}}$ performs updating the triangle counts before sampling edges, $\mathcal{E}_{\mathcal{R}} = \mathcal{E}^{(t-1)}_{\mathcal{R}}$ at the point of executing line 5 of Algorithm 3.

If $type_{uvw} = 1$, $\{v,w\}$ and $\{w,u\}$ are always stored in $\mathcal{W}^{(t-1)}$, when $\{u,v\}$ arrives. Thus, $\text{WRS}_{\text{INS}}$ discovers $\{u,v,w\}$ with probability 1.

If $type_{uvw} = 2$, when $\{u,v\}$ arrives, $\{w,u\}$ is always stored in $\mathcal{W}^{(t-1)}$, while $\{v,w\}$ cannot be in $\mathcal{W}^{(t-1)}$ but can be in $\mathcal{R}^{(t-1)}$. For $\text{WRS}_{\text{INS}}$ to discover $\{u,v,w\}$, $\{v,w\}$ should be in $\mathcal{R}^{(t-1)}$, and thus the probability is

$$\mathbb{P}[\{v,w\} \in \mathcal{R}^{(t-1)}] = \begin{cases} 1 & \text{if } |\mathcal{E}^{(t-1)}_{\mathcal{R}}| \le k(1-\alpha) \\ \frac{k(1-\alpha)}{|\mathcal{E}^{(t-1)}_{\mathcal{R}}|} & otherwise, \end{cases}$$

or equivalently,

$$\mathbb{P}[\{v,w\} \in \mathcal{R}^{(t-1)}] = \frac{z^{(t-1)}_{\mathcal{R}}}{|\mathcal{E}^{(t-1)}_{\mathcal{R}}|},$$

where $z^{(t-1)}_{\mathcal{R}} = \min(k(1-\alpha), |\mathcal{E}^{(t-1)}_{\mathcal{R}}|)$

If $type_{uvw} = 3$, $\{v,w\}$ and $\{w,u\}$ cannot be in $\mathcal{W}^{(t-1)}$, when $\{u,v\}$ arrives. For $\text{WRS}_{\text{INS}}$ to discover $\{u,v,w\}$, both $\{v,w\}$ and $\{w,u\}$ should be in $\mathcal{R}^{(t-1)}$. The probability of the event is

$$\begin{aligned} &\mathbb{P}[\{v,w\} \in \mathcal{R}^{(t-1)} \text{ and } \{w,u\} \in \mathcal{R}^{(t-1)}] \\ &= \mathbb{P}[\{v,w\} \in \mathcal{R}^{(t-1)}] \cdot \mathbb{P}[\{w,u\} \in \mathcal{R}^{(t-1)} | \{v,w\} \in \mathcal{R}^{(t-1)}] \\ &= \frac{z^{(t-1)}_{\mathcal{R}}}{|\mathcal{E}^{(t-1)}_{\mathcal{R}}|} \cdot \frac{z^{(t-1)}_{\mathcal{R}}-1}{|\mathcal{E}^{(t-1)}_{\mathcal{R}}|-1}. \end{aligned}$$

■

Notice that no additional space is required to store the arrival times of sampled edges. This is because the type of each triangle and its discovering probability (i.e., Eq. (3)) can be computed from the size of $\mathcal{E}_{\mathcal{R}}$ and whether each edge is stored in $\mathcal{W}$ or $\mathcal{R}$ at time $t$, as explained in the proof of Lemma 1. Moreover, since only its size matters, $\mathcal{E}_{\mathcal{R}}$ does not have to be maintained.

### 5.3 Handling Edge Deletion: $\text{WRS}_{\text{DEL}}$

In real-world graphs, there can exist both edge insertions and deletions. There can even be cases where previously deleted edges are re-inserted. In this subsection, we present $\text{WRS}_{\text{DEL}}$, which extends $\text{WRS}_{\text{INS}}$ for triangle counting in fully-dynamic graph streams, which is the sequence of edge insertions and deletions. To this end, the waiting room sampling, described in the previous section, is also extended to handle edge deletions. $\text{WRS}_{\text{DEL}}$ maintains unbiased estimates of both global and local triangle counts. Below, we first present the sampling policy of $\text{WRS}_{\text{DEL}}$, and then we describe how to estimate the triangle counts from sampled edges. A pseudo code of $\text{WRS}_{\text{DEL}}$ is given in Algorithm 4.

#### *5.3.1 Sampling Policy (Lines 9-27 of Algorithm 4)*

For sampling, $\text{WRS}_{\text{DEL}}$ divides the given memory space $\mathcal{S}$, where up to $k$ edges are stored, into the waiting room $\mathcal{W}$ and the reservoir $\mathcal{R}$, where up to $k\alpha$ and $k(1-\alpha)$ are stored, respectively. As in $\text{WRS}_{\text{INS}}$, the latest edges are stored in $\mathcal{W}$, and edges sampled among the other edges are stored in $\mathcal{R}$. Handling deletions of edges in $\mathcal{W}$ is straightforward, and RP, which is described in Section 3.2.2, is used for handling deletions of edges in $\mathcal{R}$ without losing uniformity. Let $\Delta^{(t)} = (e^{(t)} = \{u,v\}, \delta^{(t)})$ be the edge addition or deletion at time $t \in \{1,2,...\}$. Let $n_b$ and $n_g$ be the counters of "bad" and "good" uncompensated deletions, respectively, which is used in RP to record each type of deletions. Below, we describe how $\text{WRS}_{\text{DEL}}$ updates the

**Algorithm 4:** WRS$_{\text{DEL}}$: proposed algorithm for fully-dynamic graph streams

**Input :** (1) fully dynamic graph stream: $\{\Delta^{(1)}, \Delta^{(2)}, ...\}$,
(2) memory budget: $k$,
(3) relative size of the waiting room: $\alpha$
**Output:** (1) estimated global triangle count: $c$,
(2) estimated local triangle counts: $c_u$ for each node $u$

1 $|\mathcal{E}_{\mathcal{R}}| \leftarrow 0$, $n_b \leftarrow 0$, $n_g \leftarrow 0$
2 **foreach** $\Delta^{(t)} = (\{u, v\}, \delta)$ **do**
  /* Update estimates */
3  **foreach** *node* $w$ *in* $\hat{\mathcal{N}}_u \cap \hat{\mathcal{N}}_v$ **do**
    /* Discover the triangle $\{u, v, w\}$ */
4  initialize $c$, $c_u$, $c_v$, and $c_w$ to 0 if they have not been set
5  **if** $\delta = +$ **then**
6  increase $c$, $c_u$, $c_v$, and $c_w$ by $1/q^{(t)}_{uvw}$
7  **else if** $\delta = -$ **then**
8  decrease $c$, $c_u$, $c_v$, and $c_w$ by $1/q^{(t)}_{uvw}$
  /* Sample the inserted edge $\{u, v\}$ */
9  **if** $\delta = +$ **then**
10  **if** $|\mathcal{W}| < k\alpha$ **then** $\mathcal{W} \leftarrow \mathcal{W} \cup \{\{u, v\}\}$
11  **else**
12  $|\mathcal{E}_{\mathcal{R}}| \leftarrow |\mathcal{E}_{\mathcal{R}}| + 1$
13  remove the oldest edge $\{x, y\}$ from $\mathcal{W}$ and add $\{u, v\}$ to $\mathcal{W}$
14  **if** $n_b + n_g = 0$ **then**
15  **if** $|\mathcal{R}| < k(1-\alpha)$ **then** $\mathcal{R} \leftarrow \mathcal{R} \cup \{\{x, y\}\}$
16  **else if** Bernoulli$(k(1-\alpha)/|\mathcal{E}_{\mathcal{R}}|) = 1$ **then**
17  replace a randomly chosen edge in $\mathcal{R}$ with $\{x, y\}$
18  **else if** Bernoulli$(n_b/(n_b + n_g)) = 1$ **then**
19  $\mathcal{R} \leftarrow \mathcal{R} \cup \{\{x, y\}\}$, $n_b \leftarrow n_b - 1$
20  **else** $n_g \leftarrow n_g - 1$
  /* Delete the removed edge $\{u, v\}$ */
21  **else if** $\delta = -$ **then**
22  **if** $\{u, v\} \in \mathcal{W}$ **then** $\mathcal{W} \leftarrow \mathcal{W} \setminus \{\{u, v\}\}$
23  **else**
24  $|\mathcal{E}_{\mathcal{R}}| \leftarrow |\mathcal{E}_{\mathcal{R}}| - 1$
25  **if** $\{u, v\} \in \mathcal{R}$ **then**
26  $\mathcal{R} \leftarrow \mathcal{R} \setminus \{\{u, v\}\}$, $n_b \leftarrow n_b + 1$
27  **else** $n_g \leftarrow n_g + 1$

maintained sample (i.e., edges stored in $\mathcal{W}$ and $\mathcal{R}$) in response to edge additions and deletions.

**For Edge Insertions:** Whenever an insertion of an edge arrives, WRS$_{\text{DEL}}$ checks whether $\mathcal{W}$ is full or not.

- **(Case 1).** If $\mathcal{W}$ is not full, the new edge is stored at $\mathcal{W}$ (line 10).
- **(Case 2).** If $\mathcal{W}$ is full, WRS$_{\text{DEL}}$ removes the oldest edge in $\mathcal{W}$ and inserts the new edge to $\mathcal{W}$ by the FIFO mechanism (line 13), and then WRS$_{\text{DEL}}$ decides whether to store the edge that is removed from $\mathcal{W}$ in $\mathcal{R}$ or not, following the RP procedure (lines 14-20), which is described in Section 3.2.2. Specifically, this case is further divided into two subcases.
- **(Case 2-1).** If there is no deletion to be compensated (line 14), RP processes each edge addition as reservoir sampling does (lines 15-17). That is, if the reservoir is not full (i.e., $|\mathcal{R}| < k(1-\alpha)$), RP adds the popped edge to $\mathcal{R}$, Otherwise, RP replaces a uniformly random edge in $\mathcal{R}$ with the popped edge with probability $(k(1-\alpha))/|\mathcal{E}_{\mathcal{R}}|$.
- **(Case 2-2).** If there are deletions that need to be compensated, then RP tosses a coin (line 18). If the coin shows head, whose probability is $n_b/(n_b + n_g)$, RP adds the popped edge to $\mathcal{R}$. Otherwise, RP discards the popped edge. Next, RP decreases either $n_b$ or $n_g$ by 1, depending on whether the popped edge has been added to $\mathcal{R}$ or not (lines 19-20).

**For Edge Deletions:** Whenever a deletion of an edge arrives, WRS$_{\text{DEL}}$ first checks whether the edge is in $\mathcal{W}$ or not. If the edge is stored in $\mathcal{W}$, it is simply removed from $\mathcal{W}$ (line 22). Otherwise, WRS$_{\text{DEL}}$ removes the edge from $\mathcal{R}$ (if it is in $\mathcal{R}$) and increases $n_b$ or $n_g$ depending on whether the edge was in $\mathcal{R}$ or not (lines 24-27).

#### *5.3.2 Estimating Triangle Counting (Lines 3-8 of Algorithm 4)*

When each edge insertion or deletion $\Delta^{(t)} = (\{u, v\}, \delta^{(t)})$ arrives, WRS$_{\text{DEL}}$ first finds the triangles composed of $\{u, v\}$ and two edges in $\hat{\mathcal{G}}$. The set of such triangles is $\{\{u, v, w\} : w \in \hat{\mathcal{N}}_u \cap \hat{\mathcal{N}}_v\}$. For each triangle $\{u, v, w\}$, WRS$_{\text{DEL}}$ increases or decreases $c$, $c_u$, $c_v$, and $c_w$ by $1/q^{(t)}_{uvw}$, where $q^{(t)}_{uvw}$ is the probability that WRS$_{\text{DEL}}$ discovers $\{u, v, w\}$ at time $t$ (see Lemma 2) in line 3 of Algorithm 4, depending on whether the edge is inserted (i.e., $\delta^{(t)} = +$) or deleted (i.e., $\delta^{(t)} = -$). Then, the expected increase or decrease in each counter by each triangle $\{u, v, w\}$ becomes 1, which makes the counters unbiased estimates, as formalized in Theorem 2 in the following section.

In Lemma 2, the probability $q^{(t)}_{uvw}$ that WRS$_{\text{DEL}}$ discovers the triangle $\{u, v, w\}$ at time $t$ is formulated. To this end, in Definitions 4 and 5, we define two concepts: *added triangles* and *deleted triangles* in a fully dynamic graph stream. The added triangles and deleted triangles are the sets of all triangles that have been added to and deleted from the input graph, respectively. In a fully dynamic graph stream, if the same edge arrives again after being deleted, a triangle can be added or deleted multiple times. To distinguish the same triangle added and deleted at different times, we use $\{u, v, w\}^{(t)}$ to denote the triangle $\{u, v, w\}$ added or deleted at time $t$. Similarly, we use the superscript

$(t)$ to denote the value of each variable after the $t$-th element $\Delta^{(t)}$ is processed.

**Definition 4 (Added Triangles)** $\mathcal{A}^{(t)}$ is defined as the set of triangles that have been added to the graph $\mathcal{G}$ at time $t$ or earlier. That is,

$$\mathcal{A}^{(t)} := \{\{u,v,w\}^{(s)} : \{u,v,w\} \notin \mathcal{T}^{(s-1)}, \{u,v,w\} \in \mathcal{T}^{(s)}, where\ 1 \le s \le t\}.$$

**Definition 5 (Deleted Triangles)** $\mathcal{D}^{(t)}$ is defined as the set of triangles that have been deleted from the graph $\mathcal{G}$ at time $t$ or earlier. That is,

$$\mathcal{D}^{(t)} := \{\{u,v,w\}^{(s)} : \{u,v,w\} \in \mathcal{T}^{(s-1)}, \{u,v,w\} \notin \mathcal{T}^{(s)}, where\ 1 \le s \le t\}.$$

By the definition of added triangles, the set of triangles added at time $t$ is $\mathcal{A}^{(t)} - \mathcal{A}^{(t-1)}$. Likewise, by the definition of deleted triangles, the set of triangles deleted at time $t$ is $\mathcal{D}^{(t-1)} - \mathcal{D}^{(t)}$.

**Lemma 2 (Triangle Discovering Probability in $\mathbf{WRS_{DEL}}$.)** *Given the maximum size $k$ of sample and the relative size $\alpha$ of the waiting room, the probability $q^{(t)}_{uvw}$ that each triangle $\{u,v,w\} \in (\mathcal{A}^{(t)} - \mathcal{A}^{(t-1)}) \cup (\mathcal{D}^{(t-1)} - \mathcal{D}^{(t)})$ is discovered in line 3 of Algorithm 4 at time $t$ is as follows:*

$$q^{(t)}_{uvw} = \begin{cases} 1 & \text{if } type_{uvw} = 1 \\ \frac{y^{(t-1)}_{\mathcal{R}}}{|\mathcal{E}^{(t-1)}_{\mathcal{R}}|+d^{(t-1)}} & \text{if } type_{uvw} = 2 \\ \frac{y^{(t-1)}_{\mathcal{R}}}{|\mathcal{E}^{(t-1)}_{\mathcal{R}}|+d^{(t-1)}} \cdot \frac{y^{(t-1)}_{\mathcal{R}}-1}{|\mathcal{E}^{(t-1)}_{\mathcal{R}}|+d^{(t-1)}-1} & \text{if } type_{uvw} = 3, \end{cases} \tag{4}$$

*where $d^{(t-1)} = n^{(t-1)}_b + n^{(t-1)}_g$ and $y^{(t-1)}_{\mathcal{R}} = \min(k(1-\alpha), |\mathcal{E}^{(t-1)}_{\mathcal{R}}| + d^{(t-1)})$.*

*Proof.* $\text{WRS}_{\text{DEL}}$ discovers a triangle $\{u,v,w\}^{(t)}$ added or removed at time $t$ if any only if the other two edges are stored in $\mathcal{S}$ when an edge forming the triangle is added or removed at time $t$. Thus, the probability $q^{(t)}_{uvw}$ of discovering $\{u,v,w\}^{(t)}$ is equal to the probability that the other two edges are stored in $\mathcal{S}^{(t-1)}$. A full proof with the derivation of the probability is given in Appendix A. ■

Recall that the type of each triangle is determined simply by whether each edge is stored in $\mathcal{W}$ or $\mathcal{R}$. Moreover, Eq. (4) is directly calculated from the three counters (i.e., $n_b$, $n_g$ and $|\mathcal{E}_{\mathcal{R}}|$). Thus, $\text{WRS}_{\text{DEL}}$ requires no additional space to store any additional information or the arrival times of sampled edges. Especially, since only its size matters, $\mathcal{E}_{\mathcal{R}}$ does not have to be maintained.

### 5.4 Accuracy Analysis

In this subsection, we prove that WRS maintains unbiased estimates, whose expected values are equal to the true global and local triangle counts. Then, we analyze the variance of the estimates provided by $\text{WRS}_{\text{INS}}$. Throughout this section, we use the superscript $(t)$ to denote the value of each variable after the $t$-th element $\Delta^{(t)}$ is processed.

#### *5.4.1 Bias Analysis*

We prove the unbiasedness of the estimates given by $\text{WRS}_{\text{INS}}$ in Theorem 1.

**Theorem 1 (Unbiasedness of $\mathbf{WRS_{INS}}$.)** *If the input graph stream is insertion only and $k(1-\alpha) \ge 2$, then $\text{WRS}_{\text{INS}}$ gives unbiased estimates of the global and local triangle counts at any time $t$. That is, if we let $c^{(t)}$ and $c^{(t)}_u$ be $c$ and $c_u$ after processing $\Delta^{(t)}$, respectively, then the followings hold:*

$$\mathbb{E}[c^{(t)}] = |\mathcal{T}^{(t)}|,\ \forall t \ge 1 \tag{5}$$

$$\mathbb{E}[c^{(t)}_u] = |\mathcal{T}^{(t)}_u|,\ \forall u \in \mathcal{V}^{(t)},\ \forall t \ge 1. \tag{6}$$

*Proof.* Let $x^{(s)}_{uvw}$ be the amount of increase in each of $c^{(t)}$, $c^{(t)}_u$, $c^{(t)}_v$, and $c^{(t)}_w$ due to the discovery of $\{u,v,w\}^{(s)}$ (i.e., $\{u,v,w\}$ added at time $s$) where $s \le t$. Without loss of generality, we assume $e^{(3)}_{uvw} = e^{(s)} = \{u,v\}$ and thus $\Delta^{(s)} = (\{u,v\},+)$. Then, the following holds:

$$x^{(s)}_{uvw} = \begin{cases} 1/p^{(s)}_{uvw} & \text{if } \{v,w\} \text{ and } \{w,u\} \in \mathcal{S}^{(s-1)} \\ 0 & \text{otherwise,} \end{cases}$$

From this and Lemma 1, we have $\mathbb{E}[x^{(s)}_{uvw}] = 1/p^{(s)}_{uvw} \times p^{(s)}_{uvw} + 0 \times (1 - p^{(s)}_{uvw}) = 1$. Combining this and $c^{(t)} = \sum_{\{u,v,w\}^{(s)} \in \mathcal{T}^{(t)}} x^{(s)}_{uvw}$ gives

$$\mathbb{E}[c^{(t)}] = \mathbb{E}\left[\sum_{\{u,v,w\}^{(s)} \in \mathcal{T}^{(t)}} x^{(s)}_{uvw}\right] = \sum_{\{u,v,w\}^{(s)} \in \mathcal{T}^{(t)}} \mathbb{E}[x^{(s)}_{uvw}] = |\mathcal{T}^{(t)}|,$$

which proves Eq. (5). Likewise, combining $\mathbb{E}[x^{(s)}_{uvw}] = 1$ and $c^{(t)}_u = \sum_{\{u,v,w\}^{(s)} \in \mathcal{T}^{(t)}_u} x^{(s)}_{uvw}$ gives

$$\mathbb{E}[c^{(t)}_u] = \mathbb{E}\left[\sum_{\{u,v,w\}^{(s)} \in \mathcal{T}^{(t)}_u} x^{(s)}_{uvw}\right] = \sum_{\{u,v,w\}^{(s)} \in \mathcal{T}^{(t)}_u} \mathbb{E}[x^{(s)}_{uvw}] = |\mathcal{T}^{(t)}_u|,$$

which proves Eq. (6). ■

Before showing the unbiasedness of $\text{WRS}_{\text{DEL}}$, we extend the concept of added triangles and deleted triangles in Definitions 4 and 5 to the node level. We denote the sets of added and deleted triangles with each

node $u \in \mathcal{V}^{(t)}$ by $\mathcal{A}_u^{(t)} \subseteq \mathcal{A}^{(t)}$ and $\mathcal{D}_u^{(t)} \subseteq \mathcal{D}^{(t)}$, respectively. Then, from their counts, the number of triangles remaining without being deleted is obtained, as formalized in Lemma 3.

**Lemma 3 (Count of Triangles [38])** *Subtracting the number of deleted triangles from the number of added triangles gives the number of triangles in the input stream. Formally,*

$$|\mathcal{T}^{(t)}| = |\mathcal{A}^{(t)}| - |\mathcal{D}^{(t)}|, \ \forall t \geq 1, \tag{7}$$

$$|\mathcal{T}_u^{(t)}| = |\mathcal{A}_u^{(t)}| - |\mathcal{D}_u^{(t)}|, \ \forall t \geq 1, \ \forall u \in \mathcal{V}^{(t)}. \tag{8}$$

Based on Lemma 3, we prove the unbiasedness of the estimates given by $\text{WRS}_{\text{DEL}}$ in fully dynamic graph streams in Theorem 2. Note that Theorem 2 is a generalization of Theorem 1.

**Theorem 2 (Unbiasedness of $\text{WRS}_{\text{DEL}}$.)** *If $k(1-\alpha) \geq 2$, then $\text{WRS}_{\text{DEL}}$ gives unbiased estimates of the global and local triangle counts at any time $t$. That is, if we let $c^{(t)}$ and $c_u^{(t)}$ be $c$ and $c_u$ after processing $\Delta^{(t)}$, respectively, then the followings hold:*

$$\mathbb{E}[c^{(t)}] = |\mathcal{T}^{(t)}|, \ \forall t \geq 1, \tag{9}$$

$$\mathbb{E}[c_u^{(t)}] = |\mathcal{T}_u^{(t)}|, \ \forall u \in \mathcal{V}^{(t)}, \ \forall t \geq 1. \tag{10}$$

*Proof.* Let $x_{uvw}^{(s)}$ be the amount of increase in each of $c^{(t)}$, $c_u^{(t)}$, $c_v^{(t)}$, and $c_w^{(t)}$ due to the discovery of $\{u,v,w\}^{(s)} \in \mathcal{A}^{(t)}$ (i.e., $\{u,v,w\}$ added at time $s$) where $s \leq t$. Without loss of generality, we assume $e_{uvw}^{(3)} = e^{(s)} = \{u,v\}$ and thus $\Delta^{(s)} = (\{u,v\},+)$. Then, the following holds:

$$x_{uvw}^{(s)} = \begin{cases} 1/q_{uvw}^{(s)} & \text{if } \{v,w\} \textit{ and } \{w,u\} \in \mathcal{S}^{(s-1)} \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

Similarly, let $y_{uvw}^{(s)}$ be the amount of decrease in each of $c^{(t)}$, $c_u^{(t)}$, $c_v^{(t)}$, and $c_w^{(t)}$ due to the discovery of $\{u,v,w\}^{(s)} \in \mathcal{D}^{(t)}$ (i.e., $\{u,v,w\}$ deleted at time $s$) where $s \leq t$. Without loss of generality, we assume $e_{uvw}^{(3)} = e^{(s)} = \{u,v\}$ and thus $\Delta^{(s)} = (\{u,v\},-)$. Then, the following holds:

$$y_{uvw}^{(s)} = \begin{cases} -1/q_{uvw}^{(s)} & \text{if } \{v,w\} \textit{ and } \{w,u\} \in \mathcal{S}^{(s-1)} \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

Eq. (11), Eq. (12), and Lemma 2, which states that $\mathbb{P}[\{v,w\},\{w,u\} \in \mathcal{S}^{(s-1)}]$ is equal to $q_{uvw}^{(s)}$, imply

$$\mathbb{E}[x_{uvw}^{(s)}] = 1 \text{ and } \mathbb{E}[y_{uvw}^{(s)}] = -1. \tag{13}$$

Definition 4, Definition 5, and Algorithm 4 imply

$$c^{(t)} = \sum_{\{u,v,w\}^{(s)} \in \mathcal{A}^{(t)}} x_{uvw}^{(s)} + \sum_{\{u,v,w\}^{(s)} \in \mathcal{D}^{(t)}} y_{uvw}^{(s)}. \tag{14}$$

Combining this, linearity of expectation, Eq. (7) in Lemma 3, Eq. (13), and Eq. (14) gives

$$\begin{aligned} \mathbb{E}[c^{(t)}] &= \sum_{\{u,v,w\}^{(s)} \in \mathcal{A}^{(t)}} \mathbb{E}[x_{uvw}^{(s)}] + \sum_{\{u,v,w\}^{(s)} \in \mathcal{D}^{(t)}} \mathbb{E}[y_{uvw}^{(s)}] \\ &= \sum_{\{u,v,w\}^{(s)} \in \mathcal{A}^{(t)}} 1 + \sum_{\{u,v,w\}^{(s)} \in \mathcal{D}^{(t)}} -1 \\ &= |\mathcal{A}^{(t)}| - |\mathcal{D}^{(t)}| = |\mathcal{T}^{(t)}|. \end{aligned}$$

Hence, Eq. (9) holds. Likewise, for each node $u \in \mathcal{V}^{(t)}$, the following equation holds:

$$c_u^{(t)} = \sum_{\{u,v,w\}^{(s)} \in \mathcal{A}_u^{(t)}} x_{uvw}^{(s)} + \sum_{\{u,v,w\}^{(s)} \in \mathcal{D}_u^{(t)}} y_{uvw}^{(s)}. \tag{15}$$

Combining this, linearity of expectation, Eq. (8) in Lemma 3, Eq. (13), and Eq. (15) gives

$$\begin{aligned} \mathbb{E}[c_u^{(t)}] &= \sum_{\{u,v,w\}^{(s)} \in \mathcal{A}_u^{(t)}} \mathbb{E}[x_{uvw}^{(s)}] + \sum_{\{u,v,w\}^{(s)} \in \mathcal{D}_u^{(t)}} \mathbb{E}[y_{uvw}^{(s)}] \\ &= \sum_{\{u,v,w\}^{(s)} \in \mathcal{A}_u^{(t)}} 1 + \sum_{\{u,v,w\}^{(s)} \in \mathcal{D}_u^{(t)}} -1 \\ &= |\mathcal{A}_u^{(t)}| - |\mathcal{D}_u^{(t)}| = |\mathcal{T}_u^{(t)}| \end{aligned}$$

Hence, Eq. (10) holds. ■

### 5.4.2 Variance Analysis

We present a variance analysis through which we shed light on the effectiveness of WRS. For simplicity, we focus on the variance of the estimates provided by $\text{WRS}_{\text{INS}}$ and specifically the following:

$$\tilde{\text{Var}}[c^{(t)}] = \sum\nolimits_{\{u,v,w\}^{(s)} \in \mathcal{T}^{(t)}} \text{Var}[x_{uvw}^{(s)}], \tag{16}$$

$$\tilde{\text{Var}}[c_u^{(t)}] = \sum\nolimits_{\{u,v,w\}^{(s)} \in \mathcal{T}_u^{(t)}} \text{Var}[x_{uvw}^{(s)}], \tag{17}$$

where the dependencies between $\{x_{uvw}^{(s)}\}_{\{u,v,w\}^{(s)} \in \mathcal{T}^{(t)}}$ are ignored. In real-world graphs, the variance $\text{Var}[c^{(t)}]$ and the simplified version $\tilde{\text{Var}}[c^{(t)}]$ (i.e., Eq. (16)) are strongly correlated ($R^2 > 0.99$), as discussed in detail in Appendix B.

To show how the temporal locality, described in Section 4, is related to reducing $\tilde{\text{Var}}[c^{(t)}]$ and $\tilde{\text{Var}}[c_u^{(t)}]$, we first formulate $\text{Var}[x_{uvw}^{(t)}]$, which is summed in Eq. (16) and Eq. (17), in Lemma 4.

**Lemma 4 (Variance of $x_{uvw}$ in $\text{WRS}_{\text{INS}}$)** *If $k(1-\alpha) \geq 2$, the variance of the increment $x_{uvw}$ in $c$ by each triangle $\{u,v,w\} \in \mathcal{T}^{(t)} - \mathcal{T}^{(t-1)}$ in $\text{WRS}_{\text{INS}}$ is*

$$\text{Var}[x_{uvw}^{(t)}] = \begin{cases} 0 & \text{if } type_{uvw} = 1 \\ \frac{|\mathcal{E}_{\mathcal{R}}^{(t-1)}|}{z_{\mathcal{R}}^{(t-1)}} - 1 & \text{if } type_{uvw} = 2 \\ \frac{|\mathcal{E}_{\mathcal{R}}^{(t-1)}|}{z_{\mathcal{R}}^{(t-1)}} \times \frac{|\mathcal{E}_{\mathcal{R}}^{(t-1)}|-1}{z_{\mathcal{R}}^{(t-1)}-1} - 1 & \text{if } type_{uvw} = 3, \end{cases}$$

(18)

*where* $z_{\mathcal{R}}^{(t-1)} = \min(k(1-\alpha), |\mathcal{E}_{\mathcal{R}}^{(t-1)}|)$.

*Proof.* From $\text{Var}[x_{uvw}^{(t)}] = \mathbb{E}[(x_{uvw}^{(t)})^2] - \left(\mathbb{E}[x_{uvw}^{(t)}]\right)^2$, $\text{Var}[x_{uvw}^{(t)}] = (1/p_{uvw}^{(t)}) - 1$ holds. This and Eq. (3) imply that, if $k(1-\alpha) \geq 2$, the variance of $x_{uvw}^{(t)}$ is equal to Eq. (18). ■

As formalized in Lemma 5, compared to TRIÈSTIMPR [8], where

$$\text{Var}[x_{uvw}^{(t)}] = \frac{|\mathcal{E}^{(t-1)}|}{m^{(t-1)}} \times \frac{|\mathcal{E}^{(t-1)}| - 1}{m^{(t-1)} - 1} - 1$$

and $m^{(t-1)} = \min(k, |\mathcal{E}^{(t-1)}|)$ for every triangle added at time $t$ regardless of its type, $\text{WRS}_{\text{INS}}$ reduces the variance regarding the triangles of Type 1 and 2, while increasing the variance regarding the triangles of Type 3. Note that, in both $\text{WRS}_{\text{INS}}$ and TRIÈSTIMPR, the variance regarding the triangles added at time $t = k+1$ or earlier is 0 since every edge is stored without being discarded at time $t = k+1$ or earlier.

**Lemma 5 (Comparison of Variances between $\text{WRS}_{\text{INS}}$ and TRIÈST$_{\text{IMPR}}$)** *At any time $t > k+1$, for each triangle $\{u, v, w\} \in \mathcal{T}^{(t)} - \mathcal{T}^{(t-1)}$, $\text{Var}[x_{uvw}^{(t)}]$ is smaller in $\text{WRS}_{\text{INS}}$ than in* TRIÈST$_{\text{IMPR}}$ [8], i.e.,

$$\text{Var}[x_{uvw}^{(t)}] < \frac{|\mathcal{E}^{(t-1)}|}{m^{(t-1)}} \times \frac{|\mathcal{E}^{(t-1)}| - 1}{m^{(t-1)} - 1} - 1, \tag{19}$$

where $m^{(t-1)} = \min(k, |\mathcal{E}^{(t-1)}|)$, if **any** of the following conditions are satisfied:

- $type_{uvw} = 1$
- $type_{uvw} = 2$ and $t > 1 + \frac{\alpha}{1-\alpha}k$
- $type_{uvw} = 2$ and $\alpha < 0.5$.

*Proof.* From $t > k+1$ and $|\mathcal{E}_{\mathcal{R}}^{(t-1)}| = |\mathcal{E}^{(t-1)}| - k\alpha$, the following (in)equalities hold:

$$|\mathcal{E}^{(t-1)}| = t - 1 > k, \tag{20}$$

$$m^{(t-1)} = \min(k, |\mathcal{E}^{(t-1)}|) = k, \tag{21}$$

$$z_{\mathcal{R}}^{(t-1)} = \min(k(1-\alpha), |\mathcal{E}_{\mathcal{R}}^{(t-1)}|) = k(1-\alpha). \tag{22}$$

First, Eq. (18), Eq. (20), and Eq. (21) imply

$$\text{Var}[x_{uvw}^{(t)}] = 0 < \frac{|\mathcal{E}^{(t-1)}|}{m^{(t-1)}} \times \frac{|\mathcal{E}^{(t-1)}| - 1}{m^{(t-1)} - 1} - 1,$$

which proves Eq. (19) when $type_{uvw} = 1$.

Second, we prove Eq. (19) when $type_{uvw} = 2$ and $t > 1 + \frac{\alpha}{1-\alpha}k$. From $t > 1 + \frac{\alpha}{1-\alpha}k$, $|\mathcal{E}^{(t-1)}| > \frac{\alpha}{1-\alpha}k$ and thus $\left(1 + \frac{k}{|\mathcal{E}^{(t-1)}|}\right)\alpha < 1$ hold. This and Eq. (20) imply

$$\left(1 + \frac{k}{|\mathcal{E}^{(t-1)}|}\right)\left(1 - \frac{k-1}{|\mathcal{E}^{(t-1)}| - 1}\right)\alpha < 1 - \frac{k-1}{|\mathcal{E}^{(t-1)}| - 1}.$$

This and Eq. (20) again imply

$$\begin{aligned}
&\left(1 - \frac{k(k-1)}{|\mathcal{E}^{(t-1)}|(|\mathcal{E}^{(t-1)}| - 1)}\right)\alpha \\
&\leq \left(1 + \frac{k}{|\mathcal{E}^{(t-1)}|}\right)\left(1 - \frac{k-1}{|\mathcal{E}^{(t-1)}| - 1}\right)\alpha \\
&< 1 - \frac{k-1}{|\mathcal{E}^{(t-1)}| - 1}.
\end{aligned}$$

This is equivalent to

$$(k-1)(|\mathcal{E}^{(t-1)}| - k\alpha) < |\mathcal{E}^{(t-1)}|(|\mathcal{E}^{(t-1)}| - 1)(1-\alpha),$$

which is again equivalent to

$$\frac{|\mathcal{E}^{(t-1)}| - k\alpha}{k(1-\alpha)} - 1 < \frac{|\mathcal{E}^{(t-1)}|}{k} \times \frac{|\mathcal{E}^{(t-1)}| - 1}{k-1} - 1.$$

Combining this, $|\mathcal{E}^{(t-1)}| = |\mathcal{E}_{\mathcal{R}}^{(t-1)}| + k\alpha$, Eq. (18), and Eq. (22) gives

$$\text{Var}[x_{uvw}^{(t)}] = \frac{|\mathcal{E}_{\mathcal{R}}^{(t-1)}|}{k(1-\alpha)} - 1 < \frac{|\mathcal{E}^{(t-1)}|}{k} \times \frac{|\mathcal{E}^{(t-1)}| - 1}{k-1} - 1,$$

which proves Eq. (19).

Lastly, the same conclusion holds when $type_{uvw} = 2$ and $\alpha < 0.5$ since Eq. (20) and $\alpha < 0.5$ imply $|\mathcal{E}^{(t-1)}| > \frac{\alpha}{1-\alpha}k$, which with $type_{uvw} = 2$ corresponds to the second case, which we prove above.

Hence, Eq. (19) holds under any of the given conditions. ■

Therefore, the superiority of $\text{WRS}_{\text{INS}}$ in terms of small $\tilde{\text{Var}}[c^{(t)}]$ and $\tilde{\text{Var}}[c_u^{(t)}]$ depends on the distribution of the types of triangles in real graph streams. In the experiment section, we show that the triangles of Type 1 and 2 are abundant enough in real graph streams, as suggested by the temporal locality, so that $\text{WRS}_{\text{INS}}$ is more accurate than TRIÈST$_{\text{IMPR}}$.

Similarly, the abundance of triangles of Type 1 and 2 explains why $\text{WRS}_{\text{DEL}}$ is more accurate then its best competitors in fully dynamic graph streams, as shown experimentally in Section 6.

### 5.5 Complexity Analysis

In this subsection, we prove the time and space complexities of WRS. Especially, we show that WRS has the same time and space complexities as the state-of-the-art algorithms [27,8]. We assume that sampled edges are stored in the adjacency list format in memory, as in our implementation used for our experiments. However, storing them sequentially, as in Figure 4, does not change the results below.

*5.5.1 Time Complexity Analysis*

The worst-case time complexity of WRS is linear in the memory budget and in the number of edges in the input stream, as formalized in Theorem 3.

**Theorem 3 (Worst-Case Time Complexity of WRS)** *Processing each element in the input stream by Algorithms 3 and 4 takes $O(k)$, and thus processing $t$ elements in the input stream takes $O(kt)$.*

*Proof.* In both Algorithms 3 and 4, the most expensive step of processing an inserted or deleted edge $\Delta^{(t)} = (\{u, v\}, \delta^{(t)})$ is to find the common neighbors $\hat{\mathcal{N}}_u \cap \hat{\mathcal{N}}_v$ (line 3 in Algorithm 3 and line 3 in Algorithm 4). Assume $|\hat{\mathcal{N}}_u| \leq |\hat{\mathcal{N}}_v|$, without loss of generality, and a hash table is used to store all elements of $\hat{\mathcal{N}}_v$. $\hat{\mathcal{N}}_u \cap \hat{\mathcal{N}}_v$ is obtained by traversing all elements of $\hat{\mathcal{N}}_u$ and checking if each of them is in the hash table. Since checking the membership of an element in a hash table takes $O(1)$ time, the overall time complexity of finding the common neighbors is $O(|\hat{\mathcal{N}}_u|) = O(\min(|\hat{\mathcal{N}}_u|, |\hat{\mathcal{N}}_v|))$ time. Except for computing $\hat{\mathcal{N}}_u \cap \hat{\mathcal{N}}_v$, the others steps for processing $\Delta^{(t)} = (\{u, v\}, \delta^{(t)})$ take $O(|\hat{\mathcal{N}}_u \cap \hat{\mathcal{N}}_v|) = O(\min(|\hat{\mathcal{N}}_u|, |\hat{\mathcal{N}}_v|))$ time in total. Since $O(\min(|\hat{\mathcal{N}}_u|, |\hat{\mathcal{N}}_v|)) = O(k)$, the time complexity of processing each element is $O(k)$. ■

Notice that this analysis assuming the worst-case graph stream is pessimistic for real graph streams, where $\min(|\hat{\mathcal{N}}_u|, |\hat{\mathcal{N}}_v|)$ is usually much smaller than $k$.

*5.5.2 Space Complexity Analysis*

Theorem 4 gives the space complexity of WRS. Note that, except for the space for outputs (specifically, estimates of the local triangle counts), WRS only requires $O(k)$ space.

**Theorem 4 (Space Complexity of WRS)** *Let $\mathcal{V}^{(t)}$ be the set of nodes that appear in the first $t$ elements in the input stream. Processing $t$ elements in the input stream by Algorithms 3 and 4 requires $O(k)$ space for global triangle counting and $O(k+|\mathcal{V}^{(t)}|)$ space for local triangle counting.*

*Proof.* Algorithms 3 and 4 use $O(k)$ space for sampling edges, and they use $O(|\mathcal{V}^{(t)}|)$ space for maintaining the estimates of local triangle counts, which need not be maintained for global triangle counting. ■

## 6 Experiments

We review experiments that we designed to answer the following questions:

- **Q1. Accuracy**: How accurately does WRS estimate global and local triangle counts? Is WRS more accurate than its state-of-the-art competitors?
- **Q2. Trends**: How do true triangle counts, estimates, and estimation errors change over time?
- **Q3. Illustration of Theorems**: Does WRS give unbiased estimates with variances smaller than its competitors'?
- **Q4. Scalability**: How does WRS scale with the number of edges in input streams?
- **Q5. Effects of the Size of the Waiting Room**: How does the relative size $\alpha$ of the waiting room affect the accuracy of WRS? What is the optimal value of $\alpha$?
- **Q6. Effects of Temporal Locality**: How does the degree of temporal locality affect the accuracy of WRS?

### 6.1 Experimental Settings

**Machine:** We ran all experiments on a PC with a 3.60GHz Intel i7-4790 CPU and 32GB memory.
**Data:** The real graph streams used in our experiments are summarized in Table 3 with the following details:

- **ArXiv** [12]: A citation network between papers in ArXiv's High Energy Physics. Each edge $\{u, v\}$ represents that paper $u$ cited paper $v$. We used the submission time of $u$ as the creation time of $\{u, v\}$.
- **Facebook** [48]: A friendship network between users of Facebook. Each edge $\{u, v\}$ represents that user $v$ appeared in the friend list of user $u$. The edges whose creation times are unknown were ignored.
- **Email** [21]: An email network from Enron Corporation. Each edge $\{u, v\}$ represents that employee $u$ sent to or received from person $v$ (who may be a non-employee) at least one email. We used the creation time of the first email between $u$ and $v$ as the creation time of $\{u, v\}$.
- **Youtube** [29]: A friend network between users of Youtube. Each edge $\{u, v\}$ indicates that user $u$ and user $v$ are friends with each other. The edges created before 12/10/2006 were ignored since their exact creation times are unknown.
- **Patent** [14]: A citation network between patents. Each edge $\{u, v\}$ indicates that patent $u$ cited patent $v$. We used the time when $u$ was granted as the creation time of $\{u, v\}$.
- **Forest Fire** (FF) [26]: The forest fire model is a graph generator that reflects several structural properties (heavy-tailed degree distributions, small diameters, communities, etc.) and temporal properties (densification and shrinking diameters, etc.) of

Table 3: Summary of real-world graph streams.

| Name | # Nodes | # Edges | # Triangles | Description |
|---|---|---|---|---|
| ArXiv | 30,565 | 346,849 | 988,009 | Citation |
| Facebook | 61,096 | 614,797 | 1,756,259 | Friendship |
| Email | 86,978 | 297,456 | 1,180,387 | Email |
| Youtube | 3,181,831 | 7,505,218 | 7,766,821 | Friendship |
| Patent | 3,774,768 | 16,518,947 | 7,515,023 | Citation |
| FF-Small | 3,000,000 | 23,345,764 | 94,648,815 | Synthetic |
| FF-Large | 10,000,000 | 87,085,880 | 426,338,480 | Synthetic |
| BA-Small | 3,000,000 | $10^8$ | 1,958,656 | Synthetic |
| BA-Large | 10,000,000 | $10^9$ | 60,117,894 | Synthetic |
| ER | 1,000,000 | $10^{11}$ | $1.333 \times 10^{15}$ | Synthetic |

real-world graphs. Generated graphs grow over time with new nodes and edges.

- **Barabási-Albert** (BA) [5]: The Barabási-Albert model is a graph generator that reflects several structural properties of real-world graphs (heavy-tailed degree distributions, small diameters, giant connected components, etc.). Generated graphs grow over time with new nodes and edges.
- **Erdős-Rényi** (ER) [10]: The Erdős-Rényi model is a graph generator $G(n, p)$ that produces a random graph with $n$ vertices where each possible pair of nodes are connected by an edge independently with probability $p$. Since generated graphs remain static over time, we used them only for test the scalability of WRS in Section 6.5.

The self loops, the duplicated edges, and the directions of the edges were ignored in all the graph streams. In insertion-only streams, edges were streamed in the order by which they are created, as assumed in Section 3.3. For fully dynamic streams, however, since edge deletions are not included in the original datasets, we created edge deletions as follows: (a) choose $\beta$% of the edges and create the deletions of them, (b) locate each deletion in a random position after the corresponding edge addition. Throughout this section, we report experimental results when $\beta$ was set to 20. However, the results were consistent with other $\beta$ values.

**Implementations:** We compared the family of WRS to TRIÈST$_{\text{IMPR}}$ [8], TRIÈST$_{\text{FD}}$ [8], MASCOT [27], THINKD$_{\text{FAST}}$ [38,40], and THINKD$_{\text{ACC}}$ [38,40]. They are all single-pass streaming algorithms that estimate both global and local triangle counts within a limited memory budget, as briefly described in Section 2. We implemented all the methods in Java, and we treated negative estimates as 0.

**Parameter settings:** In WRS, the relative size $\alpha$ of the waiting room was set to 0.1 unless otherwise stated.

**Evaluation measures:** To measure the accuracy of triangle counting, we used the following metrics:

- **Global Error**: Let $x$ be the number of the global triangles at the end of the input stream and $\hat{x}$ be an estimated value of $x$. Then, the global error is defined as $|x - \hat{x}|/(x + 1)$.
- **Local Error** [27]: Let $x_u$ be the number of the local triangles of each node $u \in \mathcal{V}$ at the end of the input stream and $\hat{x}_u$ be its estimate. Then, the local error is defined as

$$\frac{1}{|\mathcal{V}|} \sum\nolimits_{u \in \mathcal{V}} \frac{|x_u - \hat{x}_u|}{x_u + 1}.$$

- **Rank Correlation**: Spearman's rank correlation coefficient [42] measures the similarity of (a) the ranking of the nodes in terms of the true local triangle counts and (b) their ranking in terms of the estimated local triangle counts, at the end of the input stream.

In the following experiments, we computed each evaluation metric 1,000 times for each algorithm, unless otherwise stated, and reported the mean.[2]

### 6.2 Q1. Accuracy

Figures 5 and 6 show the accuracies of the considered methods in the insertion-only and fully-dynamic graph streams, respectively, within different memory budgets. The memory budget $k$ was changed from 50% of the number of elements in the input stream to 0.1% until no algorithm is better than settings all estimates to 0 (i.e., the dashed line) in terms of each evaluation measure.[3, 4]

In all the graph streams, WRS was most accurate in terms of all the evaluation metrics, regardless of the memory budget. The accuracy gaps between WRS and the second best method were especially large in the ArXiv, Patent, FF-Large, and BA-Small datasets, which showed a strong degree of temporal locality (see Section 6.7 for its effects on accuracy). In the ArXiv dataset, for example, WRS$_{\text{INS}}$ gave up to *47% smaller local error* and *40% smaller global error* than the second-best method. For the same dataset, WRS$_{\text{DEL}}$ gave up to *28% smaller local error* and *36% smaller global error* than the second-best approach. WRS was most accurate since its estimates are based on the largest number of discovered triangles, which intuitively mean more information. Specifically, due to its

[2] That is, for each measure, we computed the measure of the estimates on each trial, and then we reported the mean of the computed values. We did not report each measure of the mean of the estimates obtained from all trials.

[3] For MASCOT and THINKD$_{\text{FAST}}$, we set the sampling probability $p$ so that the expected number of sampled edges is equal to the memory budget.

[4] All the considered algorithms were always better than setting all estimates to zero in terms of global error and rank correlation. However, all the considered algorithms were not in terms local error when the memory budget $k$ was extremely small and thus the variances of estimates of local triangle counts were very large.

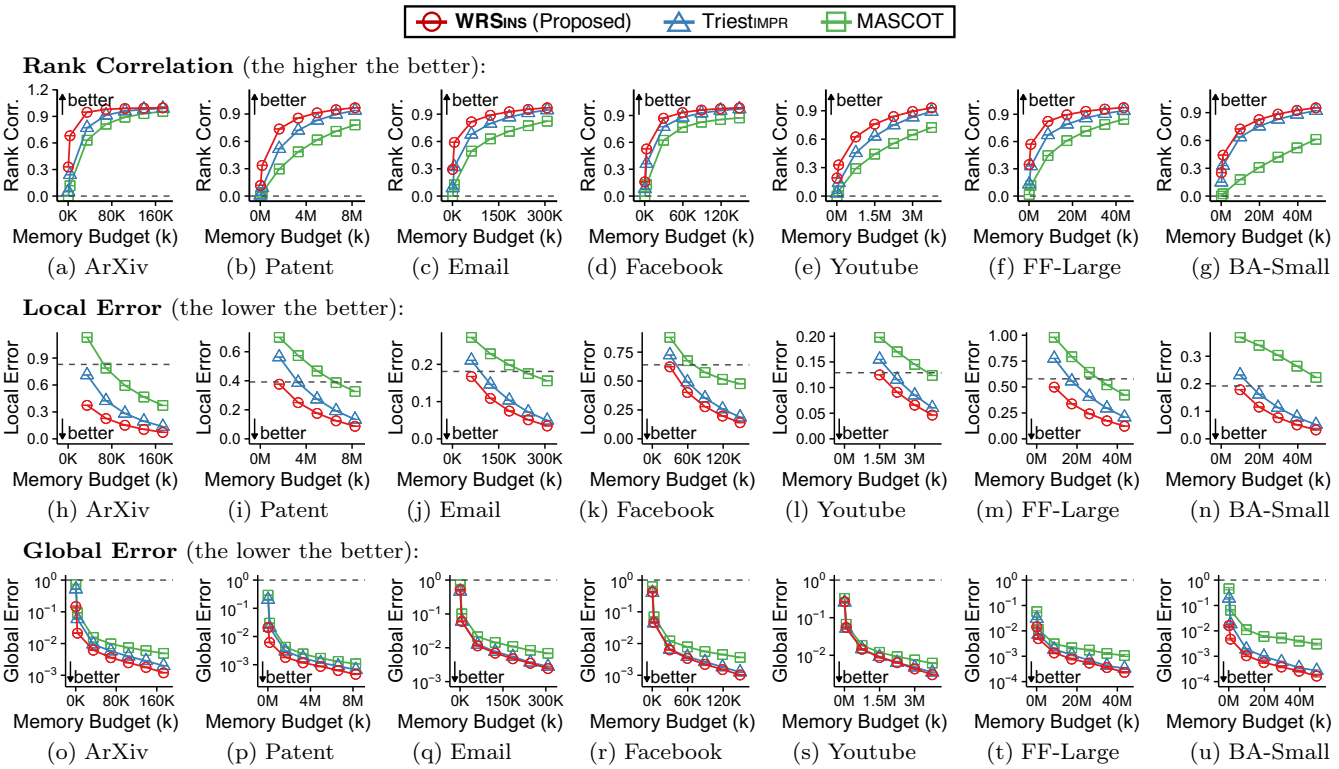



Fig. 5: WRS$_{\text{INS}}$ is accurate. M: million, K: thousand. In all insertion-only graph streams, WRS$_{\text{INS}}$ is most accurate for both global and local triangle counting, regardless of the memory budget $k$. The error bars indicate estimated standard errors.

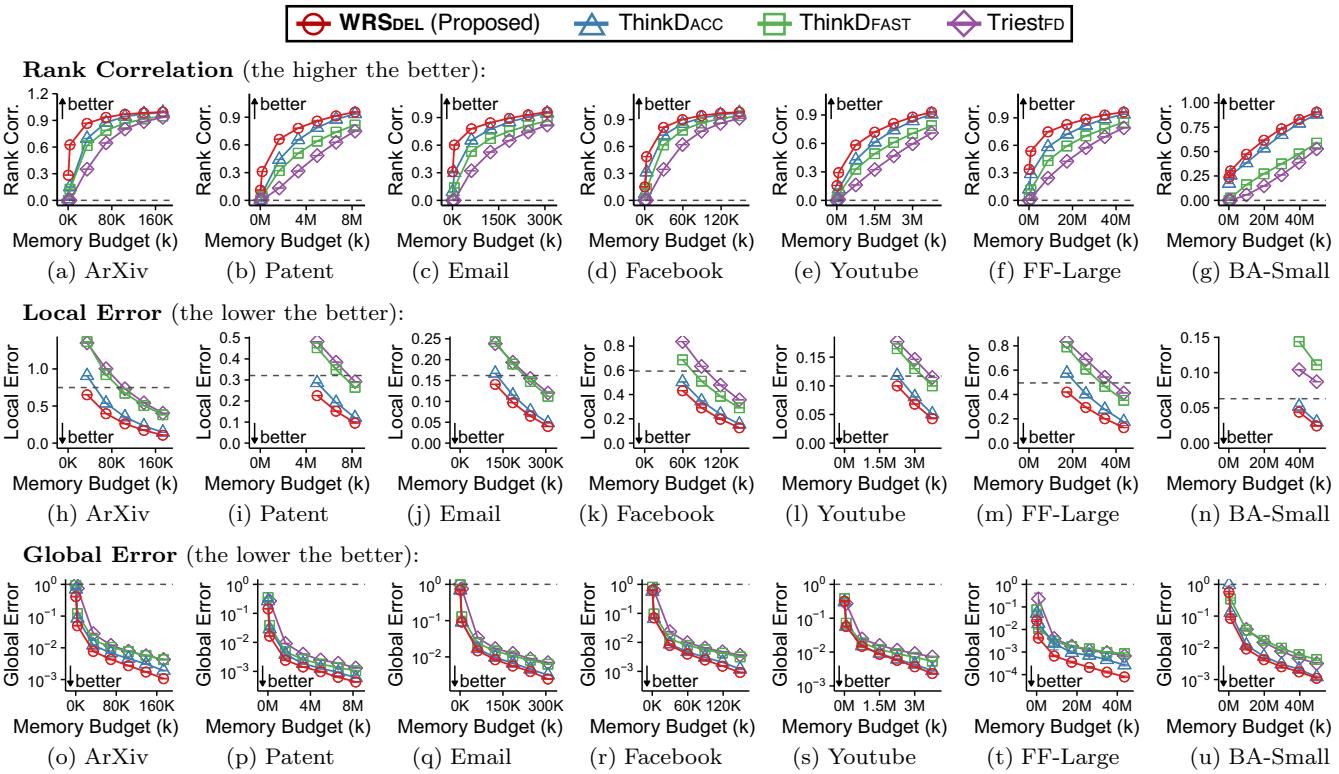
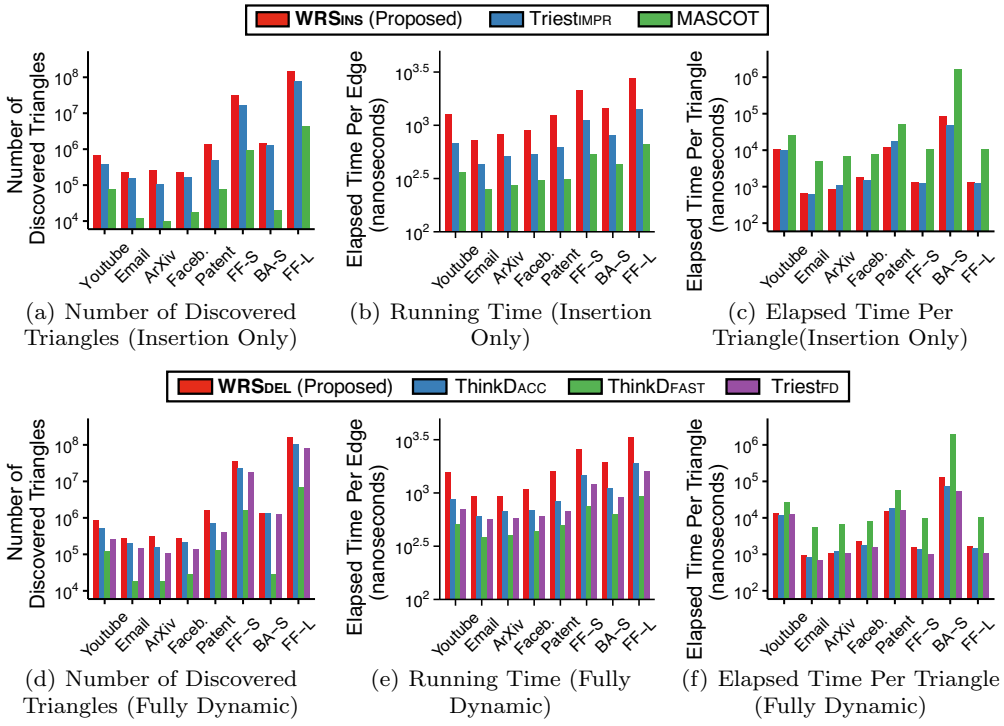



Fig. 6: WRS$_{\text{DEL}}$ is accurate. M: million, K: thousand. In all fully-dynamic graph streams, WRS$_{\text{DEL}}$ is most accurate for both global and local triangle counting, regardless of the memory budget $k$. The error bars indicate estimated standard errors.




(a) Number of Discovered Triangles (Insertion Only) (b) Running Time (Insertion Only) (c) Elapsed Time Per Triangle(Insertion Only)

(d) Number of Discovered Triangles (Fully Dynamic) (e) Running Time (Fully Dynamic) (f) Elapsed Time Per Triangle (Fully Dynamic)

Fig. 7: The sampling scheme of WRS is effective. $k$ is set to 10% of the number of the edges in each dataset, and $\alpha$ is set to 0.1. (a,d) WRS discovers up to 2.9× more triangles than the second best method, in the same streams. (b,e) The running times and the numbers of discovered triangles show similar trends. (c,f) WRS is comparable to its competitors in terms of running time per discovered triangle.

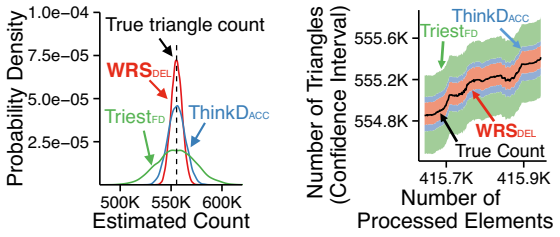
effective sampling scheme, WRS discovered up to 2.9× more triangles than its competitors while processing the same streams, as seen in Figures 7(a) and 7(d).

### 6.3 Q2. Trends

Figure 9 shows how (1) the estimate of the global triangle count, (2) the global error, (3) the local error, and (4) the number of discovered triangles changed over time in each algorithm. We set the memory budget (i.e., $k$) to the 10% of the number of elements in the input stream. The dashed line denotes the end of the stream. WRS was most accurate at all times both in terms of global and local error, and it discovered the largest number of triangles at all times. Since MASCOT and THINKDFAST gradually use more space over time, their global and local errors gradually decreased over time. The other algorithms were exact until the memory is full, and after that, their global and local errors gradually increased over time.

### 6.4 Q3. Illustration of Theorems

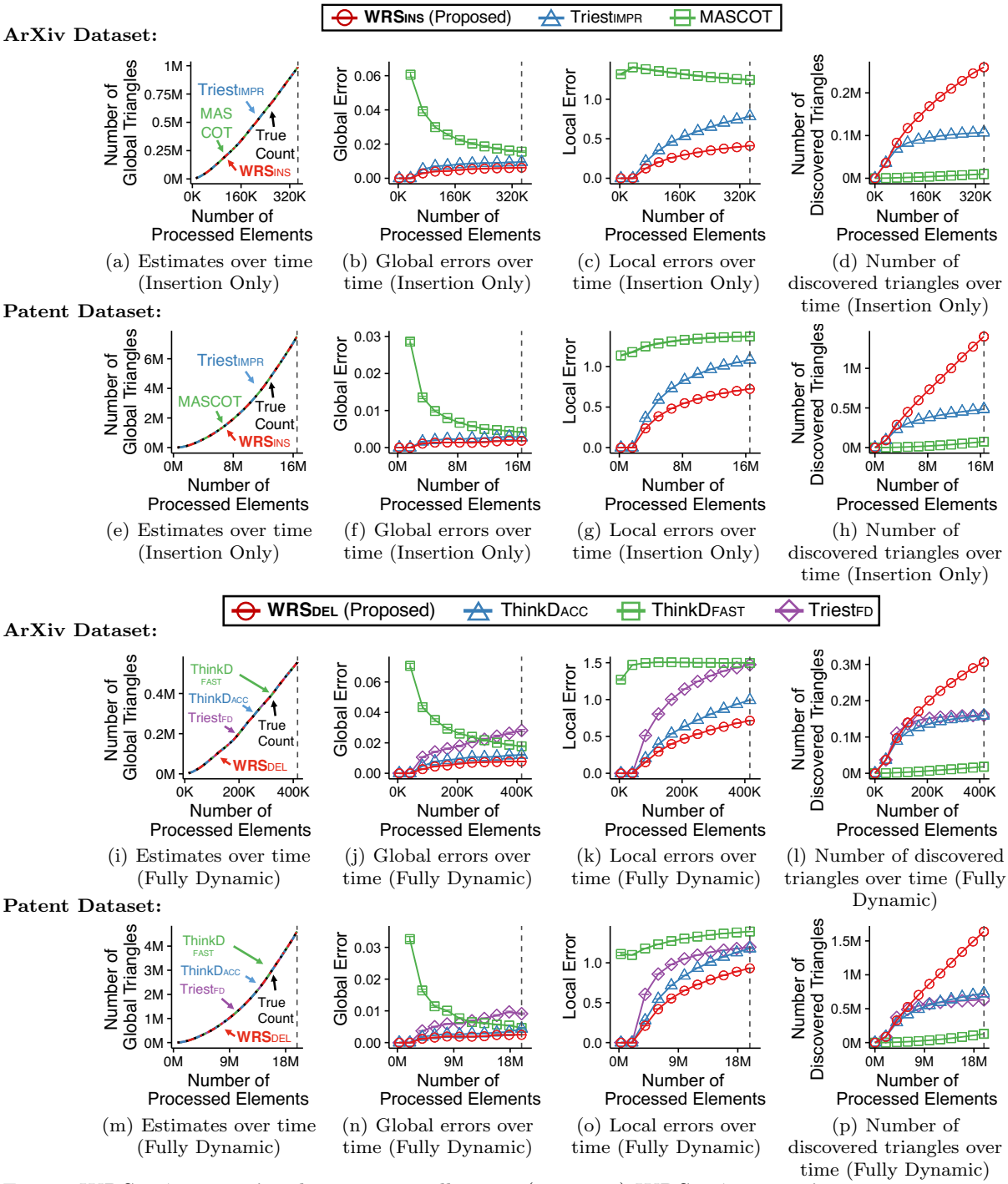
We ran experiments illustrating our analyses in Section 5.4. Figures 1(b), 1(d), and 8 show the distributions




(a) $\text{WRS}_{\text{DEL}}$ is unbiased (b) 95% confidence intervals over time

Fig. 8: $\text{WRS}_{\text{DEL}}$ is unbiased. (a) $\text{WRS}_{\text{DEL}}$ provides unbiased estimates, and their variances are small. (b) $\text{WRS}_{\text{DEL}}$ is the most accurate with the smallest confidence intervals over the entire stream. The ArXiv dataset is used for both (a) and (b).

and the 95% confidence intervals of 10,000 estimates of the global triangle count in the ArXiv dataset obtained by different algorithms. We set the memory budget $k$ to the 10% of the number of elements in the dataset. The average of the estimates of WRS was close to the true triangle count. Moreover, the estimates of WRS had smaller variances and confidence intervals than those of the competitors. These results are consistent with Theorems 1-2 and Lemma 5.

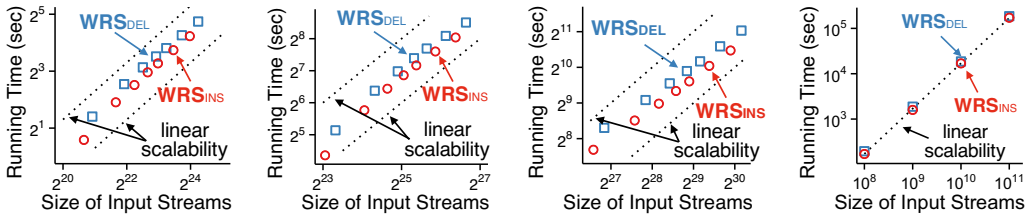



(a) Estimates over time (Insertion Only)
(b) Global errors over time (Insertion Only)
(c) Local errors over time (Insertion Only)
(d) Number of discovered triangles over time (Insertion Only)
(e) Estimates over time (Insertion Only)
(f) Global errors over time (Insertion Only)
(g) Local errors over time (Insertion Only)
(h) Number of discovered triangles over time (Insertion Only)
(i) Estimates over time (Fully Dynamic)
(j) Global errors over time (Fully Dynamic)
(k) Local errors over time (Fully Dynamic)
(l) Number of discovered triangles over time (Fully Dynamic)
(m) Estimates over time (Fully Dynamic)
(n) Global errors over time (Fully Dynamic)
(o) Local errors over time (Fully Dynamic)
(p) Number of discovered triangles over time (Fully Dynamic)

Fig. 9: WRS is 'any time' and accurate at all times. (a, e, i, m) WRS is 'any time', maintaining estimates while the input graph evolves. (b-c, f-g, j-k, n-o) WRS is most accurate at all times. (d, h, l, p) WRS discovers the most triangles at all times.

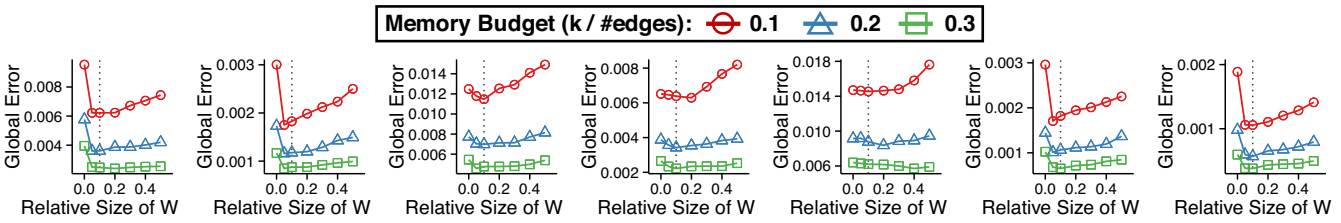
### 6.5 Q4. Scalability

We measured how the running time of WRS scales with the number of edges in the input streams. To measure the scalability independently of the speed of the input stream, we measured the time taken by WRS to process all the elements while ignoring the time taken to wait for the arrivals of elements. Figure 10(a) shows the results in graph streams that we created by sampling different numbers of edges in the Patent dataset. We set $k$ to the 10% of the number of edges in the entire dataset. Each reported value is the average over 1,000 runs. The same experiment was performed with several other datasets, and the results are shown in Figure 10. In all the considered datasets, the running time of WRS scaled linearly with the number of edges. That




(a) Patent (b) FF-Large (c) BA-Large (d) Erdős-Rényi

Fig. 10: WRS is scalable. The running time of WRS is linear in the number of edges in the input graph stream.




(a) ArXiv (b) Patent (c) Email (d) Facebook (e) Youtube (f) FF-Small (g) BA-Small

Fig. 11: Effects of $\alpha$ on the accuracy of WRS. In most cases, the accuracy of WRS is maximized when about 10% of memory space is used for the waiting room (i.e., $\alpha = 0.1$).

is, the time taken by WRS to process each element in the input stream was almost constant regardless of the number of edges that have arrived so far.

Despite its linear scalability, WRS took more time than its competitors to process the entire stream, as seen in Figures 7(b) and 7(e), mainly because it discovered and processed more triangles, as seen in Figures 7(a) and 7(d). In terms of running time per discovered triangle, WRS was comparable to the competitors, as seen in Figures 7(c) and 7(f).

### 6.6 Q5. Effects of the Size of the Waiting Room

We measured how the accuracy of WRS depends on $\alpha$, the relative size of the waiting room. Figure 11 shows the results with different memory budgets. Here, we used global error as the accuracy metric, and the average values over $1,000$ runs were reported. In all the datasets, using proper amount of memory space for the waiting room gave better accuracy than using no space for the waiting room ($\alpha = 0$) and using half the space for the waiting room ($\alpha = 0.5$), regardless of memory budgets. Although proper $\alpha$ values depended on datasets and memory budgets, the accuracy was maximized when $\alpha$ was about 0.1 in most of the cases.

### 6.7 Q6. Effects of Temporal Locality

We investigated how the degree of temporal locality affects the effectiveness of WRS. To this end, we quantified the degree of temporal locality as the fraction of the number of Type 1 or 2 triangles to the number of all discovered triangles after processing the graph stream until the given memory space is full (i.e., after processing the first $k$ non-deleted edges in the input graph stream). Table 4 shows the number of discovered triangles of each type and their proportions when we used $\text{WRS}_{\text{INS}}$. We set the memory budget $k$ to 10% of the number of elements in each dataset, and we set $\alpha$ to 0.1, following the result of Section 6.6. In the ArXiv, Patent, FF-Large, and BA-Small datasets, more than 99.9% of the discovered triangles are of Type 1 or Type 2, whose closing intervals are smaller than the size of the waiting room. In these datasets, which show strong temporal locality, $\text{WRS}_{\text{INS}}$ was especially more accurate than its competitors, as seen in Figures 5(h), 5(i), 5(m), 5(n), 5(o), 5(p), 5(t), and 5(u). The other datasets (i.e., the Email, Facebook, and Youtube datasets) show weaker temporal locality with smaller fractions of Type 1 or 2 triangles. In them, the accuracy gaps between $\text{WRS}_{\text{INS}}$ and its competitors were relatively small, as seen in Figures 5(j), 5(k), 5(l), 5(q), 5(r), and 5(s).

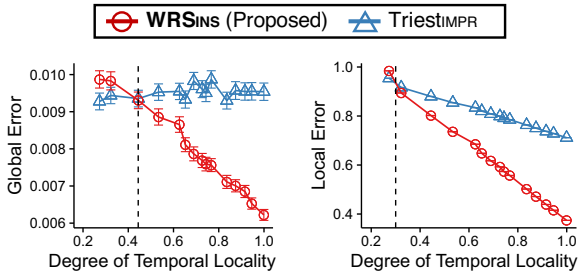
Figure 12 shows how the global and local errors of $\text{WRS}_{\text{INS}}$ and its best competitor (i.e., $\text{TRIÈST}_{\text{IMPR}}$) changed according to the degree of temporal locality. To obtain graph streams with different degrees of temporal locality, we chose different fractions (from 10% to 50%) of the edges in the ArXiv dataset and swapped them with randomly chosen edges.[5] $\text{WRS}_{\text{INS}}$ was significantly more accurate than $\text{TRIÈST}_{\text{IMPR}}$ when the input graph stream exhibited strong temporal locality. As the temporal locality became weaker, the accuracy gap de-

[5] Specifically, we ran the Knuth shuffle [22] while skipping different fractions of iterations.

Table 4: The degree of temporal locality in the input stream affects the accruacy of WRS. The table below reports the number of triangles of each type and their proportions after processing each graph stream until the given memory space is full. The ArXiv, Patent, FF-Large, and BA-Small datasets, where WRS was especially more accurate than its competitors, show strong temporal locality with large fractions of Type 1 or 2 triangles. The other datasets show weaker temporal locality with smaller fractions of Type 1 or 2 triangles.

| Dataset | $\#\Delta$ | #T1 (#T1/$\#\Delta$) | #T2 (#T2/$\#\Delta$) | #T3 (#T3/$\#\Delta$) | (#T1 + #T2)/$\#\Delta$ |
|---|---|---|---|---|---|
| **ArXiv** | 35,937 | 6,754 (**0.1879**) | 29,161 (**0.8114**) | 22 (0.0006) | **0.9994** |
| **Patent** | 92,511 | 345 (**0.0037**) | 92,166 (**0.9963**) | 0 (0) | **1.0000** |
| **Email** | 57,319 | 16,363 (0.2855) | 15,477 (0.2700) | 25,479 (0.4445) | 0.5555 |
| **Facebook** | 48,831 | 8,918 (0.1826) | 22,004 (0.4506) | 17,909 (0.3668) | 0.6332 |
| **Youtube** | 112,386 | 26,954 (0.2398) | 42,466 (0.3779) | 42,966 (0.3823) | 0.6177 |
| **FF-Large** | 38,135,620 | 13,397,029 (**0.3513**) | 24,738,591 (**0.6487**) | 0 (0) | **1.0000** |
| **BA-Small** | 1,131,350 | 645,321 (**0.5704**) | 486,029 (**0.4296**) | 0 (0) | **1.0000** |

$\Delta$: all types of triangles when the given memory space is full.
T1: Type 1 triangles among $\Delta$, T2: Type 2 triangles among $\Delta$, T3: Type 3 triangles among $\Delta$.




(a) Global Error (Insertion Only) (b) Local Error (Insertion Only)

Fig. 12: Effects of temporal locality on the accuracy of $\text{WRS}_{\text{INS}}$. $\text{WRS}_{\text{INS}}$ is significantly more accurate than its best competitor when the input graph stream exhibits strong temporal locality.

creased, and eventually $\text{TRIÈST}_{\text{IMPR}}$ became more accurate than $\text{WRS}_{\text{INS}}$.

As shown in the results, the degree of temporal locality in the input stream affects the accuracy of WRS. Thus, if the degree of temporal locality in the input stream is not known in advance, we can first measure the fraction of Type 1 or 2 triangles until the given memory space is full, as in Table 4 and Figure 12, and use WRS only if the fraction is high enough (e.g., the fraction is greater than 0.5). If the fraction is low, we can continue without the waiting room, which is equivalent to $\text{TRIÈST}_{\text{IMPR}}$, or use any other method.

## 7 Conclusion

In this work, we propose WRS, a family of single-pass streaming algorithms for global and local triangle counting in insertion-only and fully dynamic graph streams. WRS divides the memory space into the waiting room, where the latest edges are stored, and the reservoir, where the remaining edges are uniformly sampled. By doing so, WRS exploits the temporal locality in real graph streams for reducing variances, while giving unbiased estimates. To sum up, WRS has the following strengths:

- **Fast and 'any time'**: WRS scales linearly with the number of edges in the input graph stream, and it maintains estimates at any time while the input graph evolves over time (Figures 1 and 8(b)).
- **Effective**: Estimation error in WRS is up to 47% *smaller* than that in its best competitors (Figures 5 and 6).
- **Theoretically sound**: WRS gives unbiased estimates with small variances under the temporal locality (Theorems 1, 2; Lemma 5; and Figures 1, 8).

**Reproducibility**: The code and datasets used in the paper are available at `http://dmlab.kaist.ac.kr/wrs/`.

**Acknowledgements** This research was supported by Disaster-Safety Platform Technology Development Program of the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (Grant Number: 2019M3D7A1094364) and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-00075, Artificial Intelligence Graduate School Program (KAIST)). This research was also supported by the National Science Foundation under Grant No. CNS-1314632 and IIS-1408924. This research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

## A Appendix: Proof of Lemma 2

We provide a proof of Lemma 2, which is based on several properties of RP. We first introduce the uniformity of RP [38] in Lemma 6. Then, we present the mean and variance of the size of the reservoir $\mathcal{R}$ [13] in Lemma 7. Throughout this section, we use the superscript $(t)$ to indicate the value of each variable after the $t$-th element $\Delta^{(t)}$ is processed. We let $\mathcal{E}_{\mathcal{R}}^{(t)}$ be the set of edges flowing to the reservoir $\mathcal{R}$ from

the waiting room $\mathcal{W}$ at time $t$ or earlier in Algorithm 4. We also let $y_{\mathcal{R}}^{(t)} = \min(k(1-\alpha), |\mathcal{E}_{\mathcal{R}}^{(t)}| + n_b^{(t)} + n_g^{(t)})$.

**Lemma 6 (Uniformity of Random Pairing [38])** *At any time $t$, all equally sized subsets of the $\mathcal{E}_{\mathcal{R}}^{(t)}$ are equally likely to be a subset of the reservoir $\mathcal{R}^{(t)}$. Formally,*

$$\mathbb{P}[\mathcal{A} \subseteq \mathcal{R}^{(t)}] = \mathbb{P}[\mathcal{B} \subseteq \mathcal{R}^{(t)}], \quad \forall t \geq 1,\ \forall A \neq B \subset \mathcal{E}_{\mathcal{R}}^{(t)},\ s.t.\ |\mathcal{A}| = |\mathcal{B}|. \tag{23}$$

**Lemma 7 (Expectation, Variance of the Reservoir Size of Random Pairing [13])** *The expected value and the variance of the size of the reservoir $\mathcal{R}$ at any time $t$ in Algorithm 4 are formulated as follows:*

$$\mathbb{E}[|\mathcal{R}^{(t)}|] = \frac{|\mathcal{E}_{\mathcal{R}}^{(t)}|}{|\mathcal{E}_{\mathcal{R}}^{(t)}| + d^{(t)}} \cdot y_{\mathcal{R}}^{(t)}, \tag{24}$$

$$\mathrm{Var}[|\mathcal{R}^{(t)}|] = \frac{d^{(t)} \cdot y_{\mathcal{R}}^{(t)} \cdot (|\mathcal{E}_{\mathcal{R}}^{(t)}| + d^{(t)} - y_{\mathcal{R}}^{(t)}) \cdot |\mathcal{E}_{\mathcal{R}}^{(t)}|}{(|\mathcal{E}_{\mathcal{R}}^{(t)}| + d^{(t)})^2 \cdot (|\mathcal{E}_{\mathcal{R}}^{(t)}| + d^{(t)} - 1)}, \tag{25}$$

*where $d^{(t)} = n_b^{(t)} + n_g^{(t)}$.*

In Lemma 8, we formulate the probability that each edge in $\mathcal{E}_{\mathcal{R}}$ is stored in the reservoir $\mathcal{R}$ in $\mathrm{WRS}_{\mathrm{DEL}}$.

**Lemma 8 (Sampling Probability of Each Edge in Random Pairing)** *At any time $t$, the probability that each edge in $\mathcal{E}_{\mathcal{R}}$ is stored in $\mathcal{R}$ after the $t$-th element is processed in Algorithm 4 is*

$$\mathbb{P}[\{u,v\} \in \mathcal{R}^{(t)}] = \frac{y_{\mathcal{R}}^{(t)}}{|\mathcal{E}_{\mathcal{R}}^{(t)}| + n_b^{(t)} + n_g^{(t)}}. \tag{26}$$

*Proof.* Let $\mathbb{1}(\{u,v\} \in \mathcal{R}^{(t)})$ be a random variable that becomes 1 if $\{u,v\} \in \mathcal{R}^{(t)}$ and 0 otherwise. By definition,

$$|\mathcal{R}^{(t)}| = \sum_{\{u,v\} \in \mathcal{E}_{\mathcal{R}}^{(t)}} \mathbb{1}(\{u,v\} \in \mathcal{R}^{(t)}). \tag{27}$$

Then, by linearity of expectation and Eq. (27),

$$\begin{aligned}\mathbb{E}[|\mathcal{R}^{(t)}|] &= \sum_{\{u,v\} \in \mathcal{E}_{\mathcal{R}}^{(t)}} \mathbb{E}[\mathbb{1}(\{u,v\} \in \mathcal{R}^{(t)})] \\ &= \sum_{\{u,v\} \in \mathcal{E}_{\mathcal{R}}^{(t)}} \mathbb{P}[\{u,v\} \in \mathcal{R}^{(t)}].\end{aligned} \tag{28}$$

Then, Eq. (26) is obtained as follows:

$$\begin{aligned}\mathbb{P}[\{u,v\} \in \mathcal{R}^{(t)}] &= \frac{1}{|\mathcal{E}_{\mathcal{R}}^{(t)}|} \sum_{\{w,x\} \in \mathcal{E}_{\mathcal{R}}^{(t)}} \mathbb{P}[\{w,x\} \in \mathcal{R}^{(t)}] \\ &= \frac{\mathbb{E}[|\mathcal{R}^{(t)}|]}{|\mathcal{E}_{\mathcal{R}}^{(t)}|} = \frac{y_{\mathcal{R}}^{(t)}}{|\mathcal{E}_{\mathcal{R}}^{(t)}| + n_b^{(t)} + n_g^{(t)}},\end{aligned}$$

where the first, second, and last equalities are from Eq. (23), Eq. (28), and Eq. (24), respectively. ∎

**Proof of Lemma 2:** We prove Lemma 2 based on Lemma 8.

*Proof.* Without loss of generality, we assume $e_{uvw}^{(1)} = \{v,w\}$, $e_{uvw}^{(2)} = \{w,u\}$, and $e_{uvw}^{(3)} = e^{(t)} = \{u,v\}$. That is, $\{v,w\}$ arrives earlier than $\{w,u\}$, and $\{w,u\}$ arrives earlier than $\{u,v\}$. When $e^{(t)} = \{u,v\}$ arrives at time $t$, the triangle $\{u,v,w\}$ is discovered if and only if both $\{v,w\}$ and $\{w,u\}$ are in $\mathcal{S}^{(t-1)}$.

Note that, since $\mathrm{WRS}_{\mathrm{DEL}}$ updates the triangle counts before sampling or deleting edges, $\mathcal{E}_{\mathcal{R}} = \mathcal{E}_{\mathcal{R}}^{(t-1)}$, $n_b = n_b^{(t-1)}$ and $n_g = n_g^{(t-1)}$ when executing lines 6 and 8 of Algorithm 4.

If $type_{uvw} = 1$, $\{v,w\}$ and $\{w,u\}$ are always stored in $\mathcal{W}^{(t-1)}$, when $\{u,v\}$ arrives. Thus, $\mathrm{WRS}_{\mathrm{DEL}}$ discovers $\{u,v,w\}$ with probability 1.

If $type_{uvw} = 2$, when $\{u,v\}$ arrives, $\{w,u\}$ is always stored in $\mathcal{W}^{(t-1)}$, while $\{v,w\}$ cannot be in $\mathcal{W}^{(t-1)}$ but can be in $\mathcal{R}^{(t-1)}$. For $\mathrm{WRS}_{\mathrm{DEL}}$ to discover $\{u,v,w\}$, $\{v,w\}$ should be in $\mathcal{R}^{(t-1)}$, and from Eq. (26), the probability of the event is

$$\mathbb{P}[\{v,w\} \in \mathcal{R}^{(t-1)}] = \frac{y_{\mathcal{R}}^{(t-1)}}{|\mathcal{E}_{\mathcal{R}}^{(t-1)}| + d^{(t-1)}},$$

where $d^{(t-1)} = n_b^{(t-1)} + n_g^{(t-1)}$.

If $type_{uvw} = 3$, $\{v,w\}$ and $\{w,u\}$ cannot be in $\mathcal{W}^{(t-1)}$, when $\{u,v\}$ arrives. For $\mathrm{WRS}_{\mathrm{DEL}}$ to discover $\{u,v,w\}$, both $\{v,w\}$ and $\{w,u\}$ should be in $\mathcal{R}^{(t-1)}$. The probability of the event is $\mathbb{P}[\{v,w\} \in \mathcal{R}^{(t-1)}$ and $\{w,u\} \in \mathcal{R}^{(t-1)}]$.

Below, we formulate this joint probability by expanding the covariance sum $\sum_{\{v,w\} \neq \{w,u\}} Cov\big(\mathbb{1}(\{v,w\} \in \mathcal{R}^{(t-1)}), \mathbb{1}(\{w,u\} \in \mathcal{R}^{(t-1)})\big)$ in two ways and compare them. Each random variable $\mathbb{1}(\{u,v\} \in \mathcal{R}^{(t-1)})$ is 1 if $\{u,v\} \in \mathcal{R}^{(t-1)}$ and 0 otherwise.

First, we expand the variance of $\mathcal{R}^{(t-1)}$. From Eq. (27),

$$\begin{aligned}Var\big[|\mathcal{R}^{(t-1)}|\big] &= \sum_{\{u,v\} \in \mathcal{E}_{\mathcal{R}}^{(t-1)}} Var\big[\mathbb{1}(\{u,v\} \in \mathcal{R}^{(t-1)})\big] \\ &+ \sum_{\{u,v\} \neq \{x,y\}} Cov\left(\mathbb{1}(\{u,v\} \in \mathcal{R}^{(t-1)}), \mathbb{1}(\{x,y\} \in \mathcal{R}^{(t-1)})\right),\end{aligned}$$

and hence the covariance sum is be expanded as

$$\begin{aligned}&\sum_{\{u,v\} \neq \{x,y\}} Cov\left(\mathbb{1}(\{u,v\} \in \mathcal{R}^{(t-1)}), \mathbb{1}(\{x,y\} \in \mathcal{R}^{(t-1)})\right) \\ &= Var\big[|\mathcal{R}^{(t-1)}|\big] - \sum_{\{u,v\} \in \mathcal{E}_{\mathcal{R}}^{(t-1)}} Var\big[\mathbb{1}(\{u,v\} \in \mathcal{R}^{(t-1)})\big].\end{aligned} \tag{29}$$

From $Var[x] = \mathbb{E}[x^2] - (\mathbb{E}[x])^2$, we have

$$\begin{aligned}&Var\big[\mathbb{1}(\{u,v\} \in \mathcal{R}^{(t-1)})\big] = \\ &\qquad \mathbb{P}\big[\{u,v\} \in \mathcal{R}^{(t-1)}\big] - \mathbb{P}\big[\{u,v\} \in \mathcal{R}^{(t-1)}\big]^2.\end{aligned} \tag{30}$$

Applying Eq. (26) and Eq. (30) to Eq. (29) results in

$$\begin{aligned}&\sum_{\{u,v\} \neq \{x,y\}} Cov\left(\mathbb{1}(\{u,v\} \in \mathcal{R}^{(t-1)}), \mathbb{1}(\{x,y\} \in \mathcal{R}^{(t-1)})\right) \\ &= Var\big[|\mathcal{R}^{(t-1)}|\big] - \sum_{\{u,v\} \in \mathcal{E}_{\mathcal{R}}^{(t-1)}} Var\big[\mathbb{1}(\{u,v\} \in \mathcal{R}^{(t-1)})\big] \\ &= Var\big[|\mathcal{R}^{(t-1)}|\big] \\ &\quad - |\mathcal{E}_{\mathcal{R}}^{(t-1)}| \cdot \frac{y_{\mathcal{R}}^{(t-1)} \cdot (|\mathcal{E}_{\mathcal{R}}^{(t-1)}| + d^{(t-1)} - y_{\mathcal{R}}^{(t-1)})}{(|\mathcal{E}_{\mathcal{R}}^{(t-1)}| + d^{(t-1)})^2}.\end{aligned} \tag{31}$$

Then, we directly expand the covariance sum. With $Cov(x,y) = \mathbb{E}[xy] - \mathbb{E}[x] \cdot \mathbb{E}[y]$ and Eq. (26), the covariance sum is expanded as

$$\begin{aligned}
&\sum_{\{u,v\}\neq\{x,y\}} Cov\left(\mathbb{1}(\{u,v\} \in \mathcal{R}^{(t-1)}), \mathbb{1}(\{x,y\} \in \mathcal{R}^{(t-1)})\right) \\
&= \sum_{\{u,v\}\neq\{x,y\}} \Big(\mathbb{P}\big[\{u,v\} \in \mathcal{R}^{(t-1)} \cap \{x,y\} \in \mathcal{R}^{(t-1)}\big] \\
&\quad - \mathbb{P}\big[\{u,v\} \in \mathcal{R}^{(t-1)}\big] \cdot \mathbb{P}\big[\{x,y\} \in \mathcal{R}^{(t-1)}\big]\Big) \\
&= \sum_{\{u,v\}\neq\{x,y\}} \mathbb{P}\big[\{u,v\} \in \mathcal{R}^{(t-1)} \cap \{x,y\} \in \mathcal{R}^{(t-1)}\big] \\
&\quad - \frac{y_{\mathcal{R}}^{(t-1)} \cdot y_{\mathcal{R}}^{(t-1)} \cdot |\mathcal{E}_{\mathcal{R}}^{(t-1)}| \cdot (|\mathcal{E}_{\mathcal{R}}^{(t-1)}| - 1)}{(|\mathcal{E}_{\mathcal{R}}^{(t-1)}| + d^{(t-1)})^2}. \qquad (32)
\end{aligned}$$

Next, we obtain the sum of joint probabilities by comparing the two expansions (i.e., Eq. (31) and Eq. (32)) as follows:

$$\begin{aligned}
&\sum_{\{u,v\}\neq\{x,y\}} \mathbb{P}\big[\{u,v\} \in \mathcal{R}^{(t-1)} \cap \{x,y\} \in \mathcal{R}^{(t-1)}\big] \\
&= \sum_{\{u,v\}\neq\{x,y\}} Cov\left(\mathbb{1}(\{u,v\} \in \mathcal{R}^{(t-1)}), \mathbb{1}(\{x,y\} \in \mathcal{R}^{(t-1)})\right) \\
&\quad + \frac{y_{\mathcal{R}}^{(t-1)} \cdot y_{\mathcal{R}}^{(t-1)} \cdot |\mathcal{E}_{\mathcal{R}}^{(t-1)}| \cdot (|\mathcal{E}_{\mathcal{R}}^{(t-1)}| - 1)}{(|\mathcal{E}_{\mathcal{R}}^{(t-1)}| + d^{(t-1)})^2} \\
&= Var\big[|\mathcal{R}^{(t-1)}|\big] \\
&\quad - |\mathcal{E}_{\mathcal{R}}^{(t-1)}| \cdot \frac{y_{\mathcal{R}}^{(t-1)} \cdot (|\mathcal{E}_{\mathcal{R}}^{(t-1)}| + d^{(t-1)} - y_{\mathcal{R}}^{(t-1)})}{(|\mathcal{E}_{\mathcal{R}}^{(t-1)}| + d^{(t-1)})^2} \\
&\quad + \frac{y_{\mathcal{R}}^{(t-1)} \cdot y_{\mathcal{R}}^{(t-1)} \cdot |\mathcal{E}_{\mathcal{R}}^{(t-1)}| \cdot (|\mathcal{E}_{\mathcal{R}}^{(t-1)}| - 1)}{(|\mathcal{E}_{\mathcal{R}}^{(t-1)}| + d^{(t-1)})^2} \\
&= \frac{y_{\mathcal{R}}^{(t-1)} \cdot (y_{\mathcal{R}}^{(t-1)} - 1) \cdot |\mathcal{E}_{\mathcal{R}}^{(t-1)}| \cdot (|\mathcal{E}_{\mathcal{R}}^{(t-1)}| - 1)}{(|\mathcal{E}_{\mathcal{R}}^{(t-1)}| + d^{(t-1)}) \cdot (|\mathcal{E}_{\mathcal{R}}^{(t-1)}| + d^{(t-1)} - 1)}. \qquad (33)
\end{aligned}$$

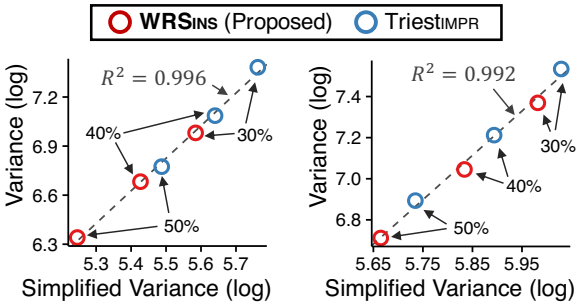
Lastly, each joint probability $\mathbb{P}\big[\{u,v\} \in \mathcal{R}^{(t-1)} \cap \{x,y\} \in \mathcal{R}^{(t-1)}\big]$ is implied from Eq. (23) and Eq. (33) as follows:

$$\begin{aligned}
&\mathbb{P}\big[\{u,v\} \in \mathcal{R}^{(t-1)} \cap \{x,y\} \in \mathcal{R}^{(t-1)}\big] \\
&= \frac{\sum_{\{u,v\}\neq\{x,y\}} \mathbb{P}\big[\{u,v\} \in \mathcal{R}^{(t-1)} \cap \{x,y\} \in \mathcal{R}^{(t-1)}\big]}{|\mathcal{E}_{\mathcal{R}}^{(t-1)}| \cdot (|\mathcal{E}_{\mathcal{R}}^{(t-1)}| - 1)} \\
&= \frac{y_{\mathcal{R}}^{(t-1)}}{|\mathcal{E}_{\mathcal{R}}^{(t-1)}| + d^{(t-1)}} \times \frac{y_{\mathcal{R}}^{(t-1)} - 1}{|\mathcal{E}_{\mathcal{R}}^{(t-1)}| + d^{(t-1)} - 1},
\end{aligned}$$

which proves Eq. (4) in Lemma 2 when $type_{uvw} = 3$. ■

## B Appendix: Variance Analysis

We measured the variance $\mathrm{Var}[c^{(t)}]$ of the estimate of the global triangle count and the simplified version $\widetilde{\mathrm{Var}}[c^{(t)}]$ (i.e., Eq. (16)) while changing the memory budget $k$ from 30% to 50% of the number of elements in the input stream. As seen in Figure 13, while there is a clear difference between the true and simplified variances, the two values are strongly correlated ($R^2 > 0.99$) on a log-log scale both in the Arxiv and Facebook datasets. This strong correlation supports the validity of our simple and illuminating analysis in Section 5.4.2, where we compare the simplified variances of $\mathrm{WRS}_{\mathrm{INS}}$ and $\mathrm{TRIÈST}_{\mathrm{IMPR}}$, instead of their true variances, to provide an intuition why $\mathrm{WRS}_{\mathrm{INS}}$ is more accurate than $\mathrm{TRIÈST}_{\mathrm{IMPR}}$.




(a) ArXiv (b) Facebook

Fig. 13: The true and simplified variances are strongly correlated ($R^2 > 0.99$) on a log-log scale. Both true and simplified variances of $\mathrm{WRS}_{\mathrm{INS}}$ are smaller than those of $\mathrm{TRIÈST}_{\mathrm{IMPR}}$ within the same memory budget.

## References


1. Ahmed, N.K., Duffield, N., Neville, J., Kompella, R.: Graph sample and hold: A framework for big-graph analytics. In: KDD, pp. 1446–1455 (2014)
2. Ahmed, N.K., Duffield, N., Willke, T.L., Rossi, R.A.: On sampling from massive graph streams. PVLDB **10**(11), 1430–1441 (2017)
3. Arifuzzaman, S., Khan, M., Marathe, M.: Patric: A parallel algorithm for counting triangles in massive networks. In: CIKM, pp. 529–538 (2013)
4. Bar-Yossef, Z., Kumar, R., Sivakumar, D.: Reductions in streaming algorithms, with an application to counting triangles in graphs. In: SODA, pp. 623–632 (2002)
5. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science **286**(5439), 509–512 (1999)
6. Becchetti, L., Boldi, P., Castillo, C., Gionis, A.: Efficient algorithms for large-scale local triangle counting. ACM Transactions on Knowledge Discovery from Data **4**(3), 13 (2010)
7. Brown, P.G., Haas, P.J.: Techniques for warehousing of sample data. In: ICDE, pp. 6–6 (2006)
8. De Stefani, L., Epasto, A., Riondato, M., Upfal, E.: Trièst: Counting local and global triangles in fully dynamic streams with fixed memory size. ACM Transactions on Knowledge Discovery from Data **11**(4), 43 (2017)
9. Eckmann, J.P., Moses, E.: Curvature of co-links uncovers hidden thematic layers in the world wide web. PNAS **99**(9), 5825–5829 (2002)
10. Erdös, P.: On the structure of linear graphs. Israel J. of Math. **1**(3), 156–160 (1963)
11. Etemadi, R., Lu, J., Tsin, Y.H.: Efficient estimation of triangles in very large graphs. In: CIKM, pp. 1251–1260 (2016)
12. Gehrke, J., Ginsparg, P., Kleinberg, J.: Overview of the 2003 kdd cup. ACM SIGKDD Explorations Newsletter **5**(2), 149–151 (2003)
13. Gemulla, R., Lehner, W., Haas, P.J.: Maintaining bounded-size sample synopses of evolving datasets. The VLDB Journal **17**(2), 173–201 (2008)
14. Hall, B.H., Jaffe, A.B., Trajtenberg, M.: The nber patent citation data file: Lessons, insights and methodological tools. Tech. rep., National Bureau of Economic Research (2001)
15. Han, G., Sethu, H.: Edge sample and discard: A new algorithm for counting triangles in large dynamic graphs. In: ASONAM, pp. 44–49 (2017)

16. Hu, X., Tao, Y., Chung, C.W.: I/O-efficient algorithms on triangle listing and counting. ACM Transactions on Database Systems **39**(4), 27 (2014)
17. Jha, M., Seshadhri, C., Pinar, A.: A space efficient streaming algorithm for triangle counting using the birthday paradox. In: KDD, pp. 589–597 (2013)
18. Jung, M., Lim, Y., Lee, S., Kang, U.: Furl: Fixed-memory and uncertainty reducing local triangle counting for multigraph streams. Data Mining and Knowledge Discovery **33**(5), 1225–1253 (2019)
19. Kallaugher, J., Price, E.: A hybrid sampling scheme for triangle counting. In: SODA, pp. 1778–1797 (2017)
20. Kim, J., Han, W.S., Lee, S., Park, K., Yu, H.: Opt: A new framework for overlapped and parallel triangulation in large-scale graphs. In: SIGMOD, pp. 637–648 (2014)
21. Klimt, B., Yang, Y.: Introducing the enron corpus. In: CEAS (2004)
22. Knuth, D.E.: Seminumerical algorithms. The art of computer programming **2**, 139–140 (1997)
23. Kolountzakis, M.N., Miller, G.L., Peng, R., Tsourakakis, C.E.: Efficient triangle counting in large graphs via degree-based vertex partitioning. In: WAW, pp. 15–24 (2010)
24. Kutzkov, K., Pagh, R.: On the streaming complexity of computing local clustering coefficients. In: WSDM, pp. 677–686 (2013)
25. Kutzkov, K., Pagh, R.: Triangle counting in dynamic graph streams. In: SWAT, pp. 306–318 (2014)
26. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: Densification and shrinking diameters. ACM Transactions on Knowledge Discovery from Data **1**(1), 2 (2007)
27. Lim, Y., Kang, U.: Mascot: Memory-efficient and accurate sampling for counting local triangles in graph streams. In: KDD, pp. 685–694 (2015)
28. McPherson, M., Smith-Lovin, L., Cook, J.M.: Birds of a feather: Homophily in social networks. Annual review of sociology **27**(1), 415–444 (2001)
29. Mislove, A.: Online social networks: Measurement, analysis, and applications to distributed information systems. Ph.D. thesis, Rice University (2009)
30. Newman, M.E.: The structure and function of complex networks. SIAM review **45**(2), 167–256 (2003)
31. Park, H.M., Myaeng, S.H., Kang, U.: Pte: Enumerating trillion triangles on distributed systems. In: KDD, pp. 1115–1124 (2016)
32. Park, H.M., Silvestri, F., Kang, U., Pagh, R.: Mapreduce triangle enumeration with guarantees. In: CIKM, pp. 1739–1748 (2014)
33. Pavan, A., Tangwongsan, K., Tirthapura, S., Wu, K.L.: Counting and sampling triangles from a graph stream. PVLDB **6**(14), 1870–1881 (2013)
34. Seshadhri, C., Pinar, A., Kolda, T.G.: Triadic measures on graphs: The power of wedge sampling. In: SDM, pp. 10–18 (2013)
35. Shin, K.: Wrs: Waiting room sampling for accurate triangle counting in real graph streams. In: ICDM, pp. 1087–1092 (2017)
36. Shin, K., Eliassi-Rad, T., Faloutsos, C.: Patterns and anomalies in k-cores of real-world graphs with applications. Knowledge and Information Systems **54**(3), 677–710 (2018)
37. Shin, K., Hammoud, M., Lee, E., Oh, J., Faloutsos, C.: Tri-fly: Distributed estimation of global and local triangle counts in graph streams. In: PAKDD, pp. 651–663 (2018)
38. Shin, K., Kim, J., Hooi, B., Faloutsos, C.: Think before you discard: Accurate triangle counting in graph streams with deletions. In: ECML/PKDD, pp. 141–157 (2018)
39. Shin, K., Lee, E., Oh, J., Hammoud, M., Faloutsos, C.: Cocos: Fast and accurate distributed triangle counting in graph streams. arXiv preprint arXiv:1802.04249 (2018)
40. Shin, K., Oh, S., Kim, J., Hooi, B., Faloutsos, C.: Fast, accurate and provable triangle counting in fully dynamic graph streams. ACM Transactions on Knowledge Discovery from Data **14**(2), 1–39 (2020)
41. Shun, J., Tangwongsan, K.: Multicore triangle computations without tuning. In: ICDE, pp. 149–160 (2015)
42. Spearman, C.: The proof and measurement of association between two things. The American journal of psychology **15**(1), 72–101 (1904)
43. Suri, S., Vassilvitskii, S.: Counting triangles and the curse of the last reducer. In: WWW, pp. 607–614 (2011)
44. Tsourakakis, C.E.: Fast counting of triangles in large real networks without counting: Algorithms and laws. In: ICDM, pp. 608–617 (2008)
45. Tsourakakis, C.E., Kang, U., Miller, G.L., Faloutsos, C.: Doulion: counting triangles in massive graphs with a coin. In: KDD, pp. 837–846 (2009)
46. Turk, A., Türkoglu, D.: Revisiting wedge sampling for triangle counting. In: TheWebConf, pp. 1875–1885 (2019)
47. Türkoglu, D., Turk, A.: Edge-based wedge sampling to estimate triangle counts in very large graphs. In: ICDM, pp. 455–464 (2017)
48. Viswanath, B., Mislove, A., Cha, M., Gummadi, K.P.: On the evolution of user interaction in facebook. In: WOSN, pp. 37–42 (2009)
49. Vitter, J.S.: Random sampling with a reservoir. ACM Transactions on Mathematical Software **11**(1), 37–57 (1985)
50. Wang, P., Qi, Y., Sun, Y., Zhang, X., Tao, J., Guan, X.: Approximately counting triangles in large graph streams including edge duplicates with a fixed memory usage. PVLDB **11**(2), 162–175 (2017)
51. Wasserman, S., Faust, K.: Social network analysis: Methods and applications, vol. 8. Cambridge university press (1994)
52. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world'networks. nature **393**(6684), 440–442 (1998)