2018

Counting Butterfies from a Large Bipartite Graph Stream

Seyed-Vahid Sanei-Mehri
Iowa State University, vas@iastate.edu

Yu Zhang
Iowa State University, yuz1988@iastate.edu

Ahmet Erdem Sariyuce
SUNY University at Buffalo

Srikanta Tirthapura
Iowa State University, snt@iastate.edu

Counting Butterfies from a Large Bipartite Graph Stream

Abstract
We consider the estimation of properties on massive bipartite graph streams, where each edge represents a connection between entities in two different partitions. We present sublinear-space one-pass algorithms for accurately estimating the number of butterflies in the graph stream. Our estimates have provable guarantees on their quality, and experiments show promising tradeoffs between space and accuracy. We also present extensions to sliding windows. While there are many works on counting subgraphs within unipartite graph streams, our work seems to be one of the few to effectively handle bipartite graph streams.

Disciplines
Computer Sciences | Electrical and Computer Engineering

Comments
This is a pre-print of the article Sanei-Mehri, Seyed-Vahid, Yu Zhang, Ahmet Erdem Sariyuce, and Srikanta Tirthapura. "Counting Butterfies from a Large Bipartite Graph Stream." *arXiv preprint arXiv:1812.03398* (2018). Posted with permission.

Counting Butterflies from a Large Bipartite Graph Stream*

Seyed-Vahid Sanei-Mehri†, Yu Zhang†, Ahmet Erdem Sarıyüce‡, Srikanta Tirthapura†

Abstract

We consider the estimation of properties on massive bipartite graph streams, where each edge represents a connection between entities in two different partitions. We present sublinear-space one-pass algorithms for accurately estimating the number of butterflies in the graph stream. Our estimates have provable guarantees on their quality, and experiments show promising tradeoffs between space and accuracy. We also present extensions to sliding windows. While there are many works on counting subgraphs within unipartite graph streams, our work seems to be one of the few to effectively handle bipartite graph streams.

1 Introduction

The discovery and counting of certain structural properties in a graph have emerged as key metrics to enhance the understanding of complex networks. Also recognized as graphlets or higher-order structures, graph motifs are important building blocks to characterize networks. The discovery and counting of motifs in a graph are one of the most important graph mining tasks and have captured tremendous attention in social networks, spam/fraud detection, and link recommendation.

Since a triangle is the smallest non-trivial complete subgraph, it plays a crucial role in the analysis of unipartite networks. For example, the number of triangles in a network is used in measuring the clustering coefficient or transitivity ratio. There has been extensive work on counting the number of triangles in a static graph [6, 9, 24] and a dynamic graph [4, 8, 9, 12, 14, 19, 21, 22, 24, 26, 34, 42, 43, 46–48, 50]. Due to the massive scale of real-life networks, and rapid rate of growth, it is imperative to develop agile methods that are capable of processing using memory sub-linear in the stream size. In many cases, an approximate answer to a query is sufficient, and we can obtain favorable tradeoffs with respect to memory, speed of processing, and accuracy.

We focus on mining from *bipartite graph* streams. A bipartite graph consists of two disjoint node sets L and R, and each edge in the graph connects a node in L with a node in R. Bipartite graphs are widely used in modeling real-world relationships. For instance, relationships between authors and papers can be modeled as a bipartite graph, where authors form one node partition, papers form the other node partition, and an author has an edge to each paper that she published. In web search and data mining, bipartite graphs have been used to model relations between queries and URLs in query logs [25], matching users to advertisements in computational advertising [2, 29], and author-paper relations in the scientific literature [15]. In computational biology, bipartite graphs are used to model enzyme-reaction links in metabolic pathways and gene-disease associations [35]. Other examples include user-product relations, word-document affiliations, and actor-movie networks. Bipartite graphs can be used to represent hypergraphs that capture many-to-many relations among entities, through having the hyperedges in one partition, and the elements in another partition.

While the literature is rich in methods for discovery of motifs in unipartite networks, these do not directly apply to bipartite networks. For instance, the number of triangles is not a useful metric, since a bipartite graph is triangle-free. Instead, the most basic motif which models cohesion in a bipartite network is the complete 2×2 biclique, also known as a butterfly [5, 39, 40] or a rectangle [49]. The butterfly has been used in defining the clustering coefficient in a bipartite graph [27,38] and can be considered as playing the same role in bipartite networks as the triangle did in unipartite networks – a building block for community structure.

1.1 Our Contributions:

We present one-pass streaming algorithms for estimating the total number of butterflies in a bipartite graph stream. Our algorithms use a fixed-size underlying storage that is sub-linear in the size of the stream, and continuously maintain an estimate of the number of butterflies as more edges arrive in the stream. The estimates are updated quickly as new edges arrive. Our algorithms are simple to implement, backed up by theoretical guarantees, and have good practical performance.

– **Infinite Window Streaming.** We first present algorithms for estimating the number of butterflies within an *infinite window* consisting of all edges seen

*Supported by NSF grants 1725702, 1527541

†{vas,yuz1988,snt}@iastate.edu, Iowa State University

‡erdem@buffalo.edu, University at Buffalo

so far in the stream. The first of these algorithms RES, is based on maintaining a random sample of the edges seen so far, followed by measuring the number of butterflies in the sample. The technical difficulty lies in the analysis of the variance of this estimator. While the sample is such that different edges in the sample are chosen with only weak dependence on each other[1], the occurrence of different butterflies in the sample depends on each other in complex ways, and an analysis needs to consider the ways in which multiple butterflies interact, in order to derive a bound on the variance. We then present an improvement called IRES, which leads to a better accuracy (i.e., smaller variance) than RES using the same memory, but has a larger runtime per element. We further present ADA algorithm that provides a different type of tradeoff, with an accuracy comparable to RES, but a smaller runtime per element. We present an analysis showing that each estimator is unbiased and give provable guarantees on the variance of each estimator.

– **Sliding Window Streaming.** We next present extensions of our algorithm to the sliding window model [7, 11, 18] which considers aggregation over the set of the W most recent edges in the stream.

– **Experimental Evaluation.** We present an evaluation of the performance of our algorithms on real-world graph streams. Our results show that proposed algorithms are able to achieve a relative error of less than 1% using samples of size approximately 100K edges on graphs that have ten millions of edges and more than a trillion butterflies. Our methods enjoy different tradeoffs between memory space, estimation accuracy, and runtime performance that make them practical for real-world applications with different requirements.

1.2 Related Works

Network motifs. Network motifs are small subgraphs that are defined on a few nodes and edges. Unlike graph communities or dense subgraphs, whose sizes do not have to be bounded, network motifs are typically subgraphs with less than six nodes. A similar concept is graphlets [37]. Network motif detection and counting is now an indispensable tool in network analysis [33]. The distribution of motif counts in a network, as well as the number of motifs that a node takes part in, help characterize the roles of networks and nodes [32]. Motifs and graphlets have been used in numerous applications in networking, web and social network analysis, and computational biology [9, 16, 17, 20, 30, 31, 41, 44, 45, 49].

Butterfly Counting. There have been relatively few works on counting motifs in a bipartite graph. Wang et al. [49] presented exact algorithms for butterfly counting in static graphs that outperform generic matrix multiplication based methods [6]. Sanei-Mehri et al. [39] proposed exact and randomized algorithms for large static bipartite graphs. Both these works have considered static graphs and not graph streams.

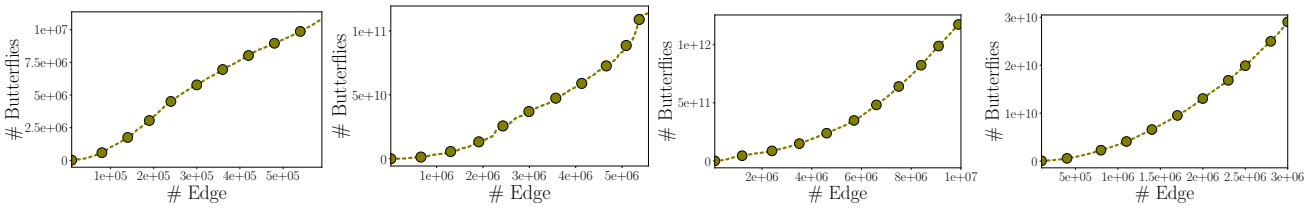
Motif Counting in Graphs. There are a number of algorithms known for triangle counting for unipartite streaming graphs [4, 8, 9, 12, 14, 19, 21, 22, 24, 26, 34, 42, 43, 46, 47, 50]. There has been some recent work on counting 4-vertex [3] and 5-vertex subgraphs [36], but these works focus on exact counting and are not designed for streaming graphs. Note that butterfly is a 4-cycle and there are some previous studies that counts cycles in unipartite graph streams [10, 13, 23, 28]. Bordino et al. [10] presents 4-cycle counting algorithms for graphs, but these take three passes through data, while we focus on single-pass streaming algorithms. Buriol et al. [13] consider 3,3-biclique counting in unipartite graph streams. They use the *incidence stream* model, where all edges incident to a given vertex arrive together, where as we work in the more general model (sometimes called the adjacency stream model), where the edges of the graph can arrive in an arbitrary order. Algorithms designed for incidence streams do not apply directly to our model. Manjunath et al. [28] and Kane et al. [23] introduce algorithms to count arbitrary size cycles in unipartite graph streams. These algorithms are based on sketches that use complex random variables, and to the best of our knowledge, these have not been implemented. In contrast, our algorithms are very easy to implement and evaluate.

2 Preliminaries

We consider simple unweighted and undirected bipartite graphs, without multiple edges between the same pair of vertices. Let $G = (V, E)$ be a bipartite graph with vertices V and edges E. The vertex set V of G is partitioned into two disjoint sets L and R. The edge set $E \subseteq L \times R$, so each edge e connects one vertex in set L and the other in set R. A **butterfly** is subgraph on four vertices $\{a, b, x, y\} \subset V$, where $a, b \in L$ and $x, y \in R$ such that edges $(a, x), (a, y), (b, x)$ and (b, y) exist in the edge set E.

A **graph stream** is a sequence of edges $\mathcal{S} = e_1, e_2, \ldots$ where e_i is the i-th edge in the stream. For $t > 0$, let $\mathcal{S}^{(t)}$ denote the first t edges of the stream and let

[1] this is due to the fact that we employ sampling without replacement, which results in the dependence between pairs of edges



(a) `Lkml` (b) `En-wiki` (c) `Movie-lens` (d) `Digg`

Figure 1: Number of butterflies as a function of stream size.

$G^{(t)} = (V^{(t)}, E^{(t)})$ denote the graph formed by the first t edges, i.e. $G^{(t)} = \{e_1, e_2, \ldots, e_t\}$. For $0 \leq i \leq j \leq t$, let $\mathcal{S}^{(t)}(i,j)$ denote the substream from e_i to e_j (both inclusive) and graph $G(i,j) = \{e_i, e_{i+1}, \ldots, e_j\}$.

Our algorithms return estimates of the number of butterflies. These estimates are random variables that are *unbiased*, i.e their expectation equals the desired subgraph count. We also bound their variance, which conveys how tightly the estimate is concentrated around its expectation. Our results do not assume a specific distribution of inputs. Instead, the input graph stream, including the set of edges and their order of arrivals, could be generated by an adversary. The randomness is solely *internal* to the algorithm, using random number generators.

Infinite Window: For any $t > 0$ and a stream $\mathcal{S}$, the goal for butterfly counting over an infinite window is to continuously maintain (an estimate of) the number of butterflies in the graph $G^{(t)}$, as t changes. Throughout this paper, we denote $\bowtie^{(t)}$ the set of all butterflies in the graph $G^{(t)}$ (when t is clear from the context, we simplify the notation by $\bowtie$) and denote $|\bowtie^{(t)}|$ as the number of butterflies in $G^{(t)}$.

Sliding Window: A *sequence-based sliding window* is defined as the set of a specific number of most recent edges. For $t > 0$ and window size W, when observing edge e_t, this is the set of edges $e_{t-W+1}, e_{t-W+2}, \ldots e_t$. Our goal is to maintain the number of butterflies that are constructed out of edges within this window. A generalization of sequence-based window is a *timestamp-based sliding window*, defined as the set of edges whose timestamps are most recent. Here, the window size does not correspond to a specific number of edges, but instead to a range of timestamps.

Networks and Experimental Setup. We present experimental results right after each algorithm description. We selected different types of real-world temporal bipartite networks from publicly available KONECT repository[2]. Basic properties of the networks are summarized in **??**. `Lkml` is the bipartite network of users and threads in the Linux Kernel Mailing list and each edge represents a timestamped post by a user. `En-wiki` is the edit network of the English Wiktionary and contains the edit activities of editors on pages. `Movie-lens` is the movie ratings by users for movies. `Digg` has the voting activities of users for news stories in the news aggregator website Digg.com. Note that, first three networks had multiple edges between the same node pairs and we simplified those by considering only the first interaction occurring between a node pair. Edges are read from the stream in the order of timestamps. **??** presents the number of butterflies with the increasing stream size (with respect to timestamps) for all graphs. We implemented all the streaming algorithms in C++ and the source code is available at [1]. We used gcc compiler with the optimization level O3. We used a machine equipped with a 2.0 GHz 8-Core Intel E5 2650 and 64.0 GB memory to run the experiments..

Graphs	#E	#V(Left)	#V(Right)	$\lvert\bowtie\rvert$	Description
`Lkml`	599,858	42,045	337,509	10M	Thread posts by users
`En-wiki`	5,573,038	29,348	2,094,520	113B	Page edits by editors
`Movie-lens`	10,000,054	69,878	10,677	1.1T	Movie ratings by users
`Digg`	3,010,898	139,409	3553	29B	Votes for stories by users

Table 1: Properties of the bipartite graphs. $|\bowtie|$ represents the total number of butterflies in a graph.

3 Streaming Algorithms for Infinite Windows

We present streaming algorithms for estimating the number of butterflies over an infinite window, i.e. all edges seen so far. We introduce fast randomized algorithms, Res, IRes, and Ada, which maintain an unbiased estimate of number of butterflies in an edge stream and use a sample (reservoir) with a fixed size. Our algorithms use a helper procedure, eBFC (e, E), which returns the number of local butterflies that contain edge e in the graph induced by edge set E – see supplementary material for more details.

We define two formulas, $\Psi_x(t)$ and p_z to help with the analysis. $\Psi_x(t)$ is the extrapolation factor needed for butterfly count estimates. When $t \leq M$, $\Psi_x(t)$ is 1. When $t > M$, it is adjusted to accommodate last x edges in the stream, and is defined as $\Psi_x(t) = \max\{1, \prod_{i=0}^{x-1}(t-i)/(M-i)\}$ where M is the size of reservoir and $x \leq t$. For $z = 0, 1, 2$, let p_z denote the number of *butterfly pairs* that have z edges in common. Note that two distinct butterflies in a bipartite graph

[2] http://konect.uni-koblenz.de/

Algorithm 1: RES $(\mathcal{S}, M)$: Reservoir sampling

Input: Edge stream $\mathcal{S}$ for reservoir of size $M\ (\geq 8)$
Output: $|\bowtie^{(t)}|$, estimate number of butterflies at t

1 $\mathcal{R} \leftarrow \varnothing, t \leftarrow 0, \beta \leftarrow 0$
2 **for** each edge e from stream $\mathcal{S}$ **do**
3 $t \leftarrow t + 1$
4 **if** $t \leq M$ **then**
5 $\mathcal{R} \leftarrow \mathcal{R} \cup e$
6 $\beta \leftarrow \beta + \text{EBFC}\,(e, \mathcal{R})$
7 **else if** `coin` (M/t) *is* `Head` **then**
8 $e' \leftarrow$ a random edge from $\mathcal{R}$
9 $\beta \leftarrow \beta - \text{EBFC}\,(e', \mathcal{R})$
10 $\mathcal{R} \leftarrow (\mathcal{R} \setminus \{e'\}) \cup e$
11 $\beta \leftarrow \beta + \text{EBFC}\,(e, \mathcal{R})$
12 $|\bowtie^{(t)}| \leftarrow \beta \times \Psi_4(t)$

cannot have three edges in common – if they do, they will have to share the fourth edge also.

3.1 Reservoir Sampling (RES): We start with the simple algorithm based on reservoir sampling, RES, which serves as a baseline for the other more involved algorithms. The intuition behind RES is to sample edges uniformly using a fixed-size reservoir, which ensures that the set of edges is chosen uniformly without replacement from the stream so far. **??** presents the RES algorithm. Each time an edge is inserted into the reservoir, the butterfly count is updated (Lines 4-6). Note that, each butterfly is counted only once, by the last edge which completes the butterfly formation. Once the reservoir is full, each insertion into the reservoir results in a deletion of an edge, upon which the butterfly count is adjusted by excluding those butterflies that contain the deleted edge (Lines 7-11). At a given time t, total number of butterflies in the edge stream seen so far can be estimated by extrapolating the butterfly count of reservoir by $\Psi_4(t)$, which accounts for the probability of sampling the four edges that form the butterfly.

We show that RES yields an unbiased estimate of $\bowtie(G^{(t)})$. Let $Y^{(t)}_{\text{RES}}$ is the return value of RES at time t.

LEMMA 1. $\mathbb{E}\left[Y^{(t)}_{\text{RES}}\right] = |\bowtie^{(t)}|$

Proof. Suppose that butterflies in $\bowtie^{(t)}$ are numbered from 1 to $|\bowtie^{(t)}|$. Let $X^{(t)}_i (1 \leq i \leq |\bowtie^{(t)}|)$ be a random variable equal to 1 if the i^{th} in $\bowtie^{(t)}$ appears in the reservoir $\mathcal{R}$, otherwise $X^{(t)}_i$ is 0. Let $X^{(t)} = \sum_{i=1}^{|\bowtie^{(t)}|} X^{(t)}_i$. Therefore, we have $Y^{(t)}_{\text{RES}} = \Psi_4(t) \cdot X^{(t)}$. In the case that $t \leq M$, all edges of the stream are selected and maintained in $\mathcal{R}$. Therefore, we have $X^{(t)} = \left|\bowtie^{(t)}\right|$. As $t \leq M$, we derive $Y^{(t)}_{\text{RES}} = \left|\bowtie^{(t)}\right|$.

When $t > M$, each edge is sampled uniformly at random since we use reservoir sampling approach. We know that each butterfly has four edges. Therefore,

$$\mathbb{E}\left[X^{(t)}_i\right] = \Pr\left[X^{(t)}_i = 1\right] = \frac{\binom{t-4}{M-4}}{\binom{t}{4}} = (\Psi_4(t))^{-1}$$

With the linearity of expectation, we obtain that $\mathbb{E}\left[X^{(t)}\right] = \sum_{i=1}^{|\bowtie^{(t)}|} \mathbb{E}\left[X^{(t)}_i\right] = \left|\bowtie^{(t)}\right| \times (\Psi_4(t))^{-1}$. Therefore, $\mathbb{E}\left[Y^{(t)}_{\text{RES}}\right] = \left|\bowtie^{(t)}\right|$.

In the next lemma, we show the variance of **??**. For an integer z $(0 \leq z \leq 2)$, let $h^{\text{RES}}_z(t) = \Psi_4(t) \times \prod_{k=4}^{7-z}\left(\frac{M-k}{t-k}\right) - 1$. As also previously mentioned, p_z represents the number of pairs of butterflies with z edges in common.

LEMMA 2. *Variance of $Y^{(t)}_{\text{RES}}$ is 0 for $t \leq M$. For $t > M$;*

$$\mathbb{V}\text{ar}\left[Y^{(t)}_{\text{RES}}\right] = |\bowtie^{(t)}|\,(\Psi_4(t) - 1) + 2 \cdot \sum_{z=0}^{2} p_z \cdot h^{\text{RES}}_z(t)$$

Proof. For an integer $i (1 \leq i \leq |\bowtie^{(t)}|)$. We have $\mathbb{V}\text{ar}[X^{(t)}_i] = \mathbb{E}[{X^{(t)}_i}^2] - \mathbb{E}^2[X^{(t)}_i] = (\Psi_4(t))^{-1} - (\Psi_4(t))^{-2}$.

$$\mathbb{V}\text{ar}\left[Y^{(t)}_{\text{RES}}\right] = \mathbb{V}\text{ar}\left[\Psi_4(t) \sum_{i=1}^{|\bowtie^{(t)}|} X^{(t)}_i\right]$$

$$= (\Psi_4(t))^2 \left[\sum_{i=1}^{|\bowtie^{(t)}|} \mathbb{V}\text{ar}[X^{(t)}_i] + \sum_{j \neq i} \mathbb{C}\text{ov}\left[X^{(t)}_i X^{(t)}_j\right]\right]$$

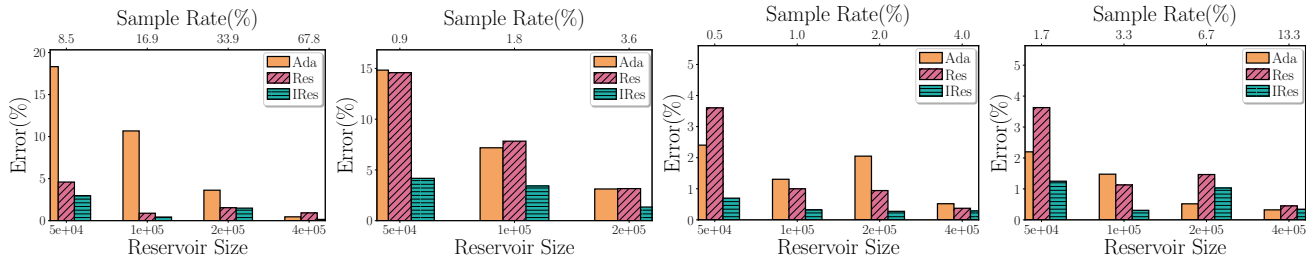
$$= \left|\bowtie^{(t)}\right| (\Psi_4(t) - 1) + (\Psi_4(t))^2 \times \sum_{j \neq i} \left(\mathbb{E}\left[X^{(t)}_i X^{(t)}_j\right] - \mathbb{E}\left[X^{(t)}_i\right]\mathbb{E}\left[X^{(t)}_j\right]\right)$$

Consider the case that i^{th} and j^{th} butterflies in $\bowtie^{(t)}$ do not share any edge. We have $\mathbb{E}[X^{(t)}_i X^{(t)}_j] = \Pr[X^{(t)}_i = 1] \Pr[X^{(t)}_j | X^{(t)}_i] = (\Psi_4(t))^{-1} \times \prod_{k=4}^{7} \frac{M-k}{t-k}$. Therefore, $(\Psi_4(t))^2 \cdot (\mathbb{E}[X^{(t)}_i X^{(t)}_j] - \mathbb{E}[X^{(t)}_i]\mathbb{E}[X^{(t)}_j]) = \Psi_4(t) \times \prod_{k=4}^{7} (\frac{M-k}{t-k}) - 1$.

When i^{th} and j^{th} butterflies have one edge in common, we get $\mathbb{E}[X^{(t)}_i X^{(t)}_j] = (\Psi_4(t))^{-1} \cdot \prod_{k=4}^{6} \frac{M-k}{t-k}$. Therefore, $(\Psi_4(t))^2 \cdot (\mathbb{E}[X^{(t)}_i X^{(t)}_j] - \mathbb{E}[X^{(t)}_i]\mathbb{E}[X^{(t)}_j]) = \Psi_4(t) \times \prod_{k=4}^{6} (\frac{M-k}{t-k}) - 1$.

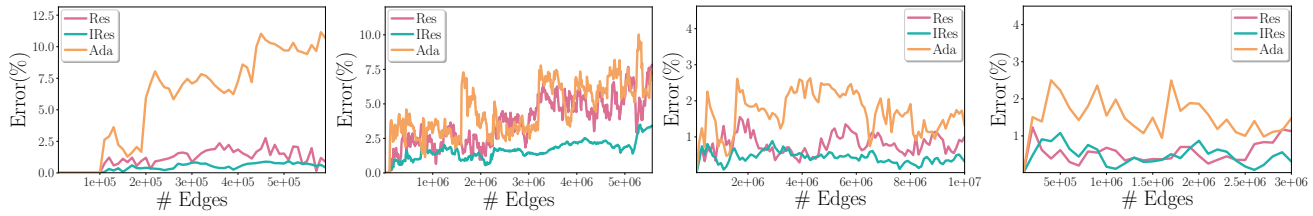
Lastly, if i^{th} and j^{th} butterflies have two edges in common, we derive $\mathbb{E}[X^{(t)}_i X^{(t)}_j] = (\Psi_4(t))^{-1} \cdot \prod_{k=4}^{5} \frac{M-k}{t-k}$. Therefore, $(\Psi_4(t))^2 \cdot (\mathbb{E}[X^{(t)}_i X^{(t)}_j] - \mathbb{E}[X^{(t)}_i]\mathbb{E}[X^{(t)}_j]) = \Psi_4(t) \times \prod_{k=4}^{5} (\frac{M-k}{t-k}) - 1$. Substituting $h^{\text{RES}}_z(t)$ and p_z in the variance of RES completes the proof.

Accuracy of estimates on real-world networks: Here we present the estimation errors of RES on real-world bipartite graph streams. The estimation error is the relative error of an estimate. If the true value of the



(a) `Lkml` (b) `En-wiki` (c) `Movie-lens` (d) `Digg`

Figure 2: Accuracy of estimates by RES, IRES, and ADA algorithms as a function of reservoir (sample) size. On the bottom x-axis is the sample size and on the top x-axis is the sample rate, defined as the ratio of the sample size to the stream size



(a) `Lkml` (b) `En-wiki` (c) `Movie-lens` (d) `Digg`

Figure 3: Accuracy of estimations by RES, IRES, and ADA as a function of stream size. Reservoir size is 100K for all graphs.

butterfly count is $x > 0$, then the error of an estimate $\hat{x}$ is defined as $\frac{|x-\hat{x}|}{x}$. We usually show these as percent error. **??** shows the estimation accuracies of RES for the entire graph stream when different reservoir (sample) sizes are used (ignore other algorithms for now). Larger reservoirs yield better accuracies, as expected. RES can keep the estimation error around 1% for `Lkml`, `Movie-lens`, and `Digg` networks by storing only 100K edges in the reservoir. This corresponds to the 16.7%, 3.3% and 0.9% of the total graph sizes for `Lkml`, `Digg` and `Movie-lens` networks, respectively. For `En-wiki` network, 3% error can be obtained with 200K edges – 3.6% of the graph size.

?? presents the accuracies of estimates while the graphs are being streamed. We observe a modest increase in the error rate with the increasing stream size. RES has a 0.6% error rate right after the reservoir gets full and it manages to keep it below 2% until the end. Trends are similar in other networks. We also observe some interesting rises and falls (e.g., `Digg`) with the increasing stream size, which might be due to the inhomogeneous distribution of butterfly formation in different time periods.

3.2 Improved Reservoir Sampling (IRES): We present an algorithm IRES, that improves the accuracy over RES by considering the contribution of *every edge in the stream*. As shown in **??**, upon receiving a new edge e, regardless of whether or not e is sampled, the estimated is updated by considering the number of butterflies that e forms along with the edges currently in the reservoir. Intuitively speaking, this approach manages to detect more butterflies than RES, since a butterfly is counted if the first three edges of the butterfly have been sampled at the time the fourth edge arrives. Whereas in RES, all four edges have to be sampled for the butterfly to be detected. In **????**, we show the expectation and variance of **??** respectively. Let $Y^{(t)}_{\text{IRES}}$ be the return value of IRES.

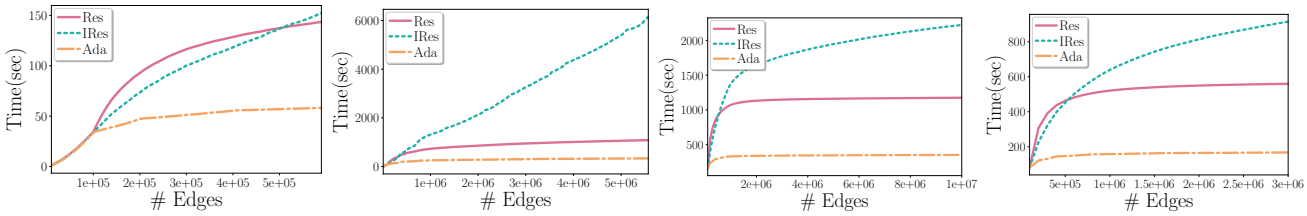
LEMMA 3. $\mathbb{E}[Y^{(t)}_{\text{IRES}}] = |\mathbb{X}^{(t)}|$

Suppose that at time t, butterflies in the stream so far are ordered based on the time stamp of their last edges from 1 to $|\mathbb{X}^{(t)}|$. For example, if $i > j$, the last edge of i^{th} butterfly does not arrive before the last edge of j^{th} butterfly in the stream. Let t_i denote the appearance time of the last edge of i^{th} butterfly on the stream ($1 \le i \le |\mathbb{X}^{(t)}|$). Let $X^{(t)}_i$ be an indicator random

Algorithm 2: IRES $(\mathcal{S}, M)$: Improved RES

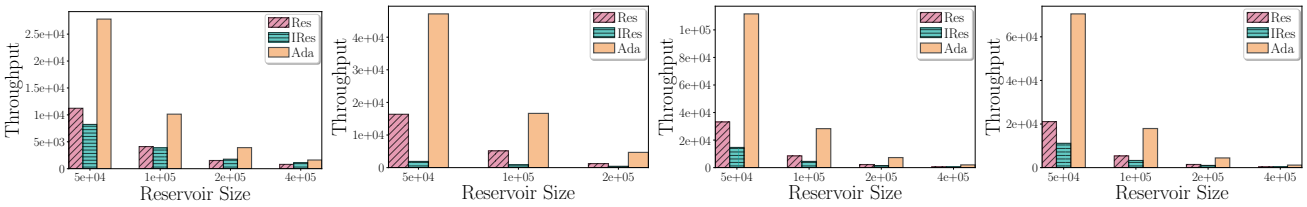
Input: Edge stream $\mathcal{S}$ for reservoir of size M (≥ 8)
Output: $|\mathbb{X}^{(t)}|$, estimate number of butterflies at t

$\mathcal{R} \leftarrow \varnothing, t \leftarrow 0, \beta \leftarrow 0$
for each edge e from stream $\mathcal{S}$ **do**
  $t \leftarrow t + 1$
  $\beta \leftarrow \beta + \Psi_3(t-1) \times \text{EBFC}\ (e, \mathcal{R} \cup e)$
  if $t \le M$ **then** $\mathcal{R} \leftarrow \mathcal{R} \cup e$
  else if `coin` (M/t) *is* `Head` **then**
    $e' \leftarrow$ a random edge from $\mathcal{R}$
    $\mathcal{R} \leftarrow (\mathcal{R} \setminus \{e'\}) \cup e$
  $|\mathbb{X}^{(t)}| \leftarrow \beta$



(a) `Lkml` (b) `En-wiki` (c) `Movie-lens` (d) `Digg`

Figure 4: Runtime behavior of RES, IRES, and ADA algorithms as a function of stream size when the reservoir size is 100K.



(a) `Lkml` (b) `En-wiki` (c) `Movie-lens` (d) `Digg`

Figure 5: Throughput of RES, IRES, and ADA algorithms as a function of reservoir size.

variable to show whether the first three edges of the i^{th} butterfly are found in the reservoir $\mathcal{R}$.

LEMMA 4. *If $t \leq M$, the variance of $Y^{(t)}_{\text{IRES}}$ is 0, otherwise we have* $\mathbb{V}\text{ar}\left[Y^{(t)}_{\text{IRES}}\right] = \sum_{i=1}^{|\mathbb{X}^{(t)}|}\left(\Psi_3(t_i-1)-1\right) + 2\times\sum_{j<i}\left(\mathbb{E}\left[X_i^{(t)}X_j^{(t)}\right]-1\right)$

Without loss of generality, we assume that $i > j$. If i^{th} and j^{th} butterflies do not share any edge, $\mathbb{E}[X_i^{(t)}X_j^{(t)}] - 1 \leq 0$. If they share only one edge, $\mathbb{E}[X_i^{(t)}X_j^{(t)}] - 1 \leq \Psi_1(t_i-1)$. If they share two edges, $\mathbb{E}[X_i^{(t)}X_j^{(t)}] - 1 \leq \Psi_2(t_i-1)$.

The proofs of **????** are moved to supplementary material due to space constraints.

Comparison of Variances of Res and IRes. Here, we compare the variances of **????**. Consider the case that the current time $t > M$. The first term in the variance of $Y^{(t)}_{\text{RES}}$ and $Y^{(t)}_{\text{IRES}}$ involve individual butterflies. Suppose the butterflies are numbered in some order. The contribution of the ith butterfly to the variance of $Y^{(t)}_{\text{RES}}$ is $|\mathbb{X}^{(t)}|(\Psi_4(t)-1)$ and its contribution to the variance of $Y^{(t)}_{\text{IRES}}$ is $\sum_{i=1}^{|\mathbb{X}^{(t)}|}(\Psi_3(t_i-1)-1)$, where t_i is the time of arrival of the last edge of the butterfly, which could be much smaller than the current time t. Hence, the first term in the variance of $Y^{(t)}_{\text{IRES}}$ is smaller than the first term in $Y^{(t)}_{\text{RES}}$.

In the second term of variances, we consider the contributions of butterfly pairs to the variances of $Y^{(t)}_{\text{IRES}}$ and $Y^{(t)}_{\text{RES}}$. Let i and j range over all pairs of butterflies that appear on stream at time t. Without loss of generality, assume that $i > j$, which means that the last edge of i^{th} butterfly does not arrive before the last edge of j^{th} butterfly on the stream. Let t_i denote the appearance time of the last edge of i^{th} butterfly on the stream ($1 < i \leq |\mathbb{X}^{(t)}|$). If butterflies i and j do not share an edge, the covariance of $Y^{(t)}_{\text{IRES}}$ is a non-positive value while the covariance of $Y^{(t)}_{\text{RES}}$ is non-negative. If they share *exactly* one edge, the covariance of $Y^{(t)}_{\text{IRES}}$ is based on the last edge of the i^{th} butterfly and is $\mathcal{O}(t_i/M)$ while the covariance of $Y^{(t)}_{\text{RES}}$ is $\mathcal{O}(t/M)$. If they share two edges, the covariance of $Y^{(t)}_{\text{IRES}}$ is $\mathcal{O}((t_i/M)^2)$, and the covariance of $Y^{(t)}_{\text{RES}}$ is $\mathcal{O}((t/M)^2)$. Thus, we verify that the variance of $Y^{(t)}_{\text{IRES}}$ cannot be greater than $Y^{(t)}_{\text{RES}}$.

Accuracy of estimates: As expected, we see from **??** and **??** that IRES has better accuracy that RES for any reservoir (sample) size on all graphs in **??**. In `En-wiki` network, for instance, error rate of IRES is 4% when the reservoir size is 50K while RES is able to give only 15%. For larger reservoir sizes, better accuracies are obtained, reaching around 1% for 200K reservoir size.

Runtime: The better accuracy of IRES comes at the cost of increased runtime performance (**??**). Note that the number of local butterfly computations in IRES equals the number of edges. On the other hand, RES shows a stable trend after the reservoir gets full, since the probability of an edge being sampled, and hence of a local butterfly computation decreases with time. **??** presents the throughput (edges processed per second) and RES has a significantly larger throughput than IRES on all graphs for all reservoir (sample) sizes, reaching up to 8.5× higher throughput on `En-wiki`. Another observation is that throughput numbers decrease with the increasing reservoir (sample) sizes for all graphs and algorithms, because the cost of local butterfly counting

Algorithm 3: ADA $(\mathcal{S}, M)$: Adaptive sampling

Input: Edge stream $\mathcal{S}$ for reservoir of size $M\ (\geq 8)$
Output: $|\bowtie^{(t)}|$, estimate number of butterflies at t

$p \leftarrow 1, \mathcal{R} \leftarrow \varnothing, t \leftarrow 0, \beta \leftarrow 0$
for each edge e from stream **do**
  $t \leftarrow t + 1$
  if $|\mathcal{R}| = M$ **then**
    $p \leftarrow p/2$
    Keep any edge in $\mathcal{R}$ with probability 0.5
    $\beta \leftarrow p^{-4} \times |\bowtie(\mathcal{R})|$ `// # butterflies in` $\mathcal{R}$
  if `coin` (p) *is* `Head` **then**
    $\mathcal{R} \leftarrow \mathcal{R} \cup e$
    $\beta \leftarrow \beta + p^{-4} \times$ EBFC $(e, \mathcal{R})$
  $|\bowtie^{(t)}| \leftarrow \beta$

increases with the reservoir size.

3.3 Adaptive Sampling (ADA):

Better accuracy of IRES, with respect to RES, comes with a larger runtime cost and this tradeoff can be leveraged by certain applications that do not have strong performance requirements. On the other end of the spectrum, however, some applications with fast edge streams might require higher throughput. Is there an algorithm that yield higher throughputs than RES without sacrificing accuracy by much? Note that the error rates by RES algorithms is around 1% for many cases and sacrificing another 1% or 2% might be okay for applications with fast edge streams as long as the throughput rate can match the incoming stream rate.

We present an adaptive sampling algorithm ADA where the sampling probability is adapted with respect to the stream size. In ADA, edges are selected and inserted to a reservoir with a sampling probability, which is set to 1 until the reservoir gets full. When there is no space in reservoir to add more edges, the sampling probability is reduced by a factor of half, and each edge in the reservoir is discarded with probability 0.5. It means that on expectation, only 50% of edges are kept in reservoir. At this step, the number of butterflies in the reservoir is recomputed. **??** outlines the ADA algorithm. Note that since we need to find the total number of butterflies in the reservoir (in **??**), we can use a batch Algorithm EXACTBFC of [39] which is a fast method for counting the total number of butterflies compared to performing EBFC for every single edge. In **??**, we show that ADA maintains an unbiased estimate of number of butterflies in the edge stream. Let $Y^{(t)}_{\text{ADA}}$ denote the estimate of butterfly count of **??** (**??**) at time t. Let $p^{(t)}$ denote the sampling probability at time t.

LEMMA 5. $\mathbb{E}[Y^{(t)}_{\text{ADA}}] = |\bowtie^{(t)}|$

Let $h^{\text{ADA}}_z(t) = (p^{(t)})^{-z} - 1$ for $0 \leq z \leq 2$, and p_z be the number of pair of butterflies with z edges in common.

LEMMA 6. *If $t \leq M$, the variance of $Y^{(t)}_{\text{ADA}}$ is 0, otherwise we have;*

$$\mathbb{V}\text{ar}[Y^{(t)}_{\text{ADA}}] = |\bowtie^{(t)}|((p^{(t)})^{-4} - 1) + 2 \times \sum_{z=0}^{2} p_z \cdot h^{\text{ADA}}_z(t)$$

Due to space constraints, proofs of **????** are moved to supplementary material.

Runtime and accuracy: From Figures 4 and 5, we see that ADA is significantly faster than RES and IRES on all graphs. With the increasing stream size, total runtime stabilizes earlier than the other alternatives. 3.29× and 3.09× speedups are observed in `Movie-lens` and `Digg` networks with respect to RES algorithm. Results for throughput rates show the significant speedup of ADA as well. The accuracy of ADA is comparable with RES on `En-wiki`, `Movie-lens`, and `Digg` networks. In `Lkml` network, RES performs significantly better than ADA when the reservoir size is 100K (**??**). However, the difference disappears when the reservoir size becomes 400K (**??**).

4 Sliding Window

We present the algorithm for a Sequence-based Sliding Window Butterfly Counting (SEQSW), and defer the algorithm for a time-based window to the appendix. As the sequence-based window contains a fix number of edges, we can sample every new edge with the same probability and use the sample edges to estimate the number of butterflies. We extend our Algorithm RES to use a fixed sampling rate when the stream size is larger than the window size, and present a description in Algorithm 4. Due to space constraints, we present the algorithm for handling a time-based sliding window in supplementary material.

Algorithm 4: SEQSW $(\mathcal{S}, M, W)$: Seq-based SW

Input: Edge stream $\mathcal{S}$, sampled edge set size $M\ (\geq 8)$, window size $W\ (\geq M)$
Output: $|\bowtie^{(t)}_W|$, estimate number of butterflies in window W at t

$\mathcal{R} \leftarrow \emptyset, t \leftarrow 0, \beta \leftarrow 0, p \leftarrow 1$
for each edge e from stream $\mathcal{S}$ **do**
  $t \leftarrow t + 1$
  $r \leftarrow \text{unif}(0, 1)$
  $\beta \leftarrow \beta +$ EBFC $(e, \mathcal{R})$
  $\mathcal{R} \leftarrow \mathcal{R} \cup (e, r, t)$ `// edge with prob and index`
  $p \leftarrow \min(t, M)/\min(t, W)$
  if $\mathcal{R} \ni (e', r', t')$ s.t. $(t' \leq (t - W) \vee r' > p)$ **then**
    $\beta \leftarrow \beta -$ EBFC $(e', \mathcal{R})$
    $\mathcal{R} \leftarrow \mathcal{R} \setminus (e', r', t')$
  $|\bowtie^{(t)}_W| \leftarrow \beta/p^4$

LEMMA 7. *The space cost of $\mathcal{R}$ in Algorithm 4 is no greater than M in expectation and* $\Pr(|\mathcal{R}| \geq 2M) \leq$

$\left(\frac{e}{4}\right)^M$. *Let* $|\bowtie_W^{(t)}|$ *denote the number of butterflies in the window of recent* W *edges. Algorithm 4 returns an unbiased estimate of* $|\bowtie_W^{(t)}|$ *with variance*

$$\mathbb{V}\mathrm{ar}(\beta/p^4) = |\bowtie^{(W)}| \cdot (p^{-4} - 1) + 2\sum_{z=0}^{2} p_{z,W} \cdot (p^{-z} - 1)$$

where $p_{z,W}$ *is the number of pairs of butterflies in window that each pair is having* z *edges in common* $(0 \le z \le 2)$.

Proof. Consider the value of probability p in under different cases on number of edges received. When $t \le M$, probability p is 1, the number of edges in $\mathcal{R}$ is strictly no greater than M. When $M < t \le W$, $p = \frac{M}{t}$ and $\mathbb{E}(|\mathcal{R}|) = t \cdot \frac{M}{t} = M$. When $t > W$, $p = \frac{M}{W}$, the number of edges to be sampled is the window size W, $\mathbb{E}(|\mathcal{R}|) = W \cdot \frac{M}{W} = M$.

So at any time, we prove that $\mathbb{E}(\mathcal{R}) \le M$. Consider the case $m > M$, $\mathbb{E}(|\mathcal{R}|) = M$. Note that the process of keeping an edge in $\mathcal{R}$ is a Bernoulli trial, by Chernoff bound we have $\Pr(|\mathcal{R}| > 2 \cdot \mathbb{E}(|\mathcal{R}|)) \le \left(\frac{e}{4}\right)^{\mathbb{E}(|\mathcal{R}|)}$ Thus, $\Pr(|\mathcal{R}| > 2M) < \left(\frac{e}{4}\right)^M$, our claim on space bound of $\mathcal{R}$ is proved.

At any given time, each edge (in the window) is in the sampled edge set $\mathcal{R}$ with probability p, $\mathbb{E}(\beta) = |\bowtie_W^{(t)}| \cdot p^4$, Algorithm 4 returns an unbiased number of butterflies in window. We rely on an analysis similar to the analysis of Algorithm Ada, to derive a bound on the variance.

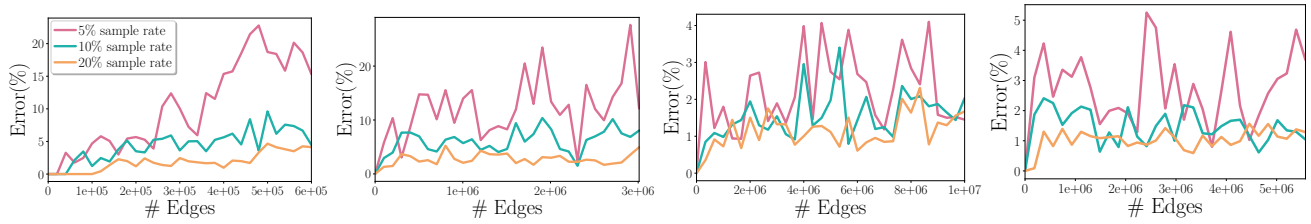
Accuracy: We compute the estimation error by fixing the window size W ito $500,000$ edges and varying the size of samples M to be 5%, 10% and 20% of the window size W (Figure 6). We observe that the accuracy improves with a larger sample rate, for example when sample rate is 20% the result on dataset `Digg` shows the error is always less than 2%, however when sample rate is 5%, the error is almost above 2% and can reach up to 5%. From Lemma 7, variance of the estimate becomes smaller when the sampling rate increases.

5 Conclusion

We presented one-pass streaming algorithms for estimating the number of butterflies in a bipartite graph stream. Our algorithms are primarily based on sampling the edges of the graph stream, and building estimators using these sampled edges; in some cases such as IRes, we consider interactions between edges that are not sampled along with those that are sampled. We provide three algorithms Res, IRes, and Ada, with different tradeoffs for accuracy, memory, and runtime. We also considered the sliding window model. Our experiments show good accuracy on large real-world graphs; for instance, relative error less than 2.5% using a sample size of 100K edges for graphs with millions of edges. This work is one of the first to explore streaming motif counting on bipartite graphs, and leads to many follow-up questions. (1) Can these be extended to estimate general motif counts on bipartite graph streams? (2) Is it possible to combine the benefits of improved accuracy as in IRes, with faster runtime as in Ada, into a single algorithm? (3) Can the algorithms take the benefit of multi-pass and external memory models?

References

[1] `https://github.com/yuz1988/stream-bfly-counting`.
[2] G. Aggarwal et al. Biobjective online bipartite matching. In *WINE*, pages 218–231, 2014.
[3] N. Ahmed et al. Efficient graphlet counting for large networks. In *ICDM*, pages 1–10, 2015.
[4] N. Ahmed et al. On sampling from massive graph streams. *Proc. VLDB Endow.*, 10(11):1430–1441, August 2017.
[5] S. Aksoy et al. Measuring and modeling bipartite graphs with community structure. *J. Complex Networks*, 5(4), 2017.
[6] N. Alon et al. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.
[7] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *SODA*, 2002.
[8] Z. Bar-Yossef et al. Reductions in streaming algorithms, with an app. to counting triangles in graphs. In *SODA*, 2002.
[9] L. Becchetti et al. Efficient algorithms for large-scale local triangle counting. *TKDD*, 4(3):13:1–13:28, October 2010.
[10] I. Bordino et al. Mining large networks with subgraph counting. In *ICDM*, pages 737–742, 2008.
[11] V. Braverman, R. Ostrovsky, and C. Zaniolo. Optimal sampling from sliding windows. In *PODS*, 2009.
[12] L. Bulteau et al. Triangle counting in dynamic graph streams. *Algorithmica*, 76(1):259–278, September 2016.
[13] Luciana S. Buriol et al. Estimating clustering indexes in data streams. In *ESA*, 2007.
[14] X. Chen and J. Lui. A unified framework to estimate global and local graphlet counts for streaming graphs. In *ASONAM*, 2017.
[15] H. Deng et al. A generalized co-hits algorithm and its application to bipartite graphs. In *SIGKDD*, 2009.
[16] F. Faisal et al. Global network alignment in the context of aging. *IEEE/ACM Tran. Comp. Biology and Bioinfo.*, 12(1):40–52, 2015.
[17] F. Faisal and T. Milenković. Dynamic networks reveal key players in aging. *Bioinformatics*, 30(12), 2014.
[18] P. B. Gibbons and S. Tirthapura. Distributed streams algorithms for sliding windows. In *SPAA*, pages 63–72, 2002.
[19] G. Han and H. Sethu. Edge sample and discard: A new algorithm for counting triangles in large dynamic graphs. In *ASONAM*, 2017.
[20] H. Ho et al. Protein interaction network topology uncovers melanogenesis regulatory network components within functional genomics datasets. *BMC systems biology*, 4(1), 2010.
[21] M. Jha et al. Path sampling: A fast and provable method for estimating 4-vertex subgraph counts. In *WWW*, 2015.
[22] H. Jowhari and M. Ghodsi. New streaming algorithms for counting triangles in graphs. In *COCOON*, 2005.



(a) `Lkml` (b) `En-wiki` (c) `Movie-lens` (d) `Digg`

Figure 6: Error to the exact number of butterflies in the sliding window vs. number of edges received, window size is 500,000 edges and size of the sampled edge set M is set to 5%, 10% and 20% to the window size respectively.

[23] Daniel M. Kane et al. Counting arbitrary subgraphs in data streams. In *ICALP*, pages 598–609, 2012.

[24] M. Kolountzakis et al. Efficient triangle counting in large graphs via degree-based vertex partitioning. *Internet Mathematics*, 8(1-2):161–185, 2012.

[25] L. Li et al. Query-url bipartite based approach to personalized query recommendation. In *AAAI*, 2008.

[26] Y. Lim and U Kang. Mascot: Memory-efficient and accurate sampling for counting local triangles in graph streams. In *SIGKDD*, 2015.

[27] P. Lind, M. González, and H. Herrmann. Cycles and clustering in bipartite networks. *Phys. Rev. E*, 72, 2005.

[28] M. Manjunath et al. Approximate counting of cycles in streams. In *ESA*, 2011.

[29] A. Mehta. Online matching and ad allocation. *Foundations and Trends in Theoretical Comp. Sci.*, 8(4), 2013.

[30] T. Milenković et al. Optimal network alignment with graphlet degree vectors. *Cancer informatics*, 9, 2010.

[31] T. Milenković et al. Dominating biological networks. *PloS one*, 6(8), 2011.

[32] T. Milenković and N. Pržulj. Uncovering biological network function via graphlet degree signatures. *Cancer informatics*, 6, 2008.

[33] R. Milo et al. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594), 2002.

[34] A. Pavan et al. Counting and sampling triangles from a graph stream. *Proc. VLDB Endow.*, 6(14), September 2013.

[35] G. Pavlopoulos et al. Bipartite graphs in systems biology and medicine: a survey of methods and applications. *GigaScience*, 7(4), 2018.

[36] A. Pinar et al. ESCAPE: Efficiently Counting All 5-Vertex Subgraphs. In *WWW*, 2017.

[37] N. Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2), 2007.

[38] G. Robins and M. Alexander. Small worlds among interlocking directors: Network structure and distance in bipartite graphs. *Comp. & Math. Org. Theory*, 10(1):69–94, 2004.

[39] S. Sanei-Mehri et al. Butterfly counting in bipartite networks. In *SIGKDD*, 2018.

[40] A. E. Sarıyüce and A. Pinar. Peeling bipartite networks for dense subgraph discovery. In *WSDM*, 2018.

[41] D. Schiöberg et al. Evolution of directed triangle motifs in the Google+ OSN. *CoRR*, abs/1502.04321, 2015.

[42] K. Shin et al. Think before you discard: accurate triangle counting in graph streams with deletions. In *ECML,PKDD*, 2018.

[43] K. Shin et al. Tri-Fly: Distributed estimation of global and local triangle counts in graph streams. In *PAKDD*, 2018.

[44] O. Singh et al. Graphlet signature-based scoring method to estimate protein–ligand binding affinity. *Royal Society open science*, 1(4), 2014.

[45] R. Solava et al. Graphlet-based edge clustering reveals pathogen-interacting proteins. *Bioinformatics*, 28(18), 2012.

[46] L. Stefani et al. Tiered sampling: An efficient method for approximate counting sparse motifs in massive graph streams. In *IEEE Big Data*, 2017.

[47] L. Stefani et al. Triest: Counting local and global triangles in fully dynamic streams with fixed memory size. *TKDD*, 11(4), 2017.

[48] K. Tangwongsan et al. Parallel triangle counting in massive streaming graphs. In *CIKM*, 2013.

[49] J. Wang, A. Fu, and J. Cheng. Rectangle counting in large bipartite graphs. In *IEEE Big Data*, 2014.

[50] P. Wang et al. Approximately counting triangles in large graph streams including edge duplicates with a fixed memory usage. *PVLDB*, 2017.

Supplementary Material.

Local Butterfly Counting. For a given bipartite graph G and a given edge e, we define **??** which counts the number of butterflies containing e. In **??** for a vertex u in G, Γ_u maintains the neighbors of u in G. V_E is the vertex set that is induced by the edge set E.

Algorithm 5: EBFC (e, E): Butterfly # per-edge

Input: An edge $e = \langle u, v\rangle \in E$ in $G = (V_E, E)$
Output: β_e, number of butterflies that contain e

```
β_e ← 0
for w ∈ Γ_u \ {v} do
    for x ∈ Γ_w do if x ∈ Γ_v \ {u} then β_e ← β_e + 1
return β_e
```

Proof of ??.

Let all butterflies in $\bowtie^{(t)}$ are ordered based on the time stamp of their last edges from 1 to $|\bowtie^{(t)}|$. For example, if $i > j$, the time step of last edge i^{th} butterflies does not appear before the last edge of j^{th} butterfly.

Let t_i denote the appearance time of the last edge of i^{th} butterfly on the stream ($1 \le i \le |\bowtie^{(t)}|$). Let $X_i^{(t)} (1 \le i \le |\bowtie^{(t)}|)$ be a random variable and takes $\Psi_3(t_i - 1)$ if all edges of i^{th} butterflies, except the last edge, found in the reservoir $\mathcal{R}$ at time $t_i - 1$, otherwise $X_i^{(t)}$ is 0.

If $t \le M$, all edges are maintained in $\mathcal{R}$. Therefore, $\mathbb{E}[Y_{\text{IRES}}^{(t)}] = |\bowtie^{(t)}|$. In the other case, we show that the estimate of IRES at any time $t > M$ is unbiased.

At time $t_i - 1$ when the last edge of i^{th} butterfly arrives, $X_i^{(t)}$ is equal to $\Psi_3(t_i - 1)$ if three other edges of the butterfly appears in the reservoir $\mathcal{R}$. Therefore, $\Pr[X_i^{(t)} = \Psi_3(t_i - 1)] = 1/\Psi_3(t_i - 1)$.

According to **??**, we can verify that $Y_{\text{IRES}}^{(t)} = \sum_{i=1}^{t} X_i^{(t)}$. Using linearity of expectation, we can derive that $\mathbb{E}[Y_{\text{IRES}}^{(t)}] = \sum_{i=1}^{t} \mathbb{E}[X_i^{(t)}] = |\bowtie^{(t)}|$.

Proof of ??.

At time t, for an integer $i (1 \le i \le |\bowtie^{(t)}|)$, we have

$$\mathbb{V}\text{ar}[X_i^{(t)}] = \mathbb{E}[(X_i^{(t)})^2] - E^2[X_i^{(t)}] = \Psi_3(t_i - 1) - 1.$$

$$\begin{aligned}
\mathbb{V}\text{ar}[Y_{\text{IRES}}^{(t)}] &= \mathbb{V}\text{ar}\left[\sum_{i=1}^{|\bowtie^{(t)}|} X_i^{(t)}\right] \\
&= \sum_{i=1}^{|\bowtie^{(t)}|} \mathbb{V}\text{ar}[X_i^{(t)}] + \sum_{i \ne j} \mathbb{C}\text{ov}\left[X_i^{(t)} X_j^{(t)}\right] \\
&\le |\bowtie^{(t)}| \left(\Psi_3(t_i - 1) - 1\right) + 2 \times \sum_{j<i} \mathbb{C}\text{ov}\left[X_i^{(t)}, X_j^{(t)}\right] \\
&= |\bowtie^{(t)}| \left(\Psi_3(t_i - 1) - 1\right) + 2 \times \sum_{j<i} \left(\mathbb{E}\left[X_i^{(t)} X_j^{(t)}\right] - 1\right)
\end{aligned}$$

Here, we define some notations used in the analysis of our proof for **??**. Suppose that all butterflies appeared on stream at time t are ordered by the time step of their last edges. Let t_i denote the time stamp of last edge of the i^{th} butterfly. Then, if $i > j$, we have $t_i \ge t_j$. Let also the i^{th} butterfly contains four edges $\{e_1^i, e_2^i, e_3^i, e_4^i\}$, ordered by their time appearance on the stream. Similarly, the j^{th} butterfly includes four edges $\{e_1^j, e_2^j, e_3^j, e_4^j\}$, sorted by their time step, they appear on the stream. We also denote $\sigma(t) \in \{0, 1\}$. $\sigma(t)$ is 1 if $t - 1 \le M$, otherwise it is zero.

Let i and j range over all pairs of butterflies at time t. Without loss of generality, we assume that $i > j$. To compute the the contribution of i^{th} and j^{th} butterflies to the variance of $Y_{\text{IRES}}^{(t)}$, we consider three cases:

– If i^{th} and j^{th} butterflies have no edge in common, we can show that they have non-positive contribution to the variance of $Y_{\text{IRES}}^{(t)}$. We prove this claim by considering different scenarios.

If we have $t_{e_1^j} < t_{e_2^j} < t_{e_3^j} < t_{e_4^j} < t_{e_1^i} < t_{e_2^i} < t_{e_3^i} < t_{e_4^i}$, two events $X_i^{(t)}$ and $X_j^{(t)}$ are independent. Therefore, we have $\mathbb{E}[X_i^{(t)} X_j^{(t)}] - \mathbb{E}[X_i^{(t)}]\mathbb{E}[X_j^{(t)}] = 0$.

Consider the case that for $k \in \{1, 2, 3\}$, the time step of edges $e_1^i, \cdots, e_k^i$ are found before $t_{e_j^4}$ on the stream. For simplicity, we define the following events to consider this case. Let an event E_1^k indicate that the edges $e_1^i, \cdots, e_k^i$ are maintained in the reservoir $\mathcal{R}$ at the time step $t_{e_4^j}$. Let E_2^k denote the event that the edge e_{k+1}^i is inserted to $\mathcal{R}$ at time $t_{e_{k+1}^i}$, if $k < 3$. If $k = 3$, E_2^k happens with probability 1. Let E_3 indicate edges e_1^i, e_1^i, and e_3^i are maintained in the reservoir $\mathcal{R}$ at time $t_{e_4^i}$. Therefore, it is verified that for some integer $k \in \{1, 2, 3\}$, we have $X_i^{(t)} = \cap_{z=1}^{2} E_z^k$.

Therefore, we have $\Pr[X_i^{(t)} | X_j^{(t)}] = \Pr[E_1^k \cap E_2^k \cap E_3 | X_j^{(t)}] = \Pr[E_1^k | X_j^{(t)}] \cdot \Pr[E_2^k \cap E_3 | E_1^k \cap X_j^{(t)}]$. If $t_{e_4^i} - 1 \le M$, by **??**, we know that all edges are

maintained. Therefore, $\Pr[E_1^k|X_j^{(t)}] = 1$.

If $t_{e_4^i} - 1 > M$, we have

$$\Pr[E_1^k|X_j^{(t)}] = \frac{\binom{t_{e_4^j}-4-k}{M-3-k}}{\binom{t_{e_4^j}-4}{M-3}} = \prod_{z=0}^{k-1} \frac{M-3-z}{t_{e_4^j}-4-z}$$

We also know that

$$\Pr[E_1^k] = \frac{\binom{t_{e_4^j}-1-k}{M-k}}{\binom{t_{e_4^j}-1}{M}} = \prod_{z=0}^{k-1} \frac{M-z}{t_{e_4^j}-1-z}$$

By comparing each term in $\prod_{z=0}^{k-1} \frac{M-3-z}{t_{e_4^j}-4-z}$ and $\prod_{z=0}^{k-1} \frac{M-z}{t_{e_4^j}-1-z}$, it is verified that $\Pr[E_1^k|X_j^{(t)}] \leq \Pr[E_1^k]$. In addition, we know that both E_2^k and E_3 are independent from the event $X_j^{(t)}$. It implies that $\Pr[X_i^{(t)}|X_j^{(t)}] \leq \Pr[E_1^k] \cdot \Pr[E_2^k \cap E_3|E_1^k] = \Pr[E_1^k \cap E_2^k \cap E_3] = \Pr[X_i^{(t)}]$. Therefore $\mathbb{E}[X_i^{(t)}X_j^{(t)}] - \mathbb{E}[X_i^{(t)}]\mathbb{E}[X_j^{(t)}] \leq 0$.

Here, we consider the case that two butterflies share an edge c.

If the edge c is the last edge of i^{th} butterfly, other 6 edges of two butterflies should be kept in the reservoir $\mathcal{R}$. Therefore, we have $\Pr[X_i^{(t)} \cap X_j^{(t)}] = 1/\Psi_6(t_i - 1)$. Therefore, $\Pr[X_i^{(t)} \cap X_j^{(t)}] \leq \Pr[X_i^{(t)}]\Pr[X_j^{(t)}]$. Therefore, $\mathbb{E}[X_i^{(t)}X_j^{(t)}] - \mathbb{E}[X_i^{(t)}]\mathbb{E}[X_j^{(t)}] \leq 0$.

In the case that $t_{e_1^j} < t_{e_2^j} < t_{e_3^j} < t_c < t_{e_2^i} < t_{e_3^i} < t_{e_4^i}$, events $X_j^{(t)}$ and $X_i^{(t)}$ are independent. Therefore, we have $\mathbb{E}[X_i^{(t)}X_j^{(t)}] - \mathbb{E}[X_i^{(t)}]\mathbb{E}[X_j^{(t)}] = 0$.

If the common edge c is the last edge of j^{th} butterfly but second or third edge of the i^{th} butterfly appear before t_c, we have $\Pr[X_i^{(t)}|X_j^{(t)}] \leq \Pr[X_i^{(t)}]$. This is because at time t_c, all first three edges of j^{th} butterfly should be maintained in the reservoir. It decreases the probability of selecting and maintaining edges of i^{th} butterfly in $\mathcal{R}$, which arrive before the edge c. Then, we have $\mathbb{E}[X_i^{(t)}X_j^{(t)}] - \mathbb{E}[X_i^{(t)}]\mathbb{E}[X_j^{(t)}] \leq 0$ for this case.

In the case that the common edge c is not the last edge among edges of j^{th} butterfly and is the first edge of i^{th} butterfly, we considered the following events: (1) Let E_1 denote that the common edge c is not deleted from the reservoir between the time steps t_j and $t_{e_2^i} - 1$. (2) Suppose E_2 indicate that the e_2^i is selected and inserted to $\mathcal{R}$, and the edge c is not removed from $\mathcal{R}$. (3) Let E_3 denote the common edge c and the edge e_2^i are maintained in $\mathcal{R}$ between time steps $t_{e_2^i} + 1$ and $t_{e_3^i} - 1$. (4) An event E_4 happens when the edge e_3^i is inserted to $\mathcal{R}$, and edges c and at that time step, e_2^i are are not removed from $\mathcal{R}$. (5) Lastly, an event E_5 shows that none of the edges c, e_2^i, or e_3^i are removed from $\mathcal{R}$ till time step $t_i - 1$. In the

$$\Pr[E_1|X_j^{(t)}] = \prod_{z=\max\{t_j, M+1\}}^{t_{e_2^i}-1} \left(\left(1 - \frac{M}{z}\right) + \frac{M}{z}\left(\frac{M-1}{M}\right)\right)$$

$$= \prod_{z=\max\{t_j, M+1\}}^{t_{e_2^i}-1} \frac{z-1}{z} = \frac{\max\{t_j, M\}}{\max\{M, t_{e_2^i}-1\}}$$

$$\Pr[E_2|E_1 \cap X_j^{(t)}] = \frac{M}{\max\{t_{e_2^i}, M\}} \frac{M - \sigma(t_{e_2^i})}{M}$$

$$\Pr[E_3|E_2 \cap E_1 \cap X_j^{(t)}]$$

$$= \prod_{z=\max\{t_{e_2^i}+1, M+1\}}^{t_{e_3^i}-1} \left(\left(1 - \frac{M}{z}\right) + \frac{M}{z}\left(\frac{M-2}{M}\right)\right)$$

$$= \prod_{z=\max\{t_{e_2^i}+1, M+1\}}^{t_{e_3^i}-1} \frac{z-2}{z}$$

$$= \frac{\max\{t_{e_2^i}, M\}\max\{t_{e_2^i}-1, M-1\}}{\max\{t_{e_3^i}-2, M-1\}\max\{t_{e_3^i}-1, M\}}$$

$$\Pr[E_4|E_3 \cap E_2 \cap E_1 \cap X_j^{(t)}] = \frac{M}{\max\{t_{e_3^i}, M\}} \frac{M - 2 \cdot \sigma(t_{e_3^i})}{M}$$

$$\Pr[E_5|E_4 \cap E_3 \cap E_2 \cap E_1 \cap X_j^{(t)}] =$$

$$\prod_{z=\max\{t_{e_3^i}+1, M+1\}}^{t_i-1} \left(\left(1 - \frac{M}{z}\right) + \frac{M}{z}\left(\frac{M-3}{M}\right)\right)$$

$$= \prod_{z=\max\{t_{e_3^i}+1, M+1\}}^{t_i-1} \frac{z-3}{z}$$

$$= \frac{\max\{t_{e_3^i}, M\}\max\{t_{e_3^i}-1, M-1\}\max\{t_{e_3^i}-2, M-2\}}{\max\{t_i-3, M-2\}\max\{t_i-2, M-1\}\max\{t_i-1, M\}}$$

$$\Pr[X_i^{(t)}|X_j^{(t)}] = \prod_{i=1}^{5} \Pr[E_i| \bigcap_{j=1}^{i-1} E_j \cap X_j^{(t)}]$$

$$\leq \frac{1}{\Psi_3(t_i-1)} \frac{\max\{X_j^{(t)}, M\}}{M}$$

$$\Pr[X_i^{(t)} \cap X_j^{(t)}] \leq \frac{1}{\Psi_3(t_j-1)\Psi_3(t_i-1)} \frac{\max\{X_j^{(t)}, M\}}{M}$$

Therefore, we get $\mathbb{E}[X_i^{(t)}X_j^{(t)}] \leq \frac{\max\{X_j^{(t)}, M\}}{M}$

Here, we consider the case that two i^{th} and j^{th} butterflies have two edges c_1 and c_2 in common. Let an edge x be the first edge of i^{th} butterfly which is not shared with the j^{th} butterfly. For the case that $t_x > t_j$, we define the following events to compute $\mathbb{E}[X_i^{(t)}X_j^{(t)}]$: (1) Let E_1 denote the event that the common edges c_1 and c_2 are removed from $\mathcal{R}$ among the time steps

t_j and $t_x - 1$. (2) Let E_2 denote the event that the edge x is added to the reservoir while c_1 and c_2 are not discarded from the reservoir at time t_x. (3) An event E_3 indicates that the edges all c_1, c_2 and x are maintained in $\mathcal{R}$ between the time steps $t_x + 1$ and $t_j - 1$. Therefore, we have

$$\Pr[E_1|X_j^{(t)}] = \prod_{z=\max\{t_j, M+1\}}^{t_x-1} \left(\left(1 - \frac{M}{z}\right) + \frac{M}{z}\left(\frac{M-2}{M}\right)\right)$$

$$= \prod_{z=\max\{t_j, M+1\}}^{t_x-1} \frac{z-2}{z}$$

$$= \frac{\max\{t_i, M\}\max\{t_i - 1, M-1\}}{\max\{M, t_x - 1\}\max\{M-1, t_x - 2\}}$$

$$\Pr[E_2|E_1 \cap X_j^{(t)}] = \frac{M}{\max\{t_x, M\}} \frac{M - 2 \cdot \sigma(t_x)}{M}$$

$$\Pr[E_3|E_2 \cap E_1 \cap X_j^{(t)}]$$

$$= \prod_{z=\max\{t_{e_3^i}+1, M+1\}}^{t_i-1} \left(\left(1 - \frac{M}{z}\right) + \frac{M}{z}\left(\frac{M-3}{M}\right)\right)$$

$$= \prod_{z=\max\{t_x+1, M+1\}}^{t_i-1} \frac{z-3}{z}$$

$$= \frac{\max\{t_x, M\}\max\{t_x - 1, M-1\}\max\{t_x - 2, M-2\}}{\max\{t_i - 3, M-2\}\max\{t_i - 2, M-1\}\max\{t_i - 1, M\}}$$

$$\mathbb{E}[X_i^{(t)} X_j^{(t)}] \le \frac{\max\{t_i, X_i^{(t)}\}\max\{t_i - 1, X_i^{(t)} - 1\}}{(X_i^{(t)} - 2)(X_i^{(t)} - 1)}$$

$$= \frac{\max\{t_i(t_i - 1), M(M-1)\}}{(M-2)(M-1)}$$

In the other case, when the edge x of the ith butterfly arrives before t_{c2}, the presence of edges of j in $\mathcal{R}$ decreases the probability of choosing and inserting edge x to the reservoir. Therefore, the probability of event $X_i^{(t)} \cap X_j^{(t)}$ is smaller than the probability of events in the previous case. Therefore, we can derive when two ith and jth butterflies share two edges $\mathbb{E}[X_i^{(t)} X_j^{(t)}] \le \frac{\max\{t_i(t_i-1), M(M-1)\}}{(M-2)(M-1)}$, where $i > j$.

Proof of ??

Proof. Let $p^{(t)}$ be the sampling probability of **??** at time t. Suppose that butterflies in $\mathbb{X}^{(t)}$ are numbered from 1 to $|\mathbb{X}^{(t)}|$. Let $X_i^{(t)} (1 \le i \le |\mathbb{X}^{(t)}|)$ be a random variable and is 1 if the i^{th} butterfly in $\mathbb{X}^{(t)}$ appears in the reservoir $\mathcal{R}$, otherwise $X_i^{(t)}$ is 0. Let $X^{(t)} = \sum_{i=1}^{|\mathbb{X}^{(t)}|} X_i^{(t)}$. Therefore, we have $Y_{\text{ADA}}^{(t)} = (p^{(t)})^{-4} \cdot X^{(t)}$.

In the case that $t \le M$, all edges of the stream are sampled. In this case $p^{(t)}$ is 1 and $X^{(t)} = |\mathbb{X}^{(t)}|$. Thus, $Y_{\text{ADA}}^{(t)} = |\mathbb{X}^{(t)}|$.

If $t > M$, each edge appears in the reservoir $\mathcal{R}$ with probability $p^{(t)}$. Therefore, we get

$$\mathbb{E}\left[X_i^{(t)}\right] = \Pr\left[X_i^{(t)} = 1\right] = \left(p^{(t)}\right)^4$$

With the linearity of expectation, we get $\mathbb{E}\left[X^{(t)}\right] = \sum_{i=1}^{|\mathbb{X}^{(t)}|} \mathbb{E}\left[X_i^{(t)}\right] = |\mathbb{X}^{(t)}| \times \left(p^{(t)}\right)^{-4}$. As $Y_{\text{ADA}}^{(t)} = (p^{(t)})^{-4} \cdot X^{(t)}$, then we get $\mathbb{E}[Y_{\text{ADA}}^{(t)}] = |\mathbb{X}^{(t)}|$.

Proof of ??.

$$\mathbb{V}\text{ar}[Y_{\text{ADA}}^{(t)}] = \mathbb{V}\text{ar}\left[(p^{(t)})^{-4} \sum_{i=1}^{|\mathbb{X}^{(t)}|} X_i^{(t)}\right]$$

$$= (p^{(t)})^{-8}\left[\sum_{i=1}^{|\mathbb{X}^{(t)}|} \mathbb{V}\text{ar}[X_i^{(t)}] + \sum_{i \ne j} \mathbb{C}\text{ov}[X_i^{(t)}, X_j^{(t)}]\right]$$

$$= |\mathbb{X}^{(t)}|((p^{(t)})^{-4} - 1) + (p^{(t)})^{-8}\left[\sum_{i \ne j} \mathbb{C}\text{ov}[X_i^{(t)}, X_j^{(t)}]\right]$$

$$= |\mathbb{X}^{(t)}|((p^{(t)})^{-4} - 1) + (p^{(t)})^{-8}\left[\sum_{i \ne j}\left(\mathbb{E}[X_i^{(t)} X_j^{(t)}] - \mathbb{E}[X_i^{(t)}]\mathbb{E}[X_j^{(t)}]\right)\right]$$

If i^{th} and j^{th} butterflies do not share any edge, we have:

$$\mathbb{E}[X_i^{(t)} X_j^{(t)}] = (p^{(t)})^4 \Pr(X_i^{(t)}|X_j^{(t)}) = (p^{(t)})^8$$

Therefore, in this case, $(p^{(t)})^{-8} \times (\mathbb{E}[X_i^{(t)} X_j^{(t)}] - \mathbb{E}[X_i^{(t)}]\mathbb{E}[X_j^{(t)}]) = 0$.

If i^{th} and j^{th} butterflies share exactly one edge we have:

$$\mathbb{E}[X_i^{(t)} X_j^{(t)}] = (p^{(t)})^4 \Pr(X_i^{(t)}|X_j^{(t)}) = (p^{(t)})^7$$

Therefore, we derive $(p^{(t)})^{-8} \times (\mathbb{E}[X_i^{(t)} X_j^{(t)}] - \mathbb{E}[X_i^{(t)}]\mathbb{E}[X_j^{(t)}]) = (p^{(t)})^{-1} - 1$.

If i^{th} and j^{th} butterflies have two edges in common we get:

$$\mathbb{E}[X_i^{(t)} X_j^{(t)}] = (p^{(t)})^4 \Pr(X_i^{(t)}|X_j^{(t)}) = (p^{(t)})^6$$

Therefore, we derive $(p^{(t)})^{-8} \times (\mathbb{E}[X_i^{(t)} X_j^{(t)}] - \mathbb{E}[X_i^{(t)}]\mathbb{E}[X_j^{(t)}]) = (p^{(t)})^{-2} - 1$.

By definition of $h_z^{\text{ADA}}(t)$ and p_z, it is verified that $\mathbb{V}\text{ar}[Y_{\text{ADA}}^{(t)}] = |\mathbb{X}^{(t)}|((p^{(t)})^{-4} - 1) + 2 \times \sum_{z=0}^{2} p_z \cdot h_z^{\text{ADA}}(t)$

Timestamp-based Sliding Window Butterfly Counting (TimeSW). Comparing to the sequence-based sliding window, counting the number of butterflies in timestamp-based sliding window is more challenging, due to the number of edges in window can vary over different time. At one timestamp, there may arrive either a burst of many edges or even no edges at all. In fact, the sequence sliding window can be seen as a special case of timestamp sliding window such that at each timestamp, there is exactly one edge arrives from the stream. In many cases, we want to answer the queries with variable length histories, where the window length is only known during the query time, but with upper bound W_{max}. Our algorithm maintains T different queues, each contains edge samples from recent stream, the value of T is related with W_{max} and will be decided later. The sampling rates of different queues are ranging from $1, 1/2, 1/4, \ldots, 1/2^{T-1}$. Each queue is with the same capacity M edges and once the queue is full, it will pop out the least recent edge and add the new edge. Thus to answer a query, the number of queues T should be no less than $\lceil 1 + \log_2 W_{max} - \log_2 M \rceil$.

Algorithm 6: TimeSW ($\mathcal{S}$, M, W_{max}): Time-based SW

Input: Edge stream $\mathcal{S}$, sampled edge set size M (≥ 8), maximum window size W_{max} ($\geq M$)

Output: $|\bowtie_w^{(t)}|$, estimate number of butterflies in time window w at time c

```
1  T ← ⌈1 + log2 W_max − log2 M⌉
   // Maintain T sampled edge sets, R_1, ..., R_T; d_i
      records the time of most recent discarded
      edge in R_i
2  ∀i ≤ T, R_i ← ∅, d_i ← 0
3  for each edge e from stream S at time t do
4      l ← 0
5      do
6          R_ℓ ← R_ℓ ∪ (e, t)
7          if |R_ℓ| > M then
8              R_ℓ ← R_ℓ \ (e', t') s.t. t' = min_{t∈R_ℓ} t
9              d_i ← t
10         ℓ ← ℓ + 1
11     while coin (1/2) is Head
       // Upone a query at time c, window size w
12     ℓ_* ← argmin_i{d_i | d_i ≥ (c − w)}
13     A ← {(e, t') ∈ R_{ℓ_*} s.t. t' > (c − w)}
14     |⋈_w^{(t)}| ← ⋈(A)/2^{4(ℓ_*−1)}
```

Lemma 8. *Suppose the maximum window size given by the query is W_{max}, Algorithm 6 uses space $O(M \cdot \log W_{max})$. The returned butterflies $|\bowtie_w^{(t)}|$ is unbiased.*

Proof. Algorithm 6 maintains $O(\log W_{max})$ queues and each has capacity M, the space cost follows. Once a query on butterfly number in window W is received, Algorithm 6 selects the sampled edge set at level ℓ_* where each edge is sampled by probability 2^{ℓ_*-1}, the estimate number of butterflies is unbiased.